Introduction

The following guide describes how to visualize warehouse process data in different chart types with the Easy Graphics Framework.

To start with, look at the example implementation delivered with standard EWM, which gives a basic overview of how static data can be visualized in an EGF chart with the most common chart types.

**Transaction:** /SCWM/EGF  
**Cockpit:** DEMO_COCKPIT  
**Object:** DEMO_OBJECT  
**Service Provider Implementation Class:** /SCWM/CL_EGF_DEMO  
**Implemented Methods:**  
/SCWM/IF_EGF_SP~GET_DATA  
/SCWM/IF_EGF_SP~DRILL_DOWN

![EGF Demo Cockpit](image)

**Screen 1:** Screenshot of the EGF DEMO_COCKPIT with four instances of the object DEMO_OBJECT in chart types: vertical bars, speedometer, traffic light, time scatter and with the drill-down function enabled, showing a gantt chart in a dialog box.

With this implementation guide, we focus more closely on the details and build an EGF cockpit showing real-time warehouse data in charts that are customized to look more attractive. The result will look like the following screenshot:
Show real-time warehouse monitoring data within an EGF cockpit: Processing Steps

To get a cockpit that looks like this one, following steps are necessary:

1) *What* is shown *how*? Decide which business data is to be visualized in which chart types.
2) What input data is necessary to get your business data? Choose input parameters and create a suitable input structure.
3) Create the implementation class for the service provider interface /SCWM/IF_EGF_SP
4) Set up EGF objects and cockpit in the Customizing
5) Implement the service provider
6) Customize the charts’ look and feel: Make more attractive charts using SAP Netweaver Chart Designer and apply the chart templates to the EGF objects with transaction /SCWM/EGF_CHART.
7) Lay out the cockpit and set default parameters

Screen 2: EGF objects to show LM Planning Results as horizontal/vertical bars and in the Business Explorer as URL charts
1. Deciding What to Show in the EGF Cockpit

1. The managers of the warehouse want to get an overview of the labor management planning data compared with the actual workload the processors in the different activity areas have to handle.

![Capacity Evaluation 02.06.2010](chart1)

![Resources per Activity](chart2)

2. Additionally, they want to see the labor management data in the Business Explorer (Crystal Reports), which we can show as a URL in the EGF Cockpit.

![Labor Management](chart3)

3. Ideally, managers would get an alert if a specific activity area is totally underplanned. This can be visualized in the EGF with chart type ‘traffic light’.

4. From the chart showing the planning data, managers want to drill down to the planning transaction /SCWM/PL LM Planning and Simulation, to plan further.

The basis for the EGF charts is the results from the LM Planning shown in transaction /SCWM/PL_LOAD Load LM Planning and Simulation Results for a specific planning variant.
Screen 3: Transaction /SCWM/PL_LOAD: Result of a planning formula (required time to handle actual workload), planned duration of a workload record, number of planned processors to handle the actual workload in the according activity area.

This data is to be shown in two horizontal/vertical bar charts. The chart ‘Capacity Evaluation’ shows the comparison of the data in the Result and Planned Duration columns, converted from minutes to hours. The chart ‘Resource per Activity’ gives an overview of the available resources for each activity area.

To get the data for the charts, we need the same data as shown in the ALV of transaction /SCWM/PL_LOAD. This transaction retrieves its business data using function module /SCWM/PLAN_READ; we should reuse that.

```FUNCTION /scwm/plan_read.
** IMPORTING
**  REFERENCE(IV_LGNUM) TYPE /SCWM/LGNUM
**  REFERENCE(IR_SAVE_ID) TYPE RSELOPT
**  REFERENCE(IR_SAVED_AT) TYPE RSELOPT
**  REFERENCE(IR_USER) TYPE RSELOPT
** EXPORTING
**  REFERENCE(ET_PLAN_AGG) TYPE /SCWM/TT_ASP_PL_OIP
**  REFERENCE(ET_PLAN_MS) TYPE /SCWM/TT_ASP_PL_ODP_MS
```
2. Specifying Necessary Input Data

The data selection for the EGF object is thus similar to that of transaction /SCWM/PL_LOAD. That is, we provide the selection option for warehouse number and planning variant (SAVE_ID), and additionally the simulation version, to the caller of our EGF object, as shown in the following screenshot:

![Selection screen shown when calling the object DEMO_FORMO_LM for display](image)

To get this selection screen, input data needs to be provided to the EGF object. If no existing structure type is applicable, create a structure type in the dictionary for that purpose.

| Structure Type: ZGRY_S_EGF_LMLOAD – Structure for LM Load Planning Variant Selection in EGF |
|---|---|
| Component | Component Type |
| LNUM | /SCWM/LNUM |
| SAVE_ID | /SCWM/DE_PL_ID |
| SIM_VER | /SCWM/DE_PL_NUM_SIM |
Screen 5: EGF Object - Input parameters for EGF object DEMO_FORMO_LM. Customizing path: Extended Warehouse Management -> Monitoring -> EGF -> Define Objects
3. Creating the Service Provider Implementation Class

In order to proceed with the EGF setup in Customizing and name the EGF object service provider, the implementation class for the service provider, based on the interface /SCWM/IF_EGF_SP, is required. Create and activate it now, and how to implement it will be explained in step 5.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET_DATA</td>
<td>Gets chart data from the application. Most charts require a group of row values and a group of data categories as data format. A simple chart, such as a traffic light, only sets the color of the traffic light.</td>
</tr>
<tr>
<td>GET_URL</td>
<td>Gets a URL from the application. This URL links to the graphic that the EGF is to display.</td>
</tr>
<tr>
<td>HANDLE_FUNCTION</td>
<td>If the application receives a function code for a follow-up activity, this interface method enters the function code for the chart.</td>
</tr>
<tr>
<td>DRILL_DOWN</td>
<td>Specifies what to do if the user chooses (drills down into) the corresponding EGF object.</td>
</tr>
<tr>
<td>OBJECT_REMOVED</td>
<td>Implements a notification of the event if the object is removed from the display.</td>
</tr>
<tr>
<td>SET_FUNCTIONS</td>
<td>Adds functions dynamically for an object which are then displayed in the object’s context menu and can be handled in method HANDLE_FUNCTION.</td>
</tr>
</tbody>
</table>

Interface: /SCWM/IF_EGF_SP
4. Setting Up EGF in Customizing – Objects, Cockpits and Functions


1. Define the EGF object and enter the service provider implementation class, the default chart type and all other supported chart types (Subnode Charts), for example 003 - horizontal bars and 016 . traffic light.

   ![Screen 6: EGF Object: Defining the default chart type for object DEMO_FORMO_LM](image)

2. Define a new cockpit and cockpit input data, or choose an existing cockpit to assign the new EGF object to.
3. Assign the new EGF object to a cockpit.

Screen 7: EGF Cockpit: Typical input data for a cockpit: the warehouse number
Customizing path: Extended Warehouse Management -> Monitoring > Easy Graphics Framework -> Define Cockpits

Screen 8: Add EGF object DEMO_FORMO_LM to cockpit DEMO_CH_EGF
Screen 9: Overview of all EGF objects assigned to cockpit DEMO_CH_EGF

Screen 11: Customizing view for the EGF object definition

4. Start the EGF Cockpit, transaction /SCWM/EGF, and maintain the input data (warehouse number) to see the objects.

Screen 12: Transaction /SCWM/EGF – cockpit with predefined objects
As the service provider is not implemented yet, calling the object for display will not show a chart but instead an error message: *Graphic data not transferred (Message no. /SCWM/EGF002)*

So the next step is to implement the service provider.
5. Implementing the Service Provider

The EGF receives data from the service provider and if you are using a chart type to display the data that is delivered in the standard SAP system, the EGF sends the data to SAP IGS (SAP Internet Graphics Server), which generates the graphics. This chapter describes how to build up this data in the service provider implementation.

The implementation consists of two parts:
1. Retrieving the business data to be shown in the charts
2. Adapting the business data to fit the supported chart types

The export objects of method /SCWM/IF_EGF_SP~GET_DATA are to be filled. These methods are:
- EO_CHART_DATA (type /SCWM/CL_EGF_CHART_DATA)
- EO_TABLE_DATA (type /SCWM/CL_EGF_TABLE_DATA)

You can implement the business logic for different objects within one service provider implementation, by delegating to different methods, for example according to the IV_OBJECT_ID (the identifier of the EGF object, specified in the EGF Customizing) as shown in the following screenshot.

Screen 13: Signature and implementation of method /SCWM/IF_EGF_SP~GET_DATA
5.1. Getting the business data

In our example, we want to visualize the data which we can select in the transaction /SCWM/PL_LOAD (Load LM Planning and Simulation Results) in the EGF cockpit, as shown in the following screenshot.

The idea is to illustrate this data in two EGF objects:

- Capacity Evaluation: a comparison of planned (Planned Dur.) and actual (Result) workload in hours for each activity area
- Resources per Activity: the available resources (NoProcess.) for the work to be done for each activity area

We checked how the transaction /SCWM/PL_LOAD retrieves the data and found out that the function module /SCWM/PLAN_READ carries out this function. This is the function module we can also use to get the data for the EGF object implementation, but for that it must be provided with reasonable input data.

```
FUNCTION /scwm/plan_read.
+ " IMPORTING
+ " REFERENCE (IV_LGNUM) TYPE /SCWM/LGNUM
+ " REFERENCE (IR_SAVE_ID) TYPE RSELOPTTION
+ " REFERENCE (IR_SAVED_AT) TYPE RSELOPTTION
+ " REFERENCE (IR_USER) TYPE RSELOPTTION
+ " EXPORTING
+ " REFERENCE (ET_PLAN_AGG) TYPE /SCWM/TT_ASP_PL_OIP
+ " REFERENCE (ET_PLAN_MS) TYPE /SCWM/TT_ASP_PL_ODP_MS
```

Signature of Function Module /SCWM/PLAN_READ

We provide the import parameters for the function module with the input data of the EGF cockpit and EGF object. The parameter IV_LGNUM (warehouse number) is already provided with the cockpit input (see Screen 7) and can therefore be retrieved from the input parameter IT_COCKPIT_INPUT:
For just providing the input parameters for the function module /SCWM/PLAN_READ (LGNUM, SAVE_ID, SAVED_AT, USER), we could reuse the existing structure /SCWM/S_ASPQ_PL_S as the input structure for defining the EGF object’s input data. However, we also want to filter the result by the simulation version (see Screen 12, field Sim. Vers.) and could not find a suitable structure for this. Therefore we created our own input structure (see Screens 4 and 5) and used it to get the input (IT_OBJECT_INPUT) for calling function module /SCWM/PLAN_READ as shown below.

```
DATA:
  ls_frange_t_read TYPE /scwm/s_ds_selopt_field,
  ls_selection TYPE /scwm/s_ds_selopt_table.
reading warehouse number
READ TABLE it_cockpit_input
  WITH KEY tablename = '/SCWM/T340D'
  INTO ls_selection.
READ TABLE ls_selection-frange_t
  WITH KEY fieldname = 'LGNUM'
  INTO ls_frange_t_read.

IF ls_frange_t_read-selopt_t IS NOT INITIAL.
  READ TABLE ls_frange_t_read-selopt_t
    INDEX 1
    INTO ls_selopt.
  lv_lgnum = ls_selopt-low.
ENDIF.

READ TABLE it_object_input
  WITH KEY tablename = 'ZGRY_S_EGF_LMLOAD'
  INTO ls_selection.
LOOP AT ls_selection-frange_t INTO ls_frange_t_read.
  IF ls_frange_t_read-selopt_t IS NOT INITIAL.
    READ TABLE ls_frange_t_read-selopt_t
      INDEX 1
      INTO ls_selopt.
    CASE ls_frange_t_read-fieldname.
      WHEN /scwm/if_ui_pl_c=>sc_field_lgnum.
        lv_lgnum = ls_selopt-low.
      WHEN /scwm/if_ui_pl_c=>sc_field_s_id.
        APPEND ls_selopt TO lr_save_id.
        lv_saveid = ls_selopt-low.
      WHEN /scwm/if_ui_pl_c=>sc_field_simver.
        lv_simver = ls_selopt-low.
    ENDIF.
  ENDIF.
ENDLOOP.
```

The function module returns the business data with table LT_OIP which we now need to adapt for the EGF object display in the different chart types.

5.2. Adapting the business data for chart display - data series and categories

Now that we have retrieved the business data to work with, we need to adapt it for the export objects of the method /SCWM/IF_EGF_SP~GET_DATA:
• EO_CHART_DATA (type /SCWM/CL_EGF_CHART_DATA)
• EO_TABLE_DATA (type /SCWM/CL_EGF_TABLE_DATA)

What makes the visualization of business data difficult for the different chart types is the understanding of what needs to be passed as series and what as categories to the export object EO_CHART_DATA (TYPE REF TO /SCWM/CL_EGF_CHART_DATA). To understand that, it is useful to have a look at the definition of EO_CHART_DATA, and the different chart types:

<table>
<thead>
<tr>
<th>Object/Method</th>
<th>Input Parameter</th>
<th>Input Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EO_CHART_DATA</td>
<td>/SCWM/CL_EGF_CHART_DATA</td>
<td></td>
</tr>
<tr>
<td>ADD_CATEGORY</td>
<td>IV_CATEGORY</td>
<td>/SCWM/DE_EGF_LABEL</td>
</tr>
<tr>
<td>ADD_SERIES_*</td>
<td>IV_LABEL</td>
<td>/SCWM/DE_EGF_LABEL</td>
</tr>
<tr>
<td></td>
<td>IV_ID</td>
<td>/SCWM/DE_EGF_ID</td>
</tr>
<tr>
<td></td>
<td>IS_POINT</td>
<td>/SCWM/S_EGF_POINT_TGENERAL</td>
</tr>
<tr>
<td>SET_CHART_SERIES</td>
<td>IT_SERIES</td>
<td>/SCWM/TT_EGF_CHART_DATA</td>
</tr>
<tr>
<td>SET_CHARTCATEGORY</td>
<td>ITCATEGORY</td>
<td>/SCWM/TT_EGF_CHART_DATA</td>
</tr>
<tr>
<td>SET_CHART_TITLE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 1:** Signature of class /SCWM/CL_EGF_CHART_DATA

**Figure 2:** Attributes of object EO_CHART_DATA to be populated with values in the service provider implementation
5.2.1 SAP IGS Chart Types

**Horizontal/Vertical Bar Chart**

- Horizontal Bars
- Category 1: Series 1, Series 2, Series 3, Series 4
- Category 2: Series 1, Series 2, Series 3, Series 4
- Category 3: Series 1, Series 2, Series 3, Series 4
- Category 4: Series 1, Series 2, Series 3, Series 4
- Category 5: Series 1, Series 2, Series 3, Series 4

**Speedometer Chart**

- Speedometer
- Category 1: 10, 20, 30, 40, 50
- Category 2: 10, 20, 30, 40, 50
- Category 3: 10, 20, 30, 40, 50
- Category 4: 10, 20, 30, 40, 50
- Category 5: 10, 20, 30, 40, 50

**Gantt Chart**

- Gantt
In our example, we want to show the values as horizontal and vertical bars:

From the definition of what series and categories are in horizontal/vertical bar charts, it is as follows:

The differently colored bars (Required/Available Capacity and No. of Processors) are seen as series. Each activity area is a category. For the chart ‘Capacity Evaluation’, each category consists of the two series.

In the service provider implementation, the assignment of values to series and categories, plus label definition, is realized as follows:
DATA:
  lt_oip TYPE /scwm/ttp_asp_pl_oip,
  lo_chart_data TYPE REF TO /scwm/cl_egf_chart_data,
  lo_table_data TYPE REF TO /scwm/cl_egf_table_data,
  ls_data_pldura TYPE /scwm/s_egf_point_general,
  ls_data_result TYPE /scwm/s_egf_point_general,
  lv_saveid TYPE string,
  lv_simver TYPE string,
  lv_overplanned_aa TYPE string,
  lv_label TYPE /scwm/de_egf_label.

CREATE OBJECT lo_chart_data.
CREATE OBJECT lo_table_data.

LOOP AT lt_oip ASSIGNING <ls_oip> WHERE sim_ver = lv_simver.
  ls_data_pldura-label = 'Required Capacity'.
  ls_data_pldura-value = <ls_oip>-pldura / 60. "value pldura in minutes converted to hours
  ls_data_pldura-id = 'PLDURA'.
  ls_data_result-label = 'Available Capacity'.
  ls_data_result-value = <ls_oip>-plan_result / 60. "value pi.result in minutes conv. to hours
  ls_data_result-id = 'PLAN_RESULT'.

CASE lv_chart_type.
  WHEN /scwm/if_egf_sp=>sc_chart_type_bars OR
    /scwm/if_egf_sp=>sc_chart_type_columns.
    IF <ls_oip>-procst IS NOT INITIAL.
      CONCATENATE <ls_oip>-aarea <ls_oip>-procst INTO lv_label SEPARATED BY space.
    ELSE.
      CONCATENATE 'Act.Area:' <ls_oip>-aarea 'Process:' <ls_oip> procs INTO lv_label SEPARATED BY space.
    ENDIF.

    CALL METHOD lo_chart_data->add_category "category 'Activity Area/Process'
      EXPORTING
        iv_category = lv_label.
    .

    CALL METHOD lo_chart_data->add_series_general "series 'planned duration, requ. capacity'
      EXPORTING
        iv_label = ls_data_pldura-label
        ls_point = ls_data_pldura
        iv_id = ls_data_pldura-id.
    CLEAR: ls_data_pldura.

    CALL METHOD lo_chart_data->add_series_general "series 'result', available capacity
      EXPORTING
        iv_label = ls_data_result-label
        ls_point = ls_data_result
        iv_id = ls_data_result-id.
    CLEAR: ls_data_result.
  WHEN /scwm/if_egf_sp=>sc_chart_type_traffic_light.
    set traffic light to red, if available capacity is lower than required capacity
    IF ls_data_result-value < ls_data_pldura-value .
      CONCATENATE 'Act.Area:' <ls_oip>-aarea 'Process:' <ls_oip>-procs INTO lv_overplanned_aa
      SEPARATED BY space.
    ENDIF.
  END_CASE.
END_LOOP.

ls_title-caption = 'Capacity Evaluation 02.06.2010'.
lo_chart_data->set_chart_title( ls_title ).
CONCATENATE 'Planning Variant:' lv_saveid INTO ls_subtitle-caption SEPARATED BY space.
lo_chart_data->set_chart_subtitle(' ls_subtitle ).
CASE iv_chart_type.

    WHEN /scwm/if_egf_c=>sc_chart_type_bars
        OR /scwm/if_egf_c=>sc_chart_type_columns.
        * creating and setting headline
        CALL METHOD lo_chart_data-->set_chart_title
            EXPORTING
                is_title = ls_title.
    WHEN /scwm/if_egf_sp=>sc_chart_type_traffic_light.
        * setting traffic light to red, if available capacity is lower than required capacity
        IF lv_overplanned_aa IS INITIAL.
            ls_subtitle-caption = 'No activity area is overplanned.'.
            CALL METHOD lo_chart_data-->set_traffic_light
                EXPORTING
                    iv_color = /scwm/if_egf_c=>sc_value_tl_green.
        ELSE.
            CONCATENATE 'Overplanned Activity Area: ' lv_overplanned_aa INTO ls_subtitle-caption
                SEPARATED BY space.
            lo_chart_data-->set_chart_subtitle( ls_subtitle ).
            CALL METHOD lo_chart_data-->set_traffic_light
                EXPORTING
                    iv_color = /scwm/if_egf_c=>sc_value_tl_red.
        ENDIF.
        * customizing the look and feel of the traffic light (blue background instead of standard black
        ls_ampel_bg-transparency = abap_true.
        ls_ampel_bg-color = 'RGB(236,244,249)'.
        lo_chart_data-->set_chart_background( ls_ampel_bg ).
    END_CASE.

* returning data to EGF
    eo_chart_data = lo_chart_data.
Table Chart

For the chart type 'Table', which simply shows the data in an ALV (and makes an EGF graphic chart accessible), the data to be shown in the table must be passed to the object EO_TABLE_DATA (type /SCWM/CL_EGF_TABLE_DATA) with the following methods:

<table>
<thead>
<tr>
<th>Object</th>
<th>EO_TABLE_DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>/SCWM/CL_EGF_TABLE_DATA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Input Parameter</th>
<th>Input Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET_DATA</td>
<td>IT_DATA</td>
<td>INDEX TABLE</td>
</tr>
<tr>
<td></td>
<td>IV_DISPLAY_STRUCTURE</td>
<td>/SCWM/DE_EGF_DISPLAY_STRUCTURE</td>
</tr>
<tr>
<td>SET_EXCLUDE_COLUMNS</td>
<td>IT_COLUMNS</td>
<td>/SCWM/TT_EGF_EXCLUDE_COLUMNS</td>
</tr>
<tr>
<td>SET_TITLE</td>
<td>IV_TITLE</td>
<td>/SCWM/DE_EGF_TEXT</td>
</tr>
<tr>
<td></td>
<td>IV_LARGE_TITLE</td>
<td>BOOLE_D (OPTIONAL)</td>
</tr>
</tbody>
</table>

Figure 3: Signature of class /SCWM/CL_EGF_TABLE_DATA

In our example, which has the ALV table shown in transaction /SCWM/PL_LOAD as a basis, the implementation of the table chart type can be made quite easily, meaning we just pass the result of the function module call to the table data object:

```plaintext
CASE iv_chart_type.
  WHEN /scwm/if_egf_sp=>sc_chart_type_table.
    * Set title
    CALL METHOD lo_table_data=>set_title
      EXPORTING
      iv_title = lv_table_title.
    *exclude columns of the display structure for display
    APPEND /scwm/if_ui_pl_c=>sc_field_s_name TO lt_ex_col.
    APPEND /scwm/if_ui_pl_c=>sc_field_s_at TO lt_ex_col.
    APPEND /scwm/if_ui_pl_c=>sc_field_s_id TO lt_ex_col.
    APPEND /scwm/if_ui_pl_c=>sc_field_simver TO lt_ex_col.
    APPEND /scwm/if_ui_pl_c=>sc_field_route TO lt_ex_col.
    lo_table_data=>set_exclude_columns( lt_ex_col ).
    DELETE lt_oip WHERE sim_ver <> lv_simver.
    CALL METHOD lo_table_data=>set_data
      EXPORTING
      it_data = lt_oip
      iv_display_structure = '/SCWM/S_ASP_PL_OIP'.
  ENDCASE.
```
Screen 15: EGF Chart Type Table

URL Chart

For showing web content inside an EGF object, for example analytical data prepared with Crystal Reports, visualized in the Business Explorer, the object is set up as a URL type chart and the method /SCWM/IF_EGF_SP~GET_URL must be implemented, generally just by adding the URL as a string to the method's export parameter EV_URL (TYPE W3URL).

```
Method /SCWM/IF_EGF_SP~GET_URL.
* shows Business Explorer in EGF Chart Type URL
endmethod.
```
Screen 16: BOBJ Business Explorer within the EGF cockpit
6. Customizing the EGF Charts’ Look and Feel

Setting up the EGF objects in the customizing and implementing the service provider does not directly lead to the charts look in the way as shown in the screenshots above. Instead, the graphics will be displayed according to the default setting of your system – somehow like in the following screenshot:

![Screen 17: Standard SAP system setting for EGF charts](image)

However, what we want to achieve is to get the EGF objects for ‘Capacity Evaluation’ and ‘Resources per Activity’ look more attractive, i.e. customize them in terms of colors, titles, legend, as shown in Screen 19.

![Screen 18: customized EGF graphics for the objects ‘Capacity Evaluation’ and ‘Resources per Activity’ in chart types horizontal/vertical bars](image)

The possibilities for changing the look of EGF graphics are:

1) **Recommended approach:** Create a chart template for the required chart types with the external tool SAP Chart Designer and apply the chart template to EGF object with transaction /SCWM/EGF_CHART (Process Chart Template)

2) All in one transaction - create the chart template and apply it to the EGF object in transaction /SCWM/EGF_CHART
3) Some UI settings can even be implemented in the EGF Service Provider interface method /SCWM/IF_EGF_SP#GET_DATA

6.1 Technical Background
The graphics in EGF are rendered by the IGS Chart Engine, which is fed by two XML Files:
- A customizing XML file, for the look of the graphics
- A data XML file, for the input data to be displayed

The customizing XML file contains the data applied to the EGF object when saving the settings of the chart template to an EGF object in transaction /SCWM/EGF_CHART.
The data XML file is generated by the EGF, basically based on the data you provide in the service provider implementation of the method /SCWM/IF_EGF_SP->GET_DATA.

Graphic settings which are not already customized in the chart template can also be defined (overridden) in the /SCWM/IF_EGF_SP->GET_DATA implementation by using the public methods of the class /SCWM/CL_EGF_CHART_DATA (see Figure 1).

Code snippet of the GET_DATA implementation:

```
DATA: lo_chart_data  TYPE REF TO  /scwm/cl_egf_chart_data,
     ls_axis       TYPE /scwm/s_egf_chart_axis.

CREATE OBJECT lo_chart_data.

*Setting category titles programmatically
lo_chart_data->add_category('In-Door1').

*Minimum and Maximum values for chart
ls_axis-min_value  =  0,
ls_axis-min_auto   =  'N'.
ls_axis-max_value  =  100.
ls_axis-max_auto   =  'N'.
CALL METHOD lo_chart_data->set_chart_x_axis_settings
         EXPORTING
             is_axis  =  ls_axis.
```

As a reference, see the EGF example implementation for the EGF object DEMO_OBJECT in the cockpit DEMO_COCKPIT:

Class: /SCWM/CL_EGF_DEMO
Method: READ_OBJECT_DATA in the /SCWM/IF_EGF_SP~GET_DATA implementation
6.2 Defining the EGF Chart Template

With the transaction /SCWM/EGF_CHART (Process Chart Template), the look of the different IGS chart types can be customized and then, either user-specifically or for all users, applied to the EGF objects. However, this tool is not that convenient to work with, so we recommend (also to customers) that you use the latest version of the SAP Chart Designer (based on IGS 6.40 or 7.00), which can be downloaded from the SDN and installed and run locally on your PC.

1. Start SAP Chart Designer and select the chart type you want to customize.

Screen 19: SAP Chart Designer: Specify chart type for creating a chart template
2. Step through the wizard to make all your settings

**Screen 20:** SAP Chart Designer: define to show the value of the series as label on the series bar
Screen 21: SAP Chart Designer: Choose SAP color palette (theme) ‘Tradeshow’
Screen 22: SAP Chart Designer: Choose horizontal position of the series' label
3. Finish the wizard and save your settings as an XML file

Go to your EWM system and start transaction /SCWM/EGF_CHART and import the XML file you defined in the external tool by choosing Import Template.
**Screen 24:** Transaction /SCWM/EGF_CHART – Process Chart Template: import the newly created SAP Chart Designer template

4. Save the imported Chart Template in order to apply it to the desired EGF cockpit/object.
5. Enter /SCWM/EGF and select the cockpit and object for which you defined the new settings. If the result is not what you expected, check whether you applied the chart template as Standard Setting, User-Specific or Cross-User, or check your Service Provider implementation, which might have overridden the settings of the chart template.
7. Customizing the Cockpit Layout

The cockpit layout is customizable. You can make the following changes:

- Change the grid from the standard two columns and rows to up to ten rows or columns to display more EGF objects
- Hide the object list tree
- Save and load layouts with the menu only, with grid layout or with the input data
- Set defaults via user parameters and thus avoid entering the same data repeatedly each time you enter the /SCWM/EGF transaction

Warehouse number: /SCWM/LGN

Warehouse cockpit : /SCWM/SG_COCKPIT

Screen 18: Save a cockpit layout

Screen 19: Load an existing cockpit layout
Important Disclaimers on Legal Aspects

SAP Library document classification: PUBLIC

This document is for informational purposes only. Its content is subject to change without notice, and SAP does not warrant that it is error-free. SAP MAKES NO WARRANTIES, EXPRESS OR IMPLIED, OR OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

Coding Samples

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

Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint where to find supplementary documentation. SAP does not warrant the availability and correctness of such supplementary documentation or the ability to serve for a particular purpose. SAP shall not be liable for any damages caused by the use of such documentation unless such damages have been caused by SAP's gross negligence or willful misconduct.

Accessibility

The information contained in the SAP Library documentation represents SAP's current view of accessibility criteria as of the date of publication; it is in no way intended to be a binding guideline on how to ensure accessibility of software products. SAP specifically disclaims any liability with respect to this document and no contractual obligations or commitments are formed either directly or indirectly by this document. This document is for internal use only and may not be circulated or distributed outside your organization without SAP's prior written authorization.

Gender-Neutral Language

As far as possible, SAP documentation is gender neutral. Depending on the context, the reader is addressed directly with "you", or a gender-neutral noun (such as “sales person” or “working days”) is used. If when referring to members of both sexes, however, the third person singular cannot be avoided or a gender-neutral noun does not exist, SAP reserves the right to use the masculine form of the noun and pronoun. This is to ensure that the documentation remains comprehensible.