Development Guide for
Custom RFC Pull Data Provider

June 2013
# TABLE OF CONTENTS

1 INTRODUCTION .................................................................................................................. 3
  1.1 Technical Pre-requisites ..................................................................................................... 4

2 CREATE DATA PROVIDER ........................................................................................................ 5
  2.1 Example Data Provider ....................................................................................................... 5
  2.2 Roadmap to Create a Custom Pull RFC Data Provider ...................................................... 6
  2.3 Define RFC Pull Data Collector Template in Solution Manager ............................................. 6
    2.3.1 Enter attributes for this DP ........................................................................................ 7
    2.3.2 Assign input parameters ............................................................................................. 8
    2.3.3 Assign Help Values ...................................................................................................... 10
    2.3.4 Save the data provider .................................................................................................. 10
  2.4 Create the Data Provider FM (Managed System) .................................................................. 11
    2.4.1 Stackable Calls ............................................................................................................ 13
    2.4.2 Test Function Module .................................................................................................. 13
    2.4.3 Source Code Part I: Extract parameter values .............................................................. 16
    2.4.4 Source Code Part II: Select Data .................................................................................. 18
    2.4.5 Source Code Part III: Process Data and Produce Result .............................................. 19
  2.5 Create Custom Template (Solution Manager) ...................................................................... 20
  2.6 Create New Metric Group for Data Provider (Solution Manager) ....................................... 21
  2.7 Assign Template to the Managed System and Activate it ..................................................... 24
  2.8 View Collected Metric ........................................................................................................ 26
  2.9 Create Alert for This Metric .............................................................................................. 28
    2.9.1 Create Alert .................................................................................................................. 28
    2.9.2 View Alert .................................................................................................................... 31

3 APPENDIX: SAMPLE CODE FOR DP FUNCTION MODULE .................................................. 32
1 Introduction

This guide provides a step by step guide to developing a customer pull RFC data provider (DP). Starting from Solution Manager 7.1, technical monitoring is based on the MAI framework (Monitoring and Alerting infrastructure). In MAI various DPs collect metric data from the managed object into Solution Manager. These metrics can be stored for reporting or further processed to produce events and alerts.

There are different types of Data Provider in the Solution Manager. This guide only focuses on RFC PULL DP. In this collection method, the data collection request is sent from Solution Manager to the Extraction Framework Extension (EFWK Extension) of the managed system (part of SAP Solution Tools Plugin ST-PI). The EFWK Framework in turn calls the individual DPs to collect the data and then sends them back to Solution Manager.

Technically data collection part of PULL RFC Data Provider is implemented as a remote-enabled FM in the managed system. The DP RFCs have the required import and export parameters dictated by the MAI Framework.
1.1 Technical Pre-requisites

Solution Manager:
Solution Manager 7.10 support package stack 5 – 8 + SAP note 1778870
or
Solution Manager 7.10 support package stack 9 and higher

Managed ABAP System:
ST-PI 2008_1_620 to 2008_1_710 SP06 – SP07 + SAP note 1778870
or
ST-PI 2008_1_620 to 2008_1_710 SP08 and higher
2 Create Data Provider

2.1 Example Data Provider

In this guide we will create a simple data provider to illustrate the end to end development steps. The purpose of the data provider is to collect the number of ABAP dumps for the selected clients, users and date. In ST-PI there is already a standard data provider DUMP_INFO which uses function module /SDF/E2E_DUMP_INFO to collect data. The example in this guide is a simplified and modified version of DUMP_INFO.

Inputs parameters of this example DP:

<table>
<thead>
<tr>
<th>Name</th>
<th>Select-Option?</th>
<th>Required Field?</th>
<th>Default value</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Yes</td>
<td>Yes</td>
<td>TOTAL*</td>
<td></td>
</tr>
<tr>
<td>User</td>
<td>Yes</td>
<td>Yes</td>
<td>TOTAL*</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>No</td>
<td>Yes</td>
<td>TODAY, YESTERDAY</td>
<td></td>
</tr>
</tbody>
</table>

*: TOTAL means to sum up all possible values, e.g. all clients or all users.

Expected output: number of ABAP dumps based on the input data.

To keep the example simple we will use Solution Manager as the managed system (i.e. self-monitoring) so that all steps can be done in the same system. However in the roadmap we will state whether the step should be performed in the Solution Manager or the managed system.
2.2 Roadmap to Create a Custom Pull RFC Data Provider

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Solution Manager</th>
<th>Managed System</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Define a RFC (Pull) Data Collector Template</td>
<td>http://solman_server:port/sap/bc/webdynpro/sap/wd_mai_dpc_main</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Create remote enabled FM in the managed system</td>
<td>Transaction SE80/SE37</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Copy custom template</td>
<td>Transaction SOLMAN_SETUP ➔ Technical Monitoring ➔ Step 4: Temple Maintenance</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Create a new metric for the new data provider</td>
<td>Transaction SOLMAN_SETUP ➔ Technical Monitoring ➔ Step 4: Temple Maintenance</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Assign template to the managed system and activate it</td>
<td>Transaction SOLMAN_SETUP ➔ Technical Monitoring ➔ Step 5 and 6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>View collected metrics</td>
<td>Technical Monitoring workcenter ➔ System Monitoring ➔ Select the managed system and open System Monitoring</td>
<td></td>
</tr>
</tbody>
</table>

2.3 Define RFC Pull Data Collector Template in Solution Manager

First of all, the data provider and its input parameters should be defined in the Solution Manager. This is done in the Data Collector Template of the Solution Manager.

Open your browser and enter URL: http://<solman_host>:<http_port>/sap/bc/webdynpro/sap/wd_mai_dpc_main to create a new DP under RFC (Pull). The screen shots are from Solution Manager 7.1 SP07 system. As of Solution Manager 7.1 SP09 this UI will look different.
2.3.1 Enter attributes for this DP

<table>
<thead>
<tr>
<th>Property field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Provider ID</strong></td>
<td>Custom data provider ID must prefixed by XX* YY* ZZ* to be preserved by template updates</td>
</tr>
<tr>
<td><strong>Data Provider Implementation</strong></td>
<td>The name of the remote enabled FM to be developed in the managed system. It should start with Y or Z for custom FM.</td>
</tr>
<tr>
<td><strong>Stackable</strong></td>
<td>If checked, the EFWK Extension in the managed system will collect all calls to the same data provider into a single call. See section 2.4.1 for more details.</td>
</tr>
</tbody>
</table>
2.3.2 Assign input parameters

In the wizard enter the values for each parameter.

<table>
<thead>
<tr>
<th>Tech Name</th>
<th>Display Name</th>
<th>Type</th>
<th>Extended Context</th>
<th>Mandatory</th>
<th>Usage</th>
<th>Range</th>
<th>Wildcard</th>
<th>Predefined Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIENT</td>
<td>Client</td>
<td>MSO</td>
<td>X</td>
<td>X</td>
<td>CONFIG</td>
<td>X</td>
<td>X</td>
<td>TOTAL</td>
</tr>
<tr>
<td>USER</td>
<td>User</td>
<td>MSO</td>
<td>X</td>
<td>X</td>
<td>CONFIG</td>
<td>X</td>
<td>X</td>
<td>TOTAL</td>
</tr>
<tr>
<td>DATE</td>
<td>Date</td>
<td>MSO</td>
<td>X</td>
<td>X</td>
<td>CONFIG</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parameter field description

<table>
<thead>
<tr>
<th>Parameter field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technical Name</td>
<td>Technical parameter name</td>
</tr>
<tr>
<td>Display Name</td>
<td>Name of the parameter used to show up in the UIs</td>
</tr>
<tr>
<td>Type</td>
<td>There are two parameter types available for RFC (PULL)</td>
</tr>
<tr>
<td></td>
<td>- <strong>MSO</strong> (metric selection option): This is the standard parameter type to be used for ABAP data collectors</td>
</tr>
<tr>
<td></td>
<td>- <strong>CCID</strong> (collector context ID): This is a special parameter type to handle some product specific use cases. Not to be used in custom data providers.</td>
</tr>
<tr>
<td>Extended Context</td>
<td>When it is flagged this parameter will be used as part of the metric path.</td>
</tr>
<tr>
<td></td>
<td>In this example we will enter &quot;**&quot; for all clients in the system. On the system monitoring, there will be a metric path to each client. We will see how this is displayed in section 2.8.</td>
</tr>
<tr>
<td>Mandatory</td>
<td>When this is flagged the parameter is a mandatory input parameter. The monitoring template maintenance UI will complain in case the parameter is not filled with a value.</td>
</tr>
</tbody>
</table>
There are three usage types available:

- **HIDDEN**: The parameter can only be maintained in this data collector template maintenance UI. It is not possible to maintain it later in the monitoring template maintenance UI.
- **CONFIG**: The parameter can be maintained in the monitoring template maintenance UI and in case a predefined value is set it will be taken as default.
- **PRECONFIG**: For the Pull RFC metric there is no difference to the CONFIG parameter. Preferably use the CONFIG parameter.

<table>
<thead>
<tr>
<th>Ranges</th>
<th>The range flag allows you to set ranges as input values.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wildcards</td>
<td>The wildcard flag allows the usage of wildcards in the input values. It requires that also the Ranges flag is set.</td>
</tr>
</tbody>
</table>

In case you have a return parameter which should be used in the metric path you have to define it in the data collector template. Set the Type to MSO, set the Extended Context flag and set the Usage to HIDDEN.
2.3.3 Assign Help Values

Parameter DATE has two possible values: TODAY and YESTERDAY. Enter them accordingly.

In case you maintain help values you can select in the Monitoring Template Maintenance only from these pre-defined values.

2.3.4 Save the data provider

At the popup, hit close.
2.4 Create the Data Provider FM (Managed System)

In this step you create an FM in the managed system to execute the data collection. Any ABAP data provider must use the following import and export parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import SELOPT_PARA</td>
<td>/SDF/E2E_SELECTION_PARA_T</td>
<td>Contains the list of parameters defined in the data collector configuration. Supports wildcards, ranges, dynamic values.</td>
</tr>
<tr>
<td>Export RESULT</td>
<td>/SDF/E2E_RESULT_TT</td>
<td>Contains the result of an atomic call (count, min, max, text)</td>
</tr>
<tr>
<td>RETURN_STATUS</td>
<td>/SDF/E2E_EFWKE_RETURN_STATUS_T</td>
<td>May contain information about the data collection execution</td>
</tr>
</tbody>
</table>

Create a new function group (FG) and then create a FM ZCUSTOM_DUMP_INFO with the required parameters in this FG. To make it easier, you can reference the FM: /SDF/E2E_DUMP_INFO to copy the import and export parameters.

```abap
FUNCTION zcustom_dump_info.

  LOCAL Interface.
  IMPORTING
    SELOPT_PARA TYPE /SDF/E2E_SELECTION_PARA_T
  EXPORTING
    RESULT TYPE /SDF/E2E_RESULT_TT
    RETURN_STATUS TYPE /SDF/E2E_EFWKE_RETURN_STATUS_T
```

In the attached file there is an example ABAP code for this FM. The same source code is also included in the appendix of this guide. The example code handles simple cases. It is for training purpose only.
The structure of the import and export parameters can be seen in data dictionary (SE11).
2.4.1 Stackable Calls

One particular field CALL_ID in both import and export structures is worth mentioning.

CALL_ID is assigned by the EFWK Extension at runtime in the managed ABAP system. The purpose is to identify each data collection call. To increase the performance of the data collection, the EFWK Extension provides a feature, which is known as Stackable Algorithm. The following slide gives an example of stackable CALL IDs:

Without the stackable algorithm, the EFWK extension calls the Data Provider 1 using Input 1, Data Provider 2 using Input 2 and again Data Provider 1 using Input 3 → 3 select statements from the Data Providers, one per call

With stackable algorithm, the EFWK extension condense the Data Provider calls, only one call per Data Provider. Data Provider 1 using Input 1 and Input 3, Data Provider 2 using Input 2 → 2 select statements form the Data Providers, one per call

To identify which input parameter matches with which metric, the EFWK extension provides a call ID for the different input parameter. The Data Providers must map different results to the corresponding call IDs.

2.4.2 Test Function Module

To better understand the import and export parameters of the FM, you can copy the example source code into your FM ZCUSTOM_DUMP_INFO and test it in SE37. The following is an example import parameter set that you may use. You can enter multiple CALL_IDs or add more entries in T_RANGES.

<table>
<thead>
<tr>
<th>CALL_ID</th>
<th>SELECTION_PARAMETER</th>
<th>T_RANGES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PARAM</td>
<td>SIGN</td>
</tr>
<tr>
<td>1</td>
<td>CLIENT</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>USER</td>
<td>I</td>
</tr>
</tbody>
</table>
Sample Result of the FM test run:

<table>
<thead>
<tr>
<th>CALL_ID</th>
<th>T_PARAMETER</th>
<th>AVERAGE</th>
<th>COUNT</th>
<th>SUM</th>
<th>MIN</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CLIENT</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>USER TOTAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATE 20130612</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CLIENT 001</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>USER TOTAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATE 20130612</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If we expand the RESULT table, it is like the following. In this example, there are some ABAP dumps with client value = space in the table SNAP.

This DP has one measurement, therefore COUNT = 1, AVERAGE = SUM = MIN = MAX = the measured value. In more general cases, the field COUNT/MIN/MAX/SUM will have their own values and AVERAGE = SUM / COUNT. For example, if the DP is to measure the performance of a certain transaction in the past one hour. The COUNT will be number of execution of this transaction. MAX will be max execution time etc.
2.4.3 Source Code Part I: Extract parameter values

In the next few sections the source code of the FM will be explained in more detailed.

The input values of CLIENT, USER and DATE to the FM are entered when users create a metric in the Solution Manager setup. In section 2.6 we will see how to configure the Solution Manager. For now we just use the following screen shot to see the configured parameter values in Solution Manager.

At runtime these input values and CALL_ID will be packed into the import parameter SELOPT_PARA. So, the first step of the FM is to extract the parameter values and put them into internal variables.

<table>
<thead>
<tr>
<th>CALL_ID</th>
<th>SELECTION_PARAMETER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PARAM</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CLIENT</td>
</tr>
<tr>
<td></td>
<td>USER</td>
</tr>
<tr>
<td></td>
<td>DATE</td>
</tr>
</tbody>
</table>

Since the CLIENT or USER can be of value TOTAL to sum up the values of all clients or users, there are flag lv_total_client and lv_total_user to indicate the case of TOTAL. The select option is set to empty in this case. In the ABAP SQL statement if a select option is empty, all possible values will be selected.
LOOP AT selopt_para ASSIGNING <ls_selopt_para>. "Loop over all CALL_ID

* Retrieve parameter values from the metric configuration
LOOP AT <ls_selopt_para>-selection_parameter ASSIGNING <ls_selection_parameter>.

CASE <ls_selection_parameter>-param.
WHEN 'CLIENT'.
   CLEAR lv_total_client.
   lt_sel_opt_client = <ls_selection_parameter>-t_ranges.
   READ TABLE lt_sel_opt_client ASSIGNING <ls_sel_opt> INDEX 1.
   IF sy-subrc EQ 0 AND <ls_sel_opt>-sign = 'I'
      AND <ls_sel_opt>-option = 'EQ' AND <ls_sel_opt>-low = c_total.
      lv_total_client = 'X'.
      CLEAR lt_sel_opt_client. "Select all clients
   ENDIF.

WHEN 'USER'.
   CLEAR lv_total_user.
   lt_sel_opt_user = <ls_selection_parameter>-t_ranges.
   READ TABLE lt_sel_opt_user ASSIGNING <ls_sel_opt> INDEX 1.
   IF sy-subrc EQ 0 AND <ls_sel_opt>-sign = 'I'
      AND <ls_sel_opt>-option = 'EQ' AND <ls_sel_opt>-low = c_total.
      lv_total_user = 'X'.
      CLEAR lt_sel_opt_user. "Select all users
   ENDIF.

WHEN 'DATE'. "Required field, can only be TODAY or YESTERDAY
   CLEAR lv_date.
   lt_sel_opt_date = <ls_selection_parameter>-t_ranges.
   READ TABLE lt_sel_opt_date ASSIGNING <ls_sel_opt> INDEX 1.
   IF sy-subrc EQ 0.
      IF <ls_sel_opt>-low EQ c_today.
         lv_date = sy-datum.
      ELSEIF <ls_sel_opt>-low EQ c_yesterday.
         lv_date = sy-datum - 1.
      ENDIF.
   ENDIF.
   IF lv_date IS INITIAL.
   * Set error to export parameter RETURN_STATUS
      ls_return_status-call_id = <ls_selopt_para>-call_id.
      ls_return_status-status = 10.
      ls_return_status-msgtext =
         'Incorrect date value: enter TODAY or YESTERDAY'. "#EC NOTEXT
      APPEND ls_return_status TO return_status.
      EXIT.
   ENDIF.

   WHEN OTHERS.
   END_CASE.

ENDLOOP.
2.4.4 Source Code Part II: Select Data

ABAP dump information is stored in table SNAP. To get the number of dumps, we can just select seqno = ‘000’. The SQL statement below is to get the number of ABAP dumps based on the entered parameter values.

```sql
* Select data from table SNAP to read number of ABAP Dumps
SELECT * FROM snap INTO TABLE lt_snap
WHERE mandt IN lt_sel_opt_client
   AND uname IN lt_sel_opt_user
   AND datum EQ lv_date
   AND seqno = '000'.
```
2.4.5 Source Code Part III: Process Data and Produce Result

The selected data is further consolidated into internal table lt_dump_stat based on the selection parameters. In the case that CLIENT or USER is of value TOTAL, we set the individual fields in lt_snap to TOTAL before consolidation.

In the end, the outcome is mapped into the export parameter RESULT. It is important that the result includes the input parameters and the result values are grouped by the CALL_ID.

```plaintext
* Collect result data for this CALL ID
CLEAR: lt_dump_stat.
LOOP AT lt_snap ASSIGNING <ls_snap>.
  IF lv_total_client = 'X'.
    ls_dump_stat-client = c_total.
  ELSE.
    ls_dump_stat-client = <ls_snap>-mandt.
  ENDIF.
  IF lv_total_user = 'X'.
    ls_dump_stat-uname = c_total.
  ELSE.
    ls_dump_stat-uname = <ls_snap>-uname.
  ENDIF.
  ls_dump_stat-datum = <ls_snap>-datum.
  ls_dump_stat-entry = 1.
  COLLECT ls_dump_stat INTO lt_dump_stat.
ENDLOOP.

* Build result table
CLEAR ls_result.
ls_result-call_id = <ls_selopt_para>-call_id.
LOOP AT lt_dump_stat INTO ls_dump_stat.
  CLEAR: ls_parameter, ls_result-result.
  ls_parameter-param = 'CLIENT'.
  ls_parameter-value = ls_dump_stat-client.
  APPEND ls_parameter TO ls_result-result-t_parameter.
  ls_parameter-param = 'USER'.
  ls_parameter-value = ls_dump_stat-uname.
  APPEND ls_parameter TO ls_result-result-t_parameter.
  ls_parameter-param = 'DATE'.
  ls_parameter-value = ls_dump_stat-datum.
  APPEND ls_parameter TO ls_result-result-t_parameter.
  ls_result-result-count = 1.
  ls_result-result-average = ls_result-result-sum =
  ls_result-result-min = ls_result-result-max = ls_dump_stat-entry.
  APPEND ls_result TO result.
ENDLOOP.
```
2.5 Create Custom Template (Solution Manager)

In this step, you want to create a new template in which you create a new metric. If you already have your own customer template for the managed system, you can skip this step.

Execute transaction SOLMAN_SETUP in the Solution Manager.
Technical Monitoring ➔ Step 4: Template Maintenance

Switch to “Edit” mode, select standard template SAP ABAP xxx (xxx depends on the version of the component SAP_ABA in your managed system) and click “Create Custom Template”. On the “Template Settings” tab give the template a meaningful name. Save the template.
2.6 Create New Metric Group for Data Provider (Solution Manager)

Select the template you just created or your own custom template, switch to “Expert Mode”, click on “Create” and select Metric from the dropdown.

Enter the values in tab “Overview”. In this example, we want to know the number of ABAP Dumps in each client, therefore we create a metric group (in the field “Class”) instead of a metric.
Select “RFC (pull)” in the field “Data Collector”. Since the new DP is already defined in section 2.3, it will appear in the dropdown of “Data Provider” for you to select.

In this example, the client is entered as “*”. We expect that the system will collect ABAP dumps from each client and show their numbers. Since “CLIENT” is also defined as “Extended Context”, there will be a metric path to each client.
At step 2 “Assignment” you can assign this metric to an unassigned alert. For now we are only interested in the data collection, therefore we just click on “Finish” to save the metric group.

Remember to save your configuration and assign a package to it.
2.7 Assign Template to the Managed System and Activate it

Go to step 5 “Define Scope” to select the managed system.

Proceed to step 6 to assign custom template to your managed system. “Apply and Activate” after the assignment.

After activation, the status will become green. You may also click on “Directory Browser” to view the metric group you just configured.
### Development Guide for Custom RFC Pull Data Provider

**June 2013**

### Database Management

- **Database:** SQL
- **Host:** system账号@Z
- **Technical System:** SQL - ABAP

#### 1. Overview

- **Help:**
  - In this step, you will be assigned to the managed systems in scope. Each managed system consists of several managed objects of type technical system, databases, and host. Managed objects can have a template per installed product.

#### 2. Configuration

- **Default configuration:**
  - Check the ‘Generate per-SAP System’ checkbox and select the SAP System to be included in the template.
  - Default template configuration: All SAP Gateway Core 2.0, SAP Gateway Core 2.1, SAP Gateway Core 2.2, and SAP Gateway Core 2.3.模板名称为'125991'.

#### 3. Standard System

- **Managed Objects:**
  - **Assign Templates:**
    - **SAP:**
      - SAP ABAP
    - **Technical System:**
      - Database Manager

#### 4. Technical System Monitoring

- **Alerting Directory Browser**
  - **Managed Object Name:** ABAP
  - **Display MFA Change Status:**
  - **Available Variants for Managed Object:**

#### 5. Metrics

- **Select Options:**
  - **Monitoring Use Case:**
    - **Entry Point:**
      - **Apply**

- **Managed Objects:**
  - **Early Warning Alerts Integration**
  - **Infrastructure Monitoring**
  - **Technical System Monitoring**
    - **Managed Object:** ABAP

#### 6. System Monitoring

- **Exceptions:**
  - Metric group: SCHEMA_ID='SCHEMA_NAME='
    - Load Job Status
    - Master Job Status
    - Trigger Status

- **Performance:**
  - Average dialog response time (last hour)
  - Average payload size sent per service call
  - Average Response Time of Gateway Services last 5 minutes
  - Average RFC Response Time (last hour)
  - Number of concurrent named users
2.8 View Collected Metric

Technical Monitoring workcenter ➔ System Monitoring ➔ Select your managed system:

On this screen, click on the system name.

A new window for system monitoring will come up. On the right hand side, expand the node “System Exceptions” to see the metric group “ZZ Simple ABAP Dump”:

- ABAP Client = (space) ➔ value 4
- ABAP Client = 001 ➔ value 3
On this screen, you can also navigate to the BI report of a metric by clicking on the icon.
2.9 Create Alert for This Metric

This step is only required if you want to create an alert based on the custom metric.

2.9.1 Create Alert

Transaction SOLMAN_SETUP → Technical Monitoring → Step 4 Template Maintenance → Select Your Template and switch to Edit Mode, switch to Expert Mode → Create Alert

Enter alert name and category

Select “Worstcase Rule” for this alert
Custom Alert Creation Wizard

Specify Alert Attributes

Overview  Incidents  Notifications  Third-Party Components  Auto Reactions  Rule  Others

Rule Type: * [ ] Self-Implemented  [ ] WorstCase Rule
Enter alert technical name

In the step 2 “Assignments”, the system will list all metric or metric groups which have no alert assigned. Select the metric group you created and click on “Finish”.

After finishing the alert creation, you can switch to the tab “Metric, Events, Alert Hierarchy” to view your alert hierarchy.

Save the template
Go to step 5 Scope to select your managed system and then step 6 to activate the template.

2.9.2 View Alert

Technical Monitoring Workcenter ➔ System Monitoring ➔ Select your managed system ➔ System Monitoring

On the right hand side, expand the node System Exceptions to see the alert. The metric group was set to threshold type “Green/Yellow/Red” with threshold values 1 (to Yellow) and 10 (to Red). The alert rule is “Worstcase Rule”. You may see yellow or red alert in your system depending on the measured values.
3 Appendix: Sample Code for DP Function Module

In this appendix the source code for the data provider is listed. Note that this is a simple example. It only handles simple cases and is intended for training purpose only.

FUNCTION zcustom_dump_info.
*------------------------------------------------------------------
* Local Interface:
* # IMPORTING
* # REFERENCE(SELOPT_PARA) TYPE /SDF/E2E_SELECTION_PARA_T
* # EXPORTING
* # REFERENCE(RETURN_STATUS) TYPE /SDF/E2E_EFWKE_RETURN_STATUS_T
*------------------------------------------------------------------
* User selection parameters, as defined in Data Collector Template
* CLIENT (select option, required, default to 'TOTAL')
* USER (select option, required, default = 'TOTAL')
* DATE (parameter, required field, possible values = YESTERDAY or TODAY)

CONSTANTS: c_total TYPE string VALUE 'TOTAL', c_today TYPE string VALUE 'TODAY', c_yesterday TYPE string VALUE 'YESTERDAY'.

TYPES: BEGIN OF lty_dump_stat,
  client TYPE symandt,
  uname TYPE syuname,
  datum TYPE datum,
  entry TYPE i,
END OF lty_dump_stat.

DATA:
  lt_selopt_para LIKE selopt_para,
  ls_parameter TYPE /sdf/e2e_para,
  ls_result TYPE /sdf/e2e_result,
  lt_sel_opt_client TYPE /sdf/e2e_selop_tt,
  lt_sel_opt_user TYPE /sdf/e2e_selop_tt,
  lt_sel_opt_date TYPE /sdf/e2e_selop_tt,
  ls_return_status LIKE LINE OF return_status,
  ls_dump_stat TYPE lty_dump_stat, lt_dump_stat TYPE TABLE OF lty_dump_stat,
  ls_snap TYPE TABLE OF snap, ls_snap TYPE snap,
  lv_total_user, lv_total_client,
  lv_date TYPE d, lv_date_string TYPE string,
  lv_where TYPE string.

FIELD-SYMBOLS:
  <ls_selopt_para> TYPE /sdf/e2e_selection_para,
  <ls_selection_parameter> TYPE /sdf/e2e_selection_params,
  <ls_sel_opt> TYPE /sdf/ranges,
  <ls_snap> LIKE LINE OF lt_snap.

  LOOP AT selopt_para ASSIGNING <ls_selopt_para>. "Loop over all CALL_ID
  * Retrieve parameter values from the metric configuration
  LOOP AT <ls_selopt_para>-selection_parameter ASSIGNING <ls_selection_parameter>. 
CASE <ls_selection_parameter>-param.
  WHEN 'CLIENT'.
    CLEAR lv_total_client.
    lt_sel_opt_client = <ls_selection_parameter>-t_ranges.
    READ TABLE lt_sel_opt_client ASSIGNING <ls_sel_opt> INDEX 1.
    IF sy-subrc EQ 0 AND <ls_sel_opt>-sign = 'I' AND <ls_sel_opt>-option = 'EQ' AND <ls_sel_opt>-low = c_total.
      lv_total_client = 'X'.
    CLEAR lt_sel_opt_client. "Select all clients
  ENDIF.

  WHEN 'USER'.
    CLEAR lv_total_user.
    lt_sel_opt_user = <ls_selection_parameter>-t_ranges.
    READ TABLE lt_sel_opt_user ASSIGNING <ls_sel_opt> INDEX 1.
    IF sy-subrc EQ 0 AND <ls_sel_opt>-sign = 'I' AND <ls_sel_opt>-option = 'EQ' AND <ls_sel_opt>-low = c_total.
      lv_total_user = 'X'.
    CLEAR lt_sel_opt_user. "Select all users
  ENDIF.

  WHEN 'DATE'. "Required field, can only be TODAY or YESTERDAY
    CLEAR lv_date.
    lt_sel_opt_date = <ls_selection_parameter>-t_ranges.
    READ TABLE lt_sel_opt_date ASSIGNING <ls_sel_opt> INDEX 1.
    IF sy-subrc EQ 0.
      IF <ls_sel_opt>-low EQ c_today.
        lv_date = sy-datum.
      ELSEIF <ls_sel_opt>-low EQ c_yesterday.
        lv_date = sy-datum - 1.
      ENDIF.
    ENDIF.
    IF lv_date IS INITIAL.
    * Set error to export parameter RETURN_STATUS
      ls_return_status-call_id = <ls_selopt_para>-call_id.
      ls_return_status-status = 10.
      ls_return_status-
      msgtext = 'Incorrect date value: enter TODAY or YESTERDAY'. "#EC NOTEXT
      APPEND ls_return_status TO return_status.
      EXIT.
    ENDIF.

  WHEN OTHERS.
  ENDCASE.

ENDLOOP.

* If parameter error ==> jump to next CALL ID
  CHECK lv_date IS NOT INITIAL.

* Select data from table SNAP to read number of ABAP Dumps
  SELECT * FROM snap INTO TABLE lt_snap WHERE mandt IN lt_sel_opt_client AND u
  name IN lt_sel_opt_user AND datum EQ lv_date AND seqno = '000'.

* Collect result data for this CALL ID
  CLEAR: lt_dump_stat.
  LOOP AT lt_snap ASSIGNING <ls_snap>.
    IF lv_total_client = 'X'.
      ls_dump_stat-client = c_total.
ELSE.
    ls_dump_stat-client = <ls_snap>-mandt.
ENDIF.

IF lv_total_user = 'X'.
    ls_dump_stat-uname = c_total.
ELSE.
    ls_dump_stat-uname = <ls_snap>-uname.
ENDIF.

ls_dump_stat-datum = <ls_snap>-datum.
ls_dump_stat-entry = 1.
COLLECT ls_dump_stat INTO lt_dump_stat.
ENDLOOP.

* Build result table
CLEAR ls_result.
ls_result-call_id = <ls_selopt_para>-call_id.
LOOP AT lt_dump_stat INTO ls_dump_stat.
    CLEAR: ls_parameter, ls_result-result.
    ls_parameter-param = 'CLIENT'.
    ls_parameter-value = ls_dump_stat-client.
    APPEND ls_parameter TO ls_result-result-t_parameter.
    ls_parameter-param = 'USER'.
    ls_parameter-value = ls_dump_stat-uname.
    APPEND ls_parameter TO ls_result-result-t_parameter.
    ls_parameter-param = 'DATE'.
    ls_parameter-value = ls_dump_stat-datum.
    APPEND ls_parameter TO ls_result-result-t_parameter.
    ls_result-result-count = 1.
    ls_result-result-average = ls_result-result-sum = ls_result-result-
    min   = ls_result-result-max = ls_dump_stat-entry.
    APPEND ls_result TO result.
ENDLOOP.

END LOOP OVER call_id

ENDDO.