BPMon Data Collectors for non-ABAP/non-SAP:

Application Monitoring

Version: 2.0

Last Change: 20.09.2017
# Table of Contents

1. Introduction .......................................................... 5
2. The monitoring object “Generic Table Entry Counter for remote DB” (NATABCNX) ... 6
   2.1 Purpose .......................................................... 6
   2.2 Technical Prerequisites ........................................ 6
   2.3 Limitations ...................................................... 6
   2.4 Performance Warning ......................................... 7
   2.5 Setup Procedure ................................................ 7
       2.5.1 Example use-case ........................................ 7
       2.5.2 Simulate data collector result in monitored system ... 8
       2.5.3 Setup Procedure in SAP Solution Manager .......... 9
   2.6 SQL Query Format ............................................... 13
   2.7 Result in Monitoring Session ................................ 14
       2.7.1 Alert Display in Solution Manager .................... 14
       2.7.2 Detail Display call into monitored system .......... 14
3. Custom-specific key figures for 3rd-party or legacy systems ........................................ 16
   3.1 Purpose .......................................................... 16
   3.2 Technical Prerequisites ........................................ 16
   3.3 Preparation ..................................................... 16
   3.4 Maintain Monitoring Object Repository .................... 19
   3.5 Maintain Monitoring Object Coding .......................... 22
4. The monitoring object “TriplePoint Commodity SL Trade Monitoring” (NASTPCT1) ... 26
   4.1 Purpose .......................................................... 26
   4.2 Technical Prerequisites ........................................ 26
   4.3 Limitations ..................................................... 26
   4.4 Setup Procedure in SAP Solution Manager .................. 27
   4.5 List of Key Figures for CSL Trade ........................... 29
       4.5.1 Key Figure "EOD: Failed End-of-Day Process records" ... 29
       4.5.2 Key Figure "Scheduling: Failed Delivery Receipts / Goods Issues" ... 30
       4.5.3 Key Figure "Scheduling: Failed Transfer Match records in CSL" ... 30
       4.5.4 Key Figure "Master Data: Failed Reference Data Request records" ... 30
       4.5.5 Key Figure "GTM Trades: Total number of Custom Events" ... 30
       4.5.6 Key Figure "GTM Trades: Failed Trade Modification records" ... 31
       4.5.7 Key Figure "GTM Trades: Failed Trade Reference Number records" ... 31
       4.5.8 Key Figure "GTM Trades: Failed Trade Release records (no SO/PO)" ... 31
       4.5.9 Key Figure "GTM Trades: Failed Trade Voiding records (no SAP update)" ... 31
       4.5.10 Key Figure "TRM Trades: Failed TRM Clearing Broker records" ... 32
       4.5.11 Key Figure "TRM Trades: Failed TRM Floor Broker records" ... 32
   4.6 Performance Warning ............................................ 32
   4.7 Result in Monitoring Session ................................. 33
       4.7.1 Alert Display in Solution Manager ..................... 33
       4.7.2 Detail Display call into monitored system .......... 33
5. Introduction to monitoring external systems with Web services .................................... 35
6. Monitoring object “Dependent Data Collector for Push WS” (NAWSPUSH) ....................... 36
   6.1 Purpose .......................................................... 36
   6.2 Technical Prerequisites ........................................ 36
   6.3 Performance Warning ........................................... 36
   6.4 Setup Procedure ................................................ 36
       6.4.1 Example use-case .......................................... 36
       6.4.2 Setup in the managed non-SAP system ............... 37
       6.4.3 Setup Procedure in SAP Solution Manager ............ 37
6.5 Result in Monitoring Session ......................................................... 41
6.5.1 Alert Display in Solution Manager ............................................. 41
6.5.2 Detail Display call into monitored system .................................. 41
7 Monitoring object “Data Collector for Pull WS” (NAWSPULL) ................. 42
7.1 Purpose ..................................................................................... 42
7.2 Technical Prerequisites .............................................................. 42
7.3 Performance Warning ................................................................. 42
7.4 Setup Procedure ........................................................................ 42
  7.4.1 Example use-case .................................................................... 42
  7.4.2 Setup in the managed system .................................................. 42
  7.4.3 Setup Procedure in SAP Solution Manager .............................. 43
7.5 Result in Monitoring Session ....................................................... 50
  7.5.1 Alert Display in Solution Manager ........................................... 50
  7.5.2 Detail Display call into monitored system .................................. 50
8 Detail Display for “Dependent Data Collector for Push WS” (NAWSPUSH) and “Data Collector for Pull WS” (NAWSPULL) ......................................................... 50
  8.1 Purpose ..................................................................................... 50
  8.2 Technical Prerequisites .............................................................. 50
  8.3 Performance Warning ................................................................. 50
  8.4 Setup Procedure ........................................................................ 51
    8.4.1 Example use-case .................................................................... 51
    8.4.2 Setup in the managed system .................................................. 51
    8.4.3 Setup Procedure in SAP Solution Manager .............................. 51
  8.5 Result in Monitoring Session ....................................................... 53
    8.5.1 Detail Display for a Web Page .................................................. 53
    8.5.2 Detail Display call to receive the data ...................................... 53
9 The monitoring object “Generic Infocube/DSO Reader for remote BW” (NABWICNX) ......................................................... 55
  9.1 Purpose ..................................................................................... 55
  9.2 Technical Prerequisites .............................................................. 55
  9.3 Limitations ................................................................................ 55
  9.4 Performance Warning ................................................................. 56
  9.5 Setup Procedure ........................................................................ 56
    9.5.1 Example use-case .................................................................... 56
    9.5.2 Simulate data collector result in monitored system ................. 56
    9.5.3 Setup Procedure in SAP Solution Manager .............................. 58
  9.6 Result in Monitoring Session ....................................................... 63
    9.6.1 Alert Display in Solution Manager ........................................... 63
    9.6.2 Detail Display call into monitored system .................................. 63
10 Appendix I: Technical Details ........................................................... 64
  10.1 Remote Database Connectivity .................................................. 64
  10.2 Derive ADBC Connection Entry ............................................... 68
    10.2.1 Derive ADBC Connection Entry from System Landscape (SMSY) – Lower SolMan 7.1 SP05 ................................................................. 68
    10.2.2 Derive ADBC Connection Entry from Landscape Management Database (LMDB) – SolMan 7.1 SP05 or higher ............................. 71
  10.3 Dynamic Date Filtering for remote database monitoring .................. 74
  10.4 WSDL-Document ..................................................................... 76
  10.5 Derive BW Connection Entry ..................................................... 80
    10.5.1 Derive BW Connection Entry from System Landscape (SMSY) .................. 80
10.5.2  Derive BW Connection Entry from Landscape Management Database (LMDB)
82
10.6  Dynamic Date Filtering for BW-queries................................................................. 84
11  Appendix II: Test Environment for Remote DB Query (program /SSA/ENA) ........... 85
11.1  Purpose.................................................................................................................... 85
11.2  Technical Prerequisites .......................................................................................... 85
11.3  Limitations ............................................................................................................ 86
11.4  Performance Warning ............................................................................................ 86
11.5  Main Selection Screen .......................................................................................... 86
11.5.1  Processing Mode ................................................................................................. 87
11.5.2  Remote Database Table ........................................................................................ 87
11.5.3  Filter Criteria for Where-Clause .......................................................................... 88
11.5.4  Display Parameters ............................................................................................ 89
11.6  Processing Mode “Count number of entries” ......................................................... 90
11.7  Processing Mode “Count number of distinct values” .............................................. 91
11.8  Processing Mode “Get single field value” ............................................................. 91
11.9  Processing Mode “Display Table Content” ............................................................ 92
11.10  Processing Mode “Compare Table with second connection” ............................... 93
11.11  Processing Mode “Free Select Query” .................................................................. 95
11.12  Further Diagnostics Features .............................................................................. 97
11.12.1  Connection Test ............................................................................................... 97
11.12.2  Display Work Process Trace ............................................................................. 99
11.13  Table of figures .................................................................................................... 102
1 Introduction
This document describes the general framework used to connect non-ABAP and non-SAP systems to Business Process Monitoring.

In order to monitor non-ABAP or non-SAP systems you can use

- **Application Monitoring on Remote Databases**
  - with generic monitoring object
    - Generic Table Counter (monitor NATABCNX) described in section
    - “The monitoring object “Generic Table Entry Counter for remote DB” (NATABCNX)”
  - with specific monitoring object
    - Custom-specific key figures for other 3rd-party or legacy systems

2 Sample key figures for TriplePoint Commodity SL (monitor NASTPCT1) described in section The monitoring object “Generic Table Entry Counter for remote DB” (NATABCNX)

The "Generic Table Entry Counter for remote DB” provides the possibility to run queries to count the number of entries (or the number of distinct values) in any remote database table. It supports a flexible filtering based on select-fields of a table. Using ADBC (ABAP Database Connectivity) it can access a remote database table on an SAP-supported DBMS (database management system), using NativeSQL statements.

This monitoring object is a substitute or workaround for areas, where no specific application monitor is available. Furthermore this monitor can help to avoid implementing the application monitoring customer-exit, as long as the collector result can be calculated by a simple SELECT statement on one single table or view.

This application monitor is part of the Business Process Monitoring framework within SAP Solution Manager.

**Technical Prerequisites**

The Generic Table Entry Counter is delivered with the add-on ST-API starting with release 01M. The technical name of the Monitoring Object is NATABCNX.

All SAP Basis (NetWeaver) releases starting from release 7.00 and later are supported. It is recommended to run the data collector directly on SAP Solution Manager for this purpose.
To monitor an external (non-ABAP) database using ADBC, the corresponding secondary database connection must be maintained with transaction DBACOCKPIT. Please see appendix “Remote Database Connectivity” for more details.

Inside the Solution Manager, you must have a solution created with at least one business process maintained. The business process should have several business process steps. The system assigned to this step can be any system, but the ABAP based data collector is executed physically on any SAP NetWeaver ABAP system, preferably the Solution Manager itself. So the actual managed system is not the system where the data collector runs, but the system to which the remote database connection refers to. So you may want to create an own Logical Component for this non-ABAP/non-SAP system (instead of using the Solution Manager’s Logical Component), in order to have a unique assignment of the generated alert records.

This tool must not be used to query databases of SAP ABAP-based systems. For that purpose you can use the “Generic Table Entry Counter” (technical name BOTABCNT), build especially to support database tables defined inside the SAP ABAP Data Dictionary. It used the preferred way of OpenSQL instead of NativeSQL.

**Limitations**

Because the data collector works on database tables, which are not defined inside the SAP ABAP Data Dictionary, there is no automatic support of certain build-in functionality, such as

- multi-lingual text labels for table columns,
- complex search helps on check tables or domain fix values,
- input and output conversion according to certain data types or user’s locale settings.

Search helps for defining filters are fed by selecting distinct field values from the corresponding table columns, which might be a performance-intensive query. Within the BPMon Setup Session, all filter criteria are stored as character fields.

The data collector accesses the remote DB data dictionary to query the table meta data (like column’s data types) in order to map the native data types into SAP dictionary data types and ABAP data types, for example to allow a detail display with the list viewer. However, it cannot be guaranteed that all native DBMS-dependent data types can be properly matched into a suitable ABAP data type.
Therefore the output of certain data might look different compared to the user interface of native database tools or especially compared to the respective non-ABAP application, which may perform its own output conversion. There might be restrictions especially for number types (like maximum amount of decimals) or date types (like other than expected formats for dates and timestamps).

2 **Performance Warning**

2 The data collector can query any table **that is accessible by the login user. Therefore it is in the responsibility of the user to avoid frequent selections on large tables. Keep in mind that the necessary SQL selection aggregates for COUNT and DISTINCT create expensive database selects, including full table scans!**

2 Using this application monitor without care can cause severe performance degradation on the managed system.

2 Please consider your setup carefully and follow these golden rules:

2 Avoid using this application monitor on large database tables

2 Avoid running the data collection with a high monitoring frequency

2 Avoid complex filtering, especially with patterns, ranges and exclusions

2 Avoid filters on fields which are not supported by an index.

2 If you are in doubt, try to simulate the query with the **Test Environment for Remote DB Query**, to get an idea about the possible runtime. Furthermore, monitor the data collection runtime which is logged into a special column of the alert list (see chapter “Alert Display”).
Setup Procedure

Example use-case

To demonstrate the setup of the Table Entry Counter we choose a simple demo table APP_SALES_ORDER_ITEM of a fictive non-ABAP sales application with the following column structure:

<table>
<thead>
<tr>
<th>S...</th>
<th>Name</th>
<th>Datatype</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ORDER_NUMBER</td>
<td>INTEGER</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>ITEM_NUMBER</td>
<td>INTEGER</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>QUANTITY</td>
<td>INTEGER</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>MATERIAL</td>
<td>INTEGER</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>ITEM_STATUS</td>
<td>CHARACTER</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>REQ_DLV_DATE</td>
<td>CHARACTER</td>
<td>8</td>
</tr>
</tbody>
</table>

Picture 1: Example data model

It contains some sample sales order items:

<table>
<thead>
<tr>
<th>ORDER_NUMBER</th>
<th>ITEM_NUMBER</th>
<th>QUANTITY</th>
<th>MATERIAL</th>
<th>ITEM_STATUS</th>
<th>REQ_DLV_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>10</td>
<td>1</td>
<td>3</td>
<td>created</td>
<td>20000630</td>
</tr>
<tr>
<td>1000</td>
<td>20</td>
<td>4</td>
<td>5</td>
<td>created</td>
<td>20000630</td>
</tr>
<tr>
<td>1001</td>
<td>10</td>
<td>50</td>
<td>1</td>
<td>picked</td>
<td>20000630</td>
</tr>
<tr>
<td>1001</td>
<td>20</td>
<td>20</td>
<td>8</td>
<td>picked</td>
<td>20000630</td>
</tr>
<tr>
<td>1001</td>
<td>30</td>
<td>2</td>
<td>9</td>
<td>open</td>
<td>20000630</td>
</tr>
<tr>
<td>1002</td>
<td>10</td>
<td>17</td>
<td>6</td>
<td>shipped</td>
<td>20001231</td>
</tr>
<tr>
<td>1003</td>
<td>10</td>
<td>12</td>
<td>2</td>
<td>waiting</td>
<td>20001231</td>
</tr>
<tr>
<td>1004</td>
<td>10</td>
<td>32</td>
<td>1</td>
<td>shipped</td>
<td>20000301</td>
</tr>
<tr>
<td>1005</td>
<td>10</td>
<td>1</td>
<td>7</td>
<td>open</td>
<td>20000701</td>
</tr>
<tr>
<td>1006</td>
<td>10</td>
<td>34</td>
<td>4</td>
<td>open</td>
<td>20000701</td>
</tr>
<tr>
<td>1007</td>
<td>10</td>
<td>66</td>
<td>7</td>
<td>open</td>
<td>20000701</td>
</tr>
<tr>
<td>1008</td>
<td>10</td>
<td>1</td>
<td>2</td>
<td>delivered</td>
<td>20000201</td>
</tr>
<tr>
<td>1008</td>
<td>20</td>
<td>1</td>
<td>4</td>
<td>delivered</td>
<td>20000201</td>
</tr>
<tr>
<td>1009</td>
<td>30</td>
<td>3</td>
<td>7</td>
<td>packed</td>
<td>20000201</td>
</tr>
<tr>
<td>1009</td>
<td>10</td>
<td>65</td>
<td>8</td>
<td>shipped</td>
<td>20000201</td>
</tr>
</tbody>
</table>

Picture 2: Example data

Simulate data collector result in monitored system

You can use the “Test Environment for Remote DB Query” (running program /SSA/ENA) to simulate the database query and to clarify the selection parameters. Assuming this remote database can be accessed with a DBCON entry called ZTN_DB6_2 and the table APP_SALES_ORDER_ITEM belongs to schema APP.
Now as we have proved that the table content can be accessed from the ABAP stack, we want to setup the following KPIs for an application monitoring:

- Number of unprocessed sales items
- Sales orders with unprocessed items
- Lowest sales order number with unprocessed item(s)

**Setup Procedure in SAP Solution Manager**

In short: For the application monitoring, choose the monitoring object NATAHCNX and specify the following selection criteria:

- Database connection name and schema name
- Table name
2 Field name for filtering and select-**options for actual filter values**
2 this is available five **times and will be used to generate a WHERE clause**

2 Detailed instructions:

2 Call the SAP Solution Manager (trx DSWP or SOLMAN_WORKCENTER depending on the Solution Manager’s release).

2 Choose your solution.

2 Go to 'Operation Setup' and navigate to 'Solution Monitoring' => 'Business Process Monitoring'.
Alternatively, you can start the Setup Session from the work center “Business Process Operations” from the link “Setup Business Process Monitoring” in the “Common Tasks” area.

2 One-time action only:
Check 'Basic Settings' => 'Update Central Application Monitoring Repository': Press Button ‘Load Monitors’ and select the System ID where the ABAP-part of the data collector shall run (which preferably is the System ID of Solution Manager itself).

2 Check 'Business Processes': select a business process for application monitoring.

2 Check for your chosen business process: choose process steps to monitor.

2 Check for your chosen step: Choose type of monitor, here 'Application Monitors: Other area'.

2 Choose application monitor ‘**Generic Table Entry Counter for remote DB**’ [NATABCNX] via search help and name your monitoring object with a speaking name. This will create a sub-tree below 'Application Monitors' with the name of your
The first two parameters "Database Connection" and "Database Schema" are to define the target database server (using a DBCON entry name) and database schema for the remote database connection. The preferred way to maintain these attributes is inside the system landscape definition (transaction SMSY/LMDB). The business process step refers to a logical component, which itself points to a system that uses a database. At the maintenance screen of this database you should enter the DBCON connection name and schema name into the free attributes called "Remote DB Conn. for Appl.Mon." and "DB Schema for Application Data". This provides a central place to store this technical information, which is then valid for all application monitors using a remote database connection to this database. In such a case it is not necessary to maintain the two monitoring object parameters "Database Connection" and "Database Schema" in the setup session. If you maintain the parameters in the session, this will override the SMSY/LMDB attributes and the monitoring will use your entered values instead.

For more information regarding the usage of a remote DB connection, please see appendixes "Remote Database Connectivity" and "Derive Connection Entry".

As the third parameter, enter the "Table Name". The search help loads the available table names from the remote database. For performance reasons, make sure to enter a pattern (like AP* for all tables prefixed by APP) to restrict the selection. The table name is the only mandatory parameter, whereas the following filters are optional.
The optional fourth parameter called "Display Columns" allows restricting the scope of the detail display screen. Instead of a SELECT * only the specified column names will be loaded from the remote database table, to allow hiding some technical fields. For each field name you can add a descriptive field label text that will be used as column header in the detail display. Just postfix a double-dash (SQL comment "--") and the text as shown in Picture 7.

Example:

![Display Columns](image)

2 **Picture 7: Defining display columns**

2 Otherwise you would see the technical field names only, as the remote database dictionary access does not provide multi-lingual text labels as known from the SAP ABAP dictionary.

As the optional fifth parameter "Filter 1 Fieldname", enter the field name for the first filter, which obviously must be a valid field of the selected database table. A search help dialog is also shown here, which shows all fields of the selected table, based on the remote database dictionary.

In the sixth line, enter the select criteria "Filter 1 Sel.options" for the first filter mentioned above, which represent the actual filter field values. When you have already entered the table name and the field name for filtering, the system tries to provide an entry help from the remote database table column content. However, this may be performance-intensive due to the selection method of distinct values, so please use with care. Please note to enter all values in the native database format, for example a date 2008/09/24 might be entered as 20080924, or timestamps are usually entered in a 24-hour format without separators. Please see the appendix on how to implement dynamic date filtering, using relative dates instead of absolute values.

Repeat these steps for further filters if required.

Example: The KPI “Number of unprocessed sales items” would be defined as all sales order items in table APP_SALES_ORDER_ITEM which have an ITEM_STATUS of “created”, “open” or “waiting”.

![Parameters](image)

2 **Picture 8: Example parameters**
When you are finished with the setup of the header parameters, press button “Validation Check” to validate the input and perform a connection test.

Check '<<your monitoring object>>' tab strip ‘Monitoring Schedule’: Enter the data for the scheduling of the collector job in the monitored **system, i.e. weekdays, collection period** or fixed start time.

Check '<<your monitoring object>>' tab strip 'Key Figures': choose the key figure(s) to focus on. There are three key figures available.

When marking them, after saving you see new sub-checks where you can do additional customizing on key figure level.

**“Number of Counted Entries” (with given filter criteria)**

**“Number of Distinct Values” (on single field name and with given filter criteria)**

Enter the parameter “Distinction Fieldname”, that is the name of the table field for which the different values are to be counted. Please make use of the value help, which only displays valid fields of the selected table.

**“Value of Single Field” (with or without aggregate and with given filter criteria)**

As additional customizing at key figure level, you have to enter the name of the single field for which you want to read its value (**"Selection Fieldname"**). Ideally the data type of this field is an integer. Numeric fields with decimals will be rounded. For other types the result may be undefined or you would even get an error message.
in the alert text regarding a conversion error. Make sure that the where-clause defined by the filter fields selects one single row only. Otherwise the result might be not unique, which will be indicated in the alert text.

2 In addition you can optionally enter an "Aggregate Function". If the select results in a single integer value is returned due to aggregation, for example MAX(ERRORCODE) would return the highest error code from the error table column. Please use the value help to select from possible aggregate functions:

2 Default: None
2 MIN: Minimum value
2 MAX: Maximum value
2 AVG: Average value
2 SUM: Sum of values

2 For each key figure, you can customize an "Alternative Alert Text", using certain placeholder variables.

2 The standard key figure alert texts return a rather technical alert message - “Number of entries in table <table> = <counter>” or - “Number of distinct values in field <field> of table <table> = <counter>” This is automatically proposed if you leave the parameter “Alternative Alerttext” empty.

2 You can use any free text together with the following placeholders to overwrite the standard alert text with your own alert text:
- $COUNT returns the measured value from the selection
- $TABLE returns the database table name
- $FIELD returns the table’s field name

2 Example for our use-case in key figure “Number of Counted Entries”:
- “$COUNT unprocessed sales items”

2 For each key figure, the parameter "Detailcall Headertext" allows to store a screen title for the detail display, to explain the user what kind of output is shown.

2 Picture 11: Detailcall headertext parameter
For each key figure you can enter a threshold for a YELLOW alert and a threshold for a RED alert as an absolute value, also supporting a four-step rating with minimum and maximum thresholds pairs (RED >= YLW >= Value >= YLW >= RED).

Picture 12: Thresholds for NATABCNX

At node 'Analysis & Monitoring Tools', do not select an ABAP-based analysis tool like a transaction code. Instead, later on inside the monitoring session you will have a button called “Detail Info” which will display the relevant table entries inside a list viewer according to the setup filters.

Once you have entered all relevant information for the monitoring objects you want to monitor, generate the monitoring customizing and activate the monitoring within the 'Business Process Monitoring' session.

**SQL Query Format**

The key figure "Number of Counted Entries" creates an SQL query in the following format:

```
SELECT COUNT(*)
FROM table
WHERE field1 IN filterrange1
  AND field2 IN filterrange2
  AND field3 IN ...
```
The key figure "Number of Distinct Values" creates an SQL query in the following format:

```sql
SELECT COUNT(DISTINCT field) 
FROM table 
WHERE field1 IN filterrange1 
AND field2 IN filterrange2 
AND field3 IN ...
```

The key figure "Value of Single Field" creates an SQL query in the following format:

```sql
SELECT field 
or
SELECT aggregate(field) 
FROM table 
WHERE field1 IN filterrange1 
AND field2 IN filterrange2 
AND field3 IN ...
```
2 **Result in Monitoring Session**

2 **Alert Display in Solution Manager**

![Image of alert display](image)

2 **Picture 13: Alert display for "Number of counted entries"**

2 The column "Alert Message" shows the first 120 characters of the alert text, which was in this case assembled from the customized alternative alert text template. In addition, there are some additional columns filled, like Database connection name, Table name, and Runtime (which shows how long it took for the data collector to query the key figure).

2 **Alert display for key figure “Number of Distinct Values”:**

2 The column "Alert Message" shows the first 120 characters of the alert text, which might be assembled from your customized alternative alert text template as well. Furthermore, there are some additional columns filled, like Database connection name, Table name, and Field name (just repeats which field was used for the distinction) and Runtime (which shows how long it took for the data collector to query the key figure).

2 **Alert display for key figure “Value of Single Field”:**

2 The column "Alert Message" shows the first 120 characters of the alert text, which might be assembled from your customized alternative alert text template as well. Furthermore, there are some additional columns filled, like Database connection name, Table name, and Field name (just repeats which field was used for the distinction) and Runtime (which shows how long it took for the data collector to query the key figure).
Detail Display call into monitored system

Mark an alert and press push button “Detail Info”. This loads the relevant remote database table entries into an SAP-ABAP-based list viewer. The display might slightly differ from what you see with native database tools or within the UI of the external application. Not all table columns might find a matching ABAP type or might be represented in some other display format.

Please note that you always see the current selection result, even if you have selected an older alert. Reason is that the selection result for the regular key figure calculation is not stored, but rather re-selected during the detail display call.

The detailed info for the number of counted entries just displays the table rows with the affected entries (applying the same filtering as the data collector). Please note that a maximum of 10,000 rows is shown. You can also see a header text as defined in the corresponding setup parameter.
The detailed info for the number of distinct values displays an overview with each distinct field value and its number of occurrences (applying the same filtering as the data collector). In our example it shows all sales order numbers and the number of affected open items. Obviously the sum of all occurrences should be equal to the value of key figure “Number of Counted Entries”.

**Detail information for key figure “Value of Single Field”**

The detailed info for the value of a single field looks identical to the one from the first key figure “Number of Counted Entries”. Instead of just showing one row with one field containing its single or aggregated value, the display shows all affected rows for a better traceability of the aggregated result.
2 Custom-specific key figures for 3rd-party or legacy systems

2 Purpose

2 The BPMon framework allows storing pre-defined SQL-Statements in its repository, which act as data selection method for monitoring key figures. As long as the whole data collection (that is calculating the alert measured value) can be described by a single Select-Statement, there is no need to create additional special ABAP code. You can just define the key figure with its Select-Statement, together with a small program entry point, which delegates all the complex connection and remote database handling to general framework subroutines.

2 The following procedure explains how to implement the BPMon customer-exit for additional Remote-DB Application Monitoring key figures, by using the framework of pre-defined Select-Statements.

2 Technical Prerequisites

2 Be familiar with the general concept of implementing the BPMon customer-exit.


2 Preparation

2 Prepare the required SQL-Select-Statements for

2 Count (calculating alert measured value; mandatory) => SELECT COUNT(*) FROM...

2 Display (detail display of selection result; optional) => SELECT <fieldlist> FROM...

2 You can test your queries with program /SSA/ENA in processing mode ‘Free Select Query’ (see chapter “Test Environment for Remote DB Query (program /SSA/ENA)”)

2
Example for Counter:

**Test Environment for Remote-DB Query**

**Processing Mode**
- Count number of entries
- Count number of distinct values
- Get single field value
- Display Table Content
- Compare Table with second connection
- Free Select Query

**Remote Database Table**
- Database Connection Name: ZTN_TPT_PWDF6667_ORA
- Database Schema Name: CSL71_BASELINE
- Table Name: 

**Edit SQL Command**

```
SELECT COUNT(*) FROM CSL71_BASELINE.CSL_MSG_MONITOR
WHERE CSL_MSG_MONITOR.SENDER = 'SAP'
```

**Database Connection**
- ZTN_TPT_PWDF6667_ORA (Triple Point CSL @ pwdf6667)
- Database System: Oracle
- Database Schema: CSL71_BASELINE
- Database Table: *
- Selection result: 1 entry
- Query runtime: 0.014 seconds
- Generated SQL Query: 

```
SELECT COUNT(*) FROM CSL71_BASELINE.CSL_MSG_MONITOR
WHERE CSL_MSG_MONITOR.SENDER = 'SAP'
```

**COUNT(*)**

2016

**Picture 16:** Key figure Count simulation in Test Environment
Example for Display:

Test Environment for Remote-DB Query

Processing Mode
- Count number of entries
- Count number of distinct values
- Get single field value
- Display Table Content
- Compare Table with second connection
- Free Select Query

Remote Database Table
- Database Connection Name: ZTN_TPT_PWDF5657_ORA
- Database Schema Name: CSL71_BASELINE
- Table Name: CSL_MSG_MONITOR

Edit SQL Command

```
SELECT CSL_MSG_MONITOR.CSL_MSG_MONITOR_NUM, CSL_MSG_MONITOR_SENDER, 
CSL_MSG_MONITOR.SENDER_NAMESPACE, CSL_MSG_MONITOR_LAST.MODIFY_DT, 
CSL_MSG_MONITOR_CLASS_NAME 
FROM CSL71_BASELINE.CSL_MSG_MONITOR 
WHERE CSL_MSG_MONITOR_SENDER = 'SAP'
```

Database Connection: ZTN_TPT_PWDF5657_ORA (Triple Point CSL @ pwdf5657)
- Database System: Oracle
- Database Schema: CSL71_BASELINE
- Database Table: CSL_MSG_MONITOR
- Selection result: 1000 entries
- Query runtime: 0.028 seconds
- Generated SQL Query:

```
SELECT CSL_MSG_MONITOR.CSL_MSG_MONITOR_NUM, CSL_MSG_MONITOR_SENDER, 
CSL_MSG_MONITOR.SENDER_NAMESPACE, CSL_MSG_MONITOR_LAST.MODIFY_DT, 
CSL_MSG_MONITOR_CLASS_NAME 
FROM CSL71_BASELINE.CSL_MSG_MONITOR 
WHERE CSL_MSG_MONITOR_SENDER = 'SAP'
```

<table>
<thead>
<tr>
<th>CSL_MSG_MONITOR</th>
<th>CSL_MSG_MONITOR_SENDER</th>
<th>CSL_MSG_MONITOR.SENDER_NAMESPACE</th>
<th>CSL_MSG_MONITOR_LAST.MODIFY_DT</th>
<th>CSL_MSG_MONITOR_CLASS_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.379</td>
<td>SAP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.382</td>
<td>SAP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.321</td>
<td>SAP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.807</td>
<td>SAP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.808</td>
<td>SAP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.815</td>
<td>SAP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.821</td>
<td>SAP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.831</td>
<td>SAP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.834</td>
<td>SAP</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Picture 17: Key figure Count simulation in Test Environment
2 Maintain Monitoring Object Repository

2 Run program /SSA/EXM for P_PROJID = ‘ECU’ in change mode.

2 Create the Monitoring Object. For ST-A/PI release 01M the name must start with an ‘S’ (like for the standard monitors in project ENA).

2 Picture 18: Definition of Monitoring Object

2 Go to “Customize Parameters”

2 Set Data Retrieval = ‘Remote DB Query’.

2 Picture 19: Definition of Monitoring Object Header Attributes

2 No Parameters on header level

2 Set Application = CUSTOMER
2 Go to “Define Key Figures” and create all required key figures (one per each different query)

<table>
<thead>
<tr>
<th>Keyfigure</th>
<th>Type</th>
<th>DCC relevant?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>81</td>
<td>KZ</td>
<td></td>
<td>Number of Messages sent by SAP</td>
</tr>
<tr>
<td>82</td>
<td>KZ</td>
<td></td>
<td>Number of Messages sent by TPT</td>
</tr>
</tbody>
</table>

2 Picture 20: Definition of Monitoring Object’s Key Figures

2 Mark Key Figure and go to “Customize Parameters”

2 On tab ‘Attributes’, set Additional Functionalities = ‘Detail Info’ if this needs to be supported.
   Validation Check and Default Value are not yet possible in customer-exit.
   Enter text labels for Threshold 1+2 (for 2-step rating) and optionally also Threshold 3+4 (for 4-step-rating).

2 Picture 21: Definition of Key Figure Attributes

2 On tab ‘Parameters’, you can define user-customizable parameters or select options, which will appear on the counter popup in BPMon Setup UI. You need to maintain the table name as Tech Name 1 and the field name as Tech Name 2 to get an
automatic value help based on distinct values in that table column.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
<th>Tech Name 1</th>
<th>Tech Name 2</th>
<th>Mandatory</th>
<th>Check</th>
<th>No dyn.</th>
<th>F4 Help</th>
<th>Selection</th>
<th>Option Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Operation Name</td>
<td><code>CSL_MSG_MONITOR</code></td>
<td><code>OPERATION_NAME</code></td>
<td><code>0</code></td>
<td><code>1</code></td>
<td><code>0</code></td>
<td><code>0</code></td>
<td><code>0</code></td>
<td><code>0</code></td>
</tr>
</tbody>
</table>

2 **Picture 22: Definition of Key Figure Parameter**

On tab 'ADBC Statement Data Collection' maintain the SQL statement fragments for SELECT-, FROM-, WHERE- and GROUP-BY-clause. The SELECT-clause must return an integer value, e.g. with using a COUNT-aggregate. The FROM-clause is mandatory and may contain the placeholder '/\*SCHEMA*/', which will be replaced with the remote DB schema name at runtime. The WHERE-clause is optional and should contain all hard-coded filters (in contrast to user-customizable filters as above). The GROUP-BY-clause is optional and should be used in conjunction with a DISTINCT-aggregate.

2 **Picture 23: Definition of SQL Statement for Counting**

On tab 'ADBC Statement Detail List' maintain the SQL statement fragments for SELECT-, FROM-, WHERE-, GROUP-BY- and ORDER-BY-clause. The SELECT-clause must contain the field list for the columns to be displayed, using a qualified column name ("TABLE".'FIELD'). The FROM-clause is mandatory and may contain the placeholder '/\*SCHEMA*/', which will be replaced with the remote DB
schema name at runtime. **The WHERE-clause is optional and should contain all hard-coded filters (in contrast to user-customizable filters as above).** Typically the FROM- and WHERE-clause are identical to those used in the Data Collection query. The GROUP-BY-clause and ORDER-BY-clause are optional.

![Image of SQL statement for detail display]

**Picture 24: Definition of SQL Statement for Detail Display**

1. Mark Key Figure and go to “Detail List”

2. For each key figure, you can define the text labels for the Short/Medium/Long column headings of the list viewer during detail display calls. The columns ‘DDIC Table’ and ‘DDIC Field’ must match with the identifiers used as field list in the SELECT-clause.

![Image of column list for detail display]

**Picture 25: Definition of Column List for Detail Display**

1. **Maintain Monitoring Object Coding**

2. Edit program Z_BPM_ECU_COLLECTOR.
Make sure that the following **Includes** are included:

```plaintext
INCLUDE /SSA/IET.
INCLUDE /SSA/IES.
INCLUDE /SSA/IEN.
```

**At event LOAD-OF-PROGRAM call form routine NABRDB_CONFIG_LOAD:**

```plaintext
LOAD-OF-PROGRAM.
PERFORM NABRDB_CONFIG_LOAD.
```

For each Monitoring Object, **maintain the entry form routines to be called from** BPMon framework.

**Value Help**

**Data Collector**

**Detail Info**

**Value Help** (VH):
There is no way to derive DBCON+SCHEMA from SMSY/LMDB, so you have to maintain the default values per Monitoring Object as described below. Besides that, you can simply route the Value Help call to the general form routine IEN_VH_RDB with the same interface.

```plaintext
FORM VH_CUSUST33

    TABLES PT_PARAMETER TYPE TT_PARAMETER
    PT_VH_CONTENT TYPE TT_CUSTH
    PT_APPMONS TYPE TT_APPMONS

    USING PF_PATTERN TYPE TF_PATTERN
    PF_MONOBJ TYPE TS_APPMONO-MONOBJ
    PF_KEYFIG TYPE TS_APPMONO-KEYFIG
    PS_APPMONS TYPE TS_APPMONS.

DATA:
    LS_INPUTTAB TYPE TS_INPUTTAB.
```

* There is no chance in ST-A/PI 01M SPO0 to get DBCON+SCHEMA from BPMon
* Framework or System Landscape. The only chance is the defaulting per
* Monitoring Object name, which can be applied automatically within
* Form IEN_VH_RDB. Using program /SSA/ENA in Expert Mode, you need to
* create the following configuration entry:
* Key 1 = DEFAULT_DBCON_MONOBJ
* Key 2 = <MonObj-Name>, e.g. CUSUST33
* Value 1 = <DBCON-Name>, e.g. ZTN_TPT_PWDF6567 ORA
* Value 2 = <Schema-Name>, e.g. CSL71_BASELINE

```plaintext
PERFORM IEN_VH_RDB

    TABLES PT_PARAMETER TYPE TT_PARAMETER
    PT_VH_CONTENT TYPE TT_CUSTH
    PT_APPMONS

    USING PF_PATTERN
    PF_MONOBJ
```
2 Data Collector (DC):

You can simply route the Data Collector call to the general form routine IEN_DC_RDB with the same interface.

FORM DC_CUSUST33

<table>
<thead>
<tr>
<th>TABLES</th>
<th>PT_INPUTTAB</th>
<th>TYPE</th>
<th>TT_INPUTTAB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PT_CUSTTABC</td>
<td>TYPE</td>
<td>TT_APPMONC</td>
</tr>
<tr>
<td></td>
<td>PT_CUSTTABL</td>
<td>TYPE</td>
<td>TT_APPMONL</td>
</tr>
<tr>
<td></td>
<td>PT_CUSTTABM</td>
<td>TYPE</td>
<td>TT_APPMONM</td>
</tr>
<tr>
<td></td>
<td>PT_CUSTTABC</td>
<td>TYPE</td>
<td>TT_APPMOO</td>
</tr>
<tr>
<td></td>
<td>PT_CUSTTABA</td>
<td>TYPE</td>
<td>TT_APPMONA</td>
</tr>
<tr>
<td></td>
<td>PT_CUSTTABS</td>
<td>TYPE</td>
<td>TT_APPMONS</td>
</tr>
<tr>
<td></td>
<td>PT_CUSTTAPB</td>
<td>TYPE</td>
<td>TT_APPMONP</td>
</tr>
<tr>
<td></td>
<td>PT_OUTPUTTAB</td>
<td>TYPE</td>
<td>TT_OUTPUTTAB</td>
</tr>
</tbody>
</table>

USING PF_SOLUTION TYPE TF_SOLUTION

CHANGING PF_SUBRC LIKE SY-SUBRC.

Note: You can directly route the Customer-Exit data collector call to the generic Form IEN_DC_RDB. The only limitation (in ST-API 01M SP00) is that the name of the custom monitoring object contains an 'S' at the third character, like CUSUST33 (instead of CUCUST33). This limitation will vanish in newer releases.

PERFORM IEN_DC_RDB

<table>
<thead>
<tr>
<th>TABLES</th>
<th>PT_INPUTTAB</th>
<th>TYPE</th>
<th>TT_INPUTTAB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PT_CUSTTABC</td>
<td>TYPE</td>
<td>TT_APPMONC</td>
</tr>
<tr>
<td></td>
<td>PT_CUSTTABL</td>
<td>TYPE</td>
<td>TT_APPMONL</td>
</tr>
<tr>
<td></td>
<td>PT_CUSTTABM</td>
<td>TYPE</td>
<td>TT_APPMONM</td>
</tr>
<tr>
<td></td>
<td>PT_CUSTTABC</td>
<td>TYPE</td>
<td>TT_APPMOO</td>
</tr>
<tr>
<td></td>
<td>PT_CUSTTABA</td>
<td>TYPE</td>
<td>TT_APPMONA</td>
</tr>
<tr>
<td></td>
<td>PT_CUSTTABS</td>
<td>TYPE</td>
<td>TT_APPMONS</td>
</tr>
<tr>
<td></td>
<td>PT_CUSTTAPB</td>
<td>TYPE</td>
<td>TT_APPMONP</td>
</tr>
<tr>
<td></td>
<td>PT_OUTPUTTAB</td>
<td>TYPE</td>
<td>TT_OUTPUTTAB</td>
</tr>
</tbody>
</table>

USING PF_SOLUTION TYPE TF_SOLUTION

CHANGING PF_PARAMETER TYPE TF_PARAMETER

CHANGING PF_SUBRC LIKE SY-SUBRC.

ENDFORM.

2 Detail Info (DI):

The general form routine IEN_DI_RDB has a different interface, and you need to do some preparation steps. Easiest way is to create the wrapper form routine Z_WRAP_DI_RDB (template see further below) inside program Z_BPM_ECU_COLLECTOR, and then you call it directly.

FORM DI_CUSUST33

<table>
<thead>
<tr>
<th>TABLES</th>
<th>PT_APPMONI</th>
<th>TYPE</th>
<th>TT_APPMONI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PT_ERRMSG</td>
<td>TYPE</td>
<td>TF_ERRMSG</td>
</tr>
<tr>
<td></td>
<td>PT_SY_SUBRC</td>
<td>TYPE</td>
<td>TF_SY_SUBRC</td>
</tr>
</tbody>
</table>

CHANGING PT_APPMONI TYPE TF_APPMONI

CHANGING PT_ERRMSG TYPE TF_ERRMSG

CHANGING PT_SY_SUBRC LIKE SY-SUBRC.

Note: Instead of calling IEN_DI_RDB directly, you need to copy the Wrapper Form Z_WRAP_DI_RDB into Program Z_BPM_ECU_COLLECTOR, to apply some special preparation steps, e.g. retrieving DBCON+SCHEMA.
2 General form routine to **prepare Detail Info call**:

```
FORM Z_WRAP_DI_RDB
  TABLES PT_APPMONI TYPE TT_APPMONI
  CHANGING PT_ERRMSG TYPE TF_ERRMSG
  PT_SY_SUBRC LIKE SY-SUBRC.

DATA:
  LV_SOLUTION  TYPE C LENGTH 50,
  LV_MONID     TYPE C LENGTH 20,
  LV_OBJECT    TYPE C LENGTH 8,
  LV_KEYFIG    TYPE C LENGTH 2,
  LS_INPUTTAB  TYPE TS_INPUTTAB,
  LS_CUSTTABA  TYPE TS_APPMONA.

° Extract main alert attributes
  MOVE PT_APPMONI-SOLUTION TO LV_SOLUTION.
  MOVE PT_APPMONI-MONID TO LV_MONID.
  MOVE PT_APPMONI-ALERTTYPE(8) TO LV_OBJECT.
  MOVE PT_APPMONI-ALERTTYPE+8(2) TO LV_KEYFIG.

° Check whether all import parameters for monitoring object are filled
  IF LV_SOLUTION IS INITIAL OR
    LV_MONID  IS INITIAL OR
    LV_OBJECT IS INITIAL OR
    LV_KEYFIG IS INITIAL.
  PT_SY_SUBRC = '90'.
  PT_ERRMSG =
    'Error at detail call: Monitoring object details missing'.
  RETURN.
ENDIF.

° Build dummy INPUTTAB, as this is required for Form IEN_DI_RDB
  LS_INPUTTAB-SOLUTION = LV_SOLUTION.
  LS_INPUTTAB-MONID    = LV_MONID.
  LS_INPUTTAB-DATARETR = 'R'.

° Try to get DBCON+SCHEMA from APPMONI additional alert attributes
  IF PT_APPMONI-INFO2 IS NOT INITIAL AND
    PT_APPMONI-INFO3 IS NOT INITIAL.
    LS_INPUTTAB-DBCON_NAME = PT_APPMONI-INFO2.
    LS_INPUTTAB-DB_SCHEMA  = PT_APPMONI-INFO3.
  ENDIF.

° Another try to get DBCON+SCHEMA from SMSY (works on SolMan only)
  IF LS_INPUTTAB-DBCON_NAME IS INITIAL.
    TRY.
      CALL FUNCTION 'DSWP_BPMRFC_DEST_GET'
        EXPORTING
          PF_SID = PT_APPMONI-SID
          PF_CLIENT = PT_APPMONI-MANDT
          PF_MONTy = 'APPMONI'
          PF_DATARETR = 'R'
        IMPORTING
          PF_DBCON_NAME = LS_INPUTTAB-DBCON_NAME
          PF_DB_SCHEMA  = LS_INPUTTAB-DB_SCHEMA
        EXCEPTIONS
          OTHERS = 1.
      IF SY-SUBRC <> 0.
```

PERFORM Z_WRAP_DI_RDB
  TABLES PT_APPMONI
  CHANGING PT_ERRMSG
  PT_SY_SUBRC.
ENDFORM.
ENDIF.
CATCH CX_SY_DYN_CALL_ILLEGAL_FUNC
    CX_SY_DYN_CALL_ILLEGAL_TYPE
    CX_SY_DYN_CALL_PARAM_MISSING
    CX_SY_DYN_CALL_PARAM_NOT_FOUND.
ENDTRY.
ENDIF.

° Last try to get DBCON+SCHEMA as MONID/MONOBJ defaults from RDB config
IF LS_INPUTTAB-DBCON_NAME IS INITIAL.
    LS_CUSTTABA-SOLUTION = LV_SOLUTION.
    LS_CUSTTABA-MONID = LV_MONID.
    LS_CUSTTABA-MONOBJ = LV_OBJECT.
    PERFORM NABRDB_DEFAULT_DBCON_BPMON
       USING LS_CUSTTABA
       CHANGING LS_INPUTTAB-DBCON_NAME
                 LS_INPUTTAB-DB_SCHEMA.
ENDIF.

° Use case 'S' => Predefined Select-Statement in virtual mon.obj.
PERFORM IEN_DI_RDB
   USING LV_SOLUTION
       LV_MONID
       LV_OBJECT
       LV_KEYFIG
       LS_INPUTTAB
   CHANGING PT_ERRMSG
       PT_SY_SUBRC.
ENDFORM.
The monitoring object “Triple Point Commodity SL Trade Monitoring” (NASTPCT1)
  o with custom-defined monitoring object
    ▪ Own complex data collector logic, reusing the remote DB access framework within the Business Process Monitoring Userexit

- Application Monitoring using Web services
  o Push-mechanism
    ▪ Non-ABAP system pushes alert information into Solution Manager (monitor NAWSPUSH) described in section “Monitoring object “Dependent Data Collector for Push WS” (NAWSPUSH)”
  o Pull-mechanism
    ▪ Solution Manager pulls alert information from Non-ABAP system (monitor NAWSPULL) described in section “Monitoring object “Data Collector for Pull WS” (NAWSPULL)”

- based on data extracted to SAP BW
  o Generic Infocube/DSO Reader for remote BW (monitor NABWICNX) described in section “The monitoring object “Generic Infocube/DSO Reader for remote BW” (NABWICNX)”.

Each section contains details about the available monitoring objects and the meaning of the parameters available in the setup. It does not describe the UI in detail or reflects changes in the Business Process Monitoring setup or monitoring UI that do not influence the general procedure. For details how to work in the Business Process Monitoring Setup UI which looks slightly different in the different releases refer to the general setup guide for Business Process Monitoring available under http://www.service.sap.com/~sapidb/01100035870006137532006E/BPMon_Setup_Guide.pdf. The overall procedure described in this document is applicable to all releases.

An appendix describes the different access possibilities in more detail including the prerequisites to connect a system like important parameters in Solution Manager containing the different connection parameters. Within the appendix you will also find the description of a test environment for remote DB accesses.

4 The monitoring object “Generic Table Entry Counter for remote DB” (NATABCNX)

4.1 Purpose
The “Generic Table Entry Counter for remote DB” provides the possibility to run generic database queries to count the number of entries (or the number of distinct values) in any remote database table. It supports a flexible filtering based on select-options on five fields of a table. Using ADBC (ABAP Database Connectivity) it can access a remote database table on an SAP-supported DBMS (database management system), using NativeSQL statements. This monitoring object is a substitute or workaround for areas, where no specific application monitor is available. Furthermore this monitor can help to avoid implementing the application monitoring customer-exit, as long as the collector result can be calculated by a simple SELECT statement on one single table or view.
This application monitor is part of the Business Process Monitoring framework within SAP Solution Manager.

4.2 Technical Prerequisites

The Generic Table Entry Counter is delivered with the add-on ST-A/PI starting with release 01M. The technical name of the Monitoring Object is NATABCNX.

All SAP Basis (NetWeaver) releases starting from release 7.00 and later are supported. It is recommended to run the data collector directly on SAP Solution Manager for this purpose.

To monitor an external (non-ABAP) database using ADBC, the corresponding secondary database connection must be maintained with transaction DBACOCKPIT. Please see appendix “Remote Database Connectivity” for more details.

Inside the Solution Manager, you must have a solution created with at least one business process maintained. The business process should have several business process steps. The system assigned to this step can be any system, but the ABAP based data collector is executed physically on any SAP NetWeaver ABAP system, preferably the Solution Manager itself. So the actual managed system is not the system where the data collector runs, but the system to which the remote database connection refers to. So you may want to create an own Logical Component for this non-ABAP/non-SAP system (instead of using the Solution Manager’s Logical Component), in order to have a unique assignment of the generated alert records.

This data collector must not be used to query databases of SAP ABAP-based systems. For that purpose you can use the “Generic Table Entry Counter” (technical name BOTABCNT), build especially to support database tables defined inside the SAP ABAP Data Dictionary. It used the preferred way of OpenSQL instead of NativeSQL.

4.3 Limitations

Because the data collector works on database tables, which are not defined inside the SAP ABAP Data Dictionary, there is no automatic support of certain build-in functionality, such as
- multi-lingual text labels for table columns,
- complex search helps on check tables or domain fix values,
- input and output conversion according to certain data types or user’s locale settings.

Search helps for defining filters are fed by selecting distinct field values from the corresponding table columns, which might be a performance-intensive query. Within the BPMon Setup Session, all filter criteria are stored as character fields.

The data collector accesses the remote DB data dictionary to query the table meta data (like column’s data types) in order to map the native data types into SAP dictionary data types and ABAP data types, for example to allow a detail display with the list viewer. However, it cannot be guaranteed that all native DBMS-dependent data types can be properly matched into a suitable ABAP data type. Therefore the output of certain data might look different compared to the user interface of native database tools or especially compared to the respective non-ABAP application, which may perform its own output conversion. There might be restrictions especially for number types (like maximum amount of decimals) or date types (like other than expected formats for dates and timestamps).
4.4 Performance Warning

The data collector can query any table that is accessible by the login user. Therefore it is in the responsibility of the user to avoid frequent selections on large tables. Keep in mind that the necessary SQL selection aggregates for COUNT and DISTINCT create expensive database selects, including full table scans!

Using this application monitor without care can cause severe performance degradation on the managed system.

Please consider your setup carefully and follow these golden rules:

• Avoid using this application monitor on large database tables
• Avoid running the data collection with a high monitoring frequency
• Avoid complex filtering, especially with patterns, ranges and exclusions
• Avoid filters on fields which are not supported by an index.

If you are in doubt, try to simulate the query with the “Test Environment for Remote DB Query”, to get an idea about the possible runtime. Furthermore, monitor the data collection runtime which is logged into a special column of the alert list (see chapter “Alert Display”).

4.5 Setup Procedure

4.5.1 Example use-case

To demonstrate the setup of the Table Entry Counter we choose a simple demo table APP_SALES_ORDER_ITEM of a fictive non-ABAP sales application with the following column structure:

<table>
<thead>
<tr>
<th>S...</th>
<th>Name</th>
<th>Datentyp</th>
<th>Länge</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDER_NUMBER</td>
<td>INTEGER</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>ITEM_NUMBER</td>
<td>INTEGER</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>QUANTITY</td>
<td>INTEGER</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>MATERIAL</td>
<td>INTEGER</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>ITEM_STATUS</td>
<td>CHARACTER</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>REQ_DLV_DATE</td>
<td>CHARACTER</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

Picture 1: Example data model

It contains some sample sales order items:

<table>
<thead>
<tr>
<th>ORDER_NUMBER</th>
<th>ITEM_NUMBER</th>
<th>QUANTITY</th>
<th>MATERIAL</th>
<th>ITEM_STATUS</th>
<th>REQ_DLV_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>10</td>
<td>1</td>
<td>3 created</td>
<td>20090630</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>20</td>
<td>4</td>
<td>6 created</td>
<td>20090630</td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td>10</td>
<td>50</td>
<td>1 picked</td>
<td>20090630</td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td>20</td>
<td>20</td>
<td>8 picked</td>
<td>20090630</td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td>30</td>
<td>2</td>
<td>9 open</td>
<td>20090630</td>
<td></td>
</tr>
<tr>
<td>1002</td>
<td>10</td>
<td>17</td>
<td>6 shipped</td>
<td>20081231</td>
<td></td>
</tr>
<tr>
<td>1003</td>
<td>10</td>
<td>12</td>
<td>2 waiting</td>
<td>20081231</td>
<td></td>
</tr>
<tr>
<td>1004</td>
<td>10</td>
<td>32</td>
<td>1 shipped</td>
<td>20090301</td>
<td></td>
</tr>
<tr>
<td>1005</td>
<td>10</td>
<td>1</td>
<td>7 open</td>
<td>20090701</td>
<td></td>
</tr>
<tr>
<td>1006</td>
<td>10</td>
<td>34</td>
<td>4 open</td>
<td>20090701</td>
<td></td>
</tr>
<tr>
<td>1007</td>
<td>10</td>
<td>68</td>
<td>7 open</td>
<td>20090701</td>
<td></td>
</tr>
<tr>
<td>1008</td>
<td>10</td>
<td>1</td>
<td>2 delivered</td>
<td>20090201</td>
<td></td>
</tr>
<tr>
<td>1008</td>
<td>20</td>
<td>1</td>
<td>4 delivered</td>
<td>20090201</td>
<td></td>
</tr>
<tr>
<td>1008</td>
<td>30</td>
<td>3</td>
<td>7 packed</td>
<td>20090201</td>
<td></td>
</tr>
<tr>
<td>1009</td>
<td>10</td>
<td>55</td>
<td>8 shipped</td>
<td>20090201</td>
<td></td>
</tr>
</tbody>
</table>

Picture 2: Example data
4.5.2 Simulate data collector result in monitored system

You can use the “Test Environment for Remote DB Query” (running program /SSA/ENA) to simulate the database query and to clarify the selection parameters. Assuming this remote database can be accessed with a DBCON entry called ZTN_DB6_2 and the table APP_SALES_ORDER_ITEM belongs to schema APP.

![Database Configuration](image1.png)

**Picture 3: Maintaining connection details for the test environment**

<table>
<thead>
<tr>
<th>Database Connection Name</th>
<th>ZTN_DB6_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Schema Name</td>
<td>APP</td>
</tr>
<tr>
<td>Table Name</td>
<td>APP_SALES_ORDER_ITEM</td>
</tr>
</tbody>
</table>

**Generated SQL Query**

```sql
SELECT "ORDER_NUMBER", "ITEM_NUMBER", "QUANTITY", "MATERIAL", "ITEM_STATUS", "RED_DLV_DATE"
FROM APP."APP_SALES_ORDER_ITEM"
```

![Example Data](image2.png)

**Picture 4: Selected example data**

Now as we have proved that the table content can be accessed from the ABAP stack, we want to setup the following KPIs for an application monitoring:

- Number of unprocessed sales items
- Sales orders with unprocessed items
- Lowest sales order number with unprocessed item(s)

4.5.3 Setup Procedure in SAP Solution Manager

In short: For the application monitoring, choose the monitoring object NATABCNX and specify the following selection criteria:

- Database connection name and schema name
- Table name
- Field name for filtering and select-options for actual filter values
  - this is available five times and will be used to generate a WHERE clause

**Detailed instructions:**

14. Call the SAP Solution Manager (trx DSWP or SOLMAN_WORKCENTER depending on the Solution Manager’s release).
15. Choose your solution.

16. Go to 'Operation Setup' and navigate to 'Solution Monitoring' => 'Business Process Monitoring'.
   Alternatively, you can start the Setup Session from the work center “Business Process Operations” from the link “Setup Business Process Monitoring” in the “Common Tasks” area.

17. One-time action only:
   Check 'Basic Settings' => 'Update Central Application Monitoring Repository': Press Button 'Load Monitors' and select the System ID where the ABAP-part of the data collector shall run (which preferably is the System ID of Solution Manager itself).


19. Check for your chosen business process: choose process steps to monitor.

20. Check for your chosen step: Choose type of monitor, here 'Application Monitors: Other area'.

21. Choose application monitor ‘**Generic Table Entry Counter for remote DB**’ [NATABCNX] via search help and name your monitoring object with a speaking name. This will create a sub-tree below 'Application Monitors' with the name of your monitoring object

![Picture 5: Selecting the Generic Table Entry Counter for Remote DB](image)

22. Check '<your monitoring object>' tab strip 'Detail Information': Customize the first counter line by double-clicking on counter ‘001’.

![Picture 6: Parameter entry for NATABCNX](image)

The first two parameters "**Database Connection**" and "**Database Schema**" are to define the target database server (using a DBCON entry name) and database schema for the remote database connection. The preferred way to maintain these attributes is inside the System Landscape definition (transaction SMSY/LMDB). The business process step refers to a logical component, which itself points to a system that uses a database. At the
maintenance screen of this database you should enter the DBCON connection name and schema name into the free attributes called "Remote DB Conn. for Appl.Mon." and "DB Schema for Application Data". This provides a central place to store this technical information, which is then valid for all application monitors using a remote database connection to this database. In such a case it is not necessary to maintain the two monitoring object parameters "Database Connection" and "Database Schema" in the setup session. If you maintain the parameters in the session, this will override the SMSY/LMDB attributes and the monitoring will use your entered values instead.

For more information regarding the usage of a remote DB connection, please see appendixes “Remote Database Connectivity” and “Derive Connection Entry”.

As the third parameter, enter the "**Table Name**". The search help loads the available table names from the remote database. For performance reasons, make sure to enter a pattern (like APP* for all tables prefixed by APP) to restrict the selection. The table name is the only mandatory parameter, whereas the following filters are optional.

The optional fourth parameter called "**Display Columns**" allows restricting the scope of the detail display screen. Instead of a SELECT * only the specified column names will be loaded from the remote database table, to allow hiding some technical fields. For each field name you can add a descriptive field label text that will be used as column header in the detail display. Just postfix a double-dash (SQL comment "--") and the text as shown in Picture 7.

**Example:**

```
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Label Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUANTITY</td>
<td>Quantity</td>
</tr>
<tr>
<td>ORDER_NUMBER</td>
<td>Order Number</td>
</tr>
<tr>
<td>MATERIAL</td>
<td>Material Number</td>
</tr>
<tr>
<td>ITEM_STATUS</td>
<td>Item Status</td>
</tr>
<tr>
<td>ITEM_NUMBER</td>
<td>Item Number</td>
</tr>
<tr>
<td>REQ_DLVR_DATE</td>
<td>Req.Dlvr.Date</td>
</tr>
</tbody>
</table>
```

*Picture 7: Defining display columns*

Otherwise you would see the technical field names only, as the remote database dictionary access does not provide multi-lingual text labels as known from the SAP ABAP dictionary.

As the optional fifth parameter "**Filter 1 Fieldname**", enter the field name for the first filter, which obviously must be a valid field of the selected database table. A search help dialog is also shown here, which shows all fields of the selected table, based on the remote database dictionary.

In the sixth line, enter the select criteria "**Filter 1 Sel.options**" for the first filter mentioned above, which represent the actual filter field values. When you have already entered the table name and the field name for filtering, the system tries to provide an entry help from the remote database table column content. However, this may be performance-intensive due to the selection method of distinct values, so please use with care. Please note to enter all values in the native database format, for example a date 2008/09/24 might be entered as 20080924, or timestamps are usually entered in a 24-hour format without separators. Please see the appendix on how to implement a dynamic date filtering, using relative dates instead of absolute values.

Repeat these steps for further filters if required.
Example: The KPI “Number of unprocessed sales items” would be defined as all sales order items in table APP_SALES_ORDER_ITEM which have an ITEM_STATUS of “created”, “open” or “waiting”.

When you are finished with the setup of the header parameters, press button “Validation Check” to validate the input and perform a connection test.

23. Check '<your monitoring object>' tab strip ‘Monitoring Schedule’: Enter the data for the scheduling of the collector job in the monitored system, i.e. weekdays, collection period or fixed start time.

24. Check '<your monitoring object>' tab strip 'Key Figures': choose the key figure(s) to focus on. There are three key figures available.

When marking them, after saving you see new sub-checks where you can do additional customizing on key figure level.

- “Number of Counted Entries” (with given filter criteria)
- “Number of Distinct Values” (on single field name and with given filter criteria)
  a. Enter the parameter “Distinction Fieldname”, that is the name of the table field for which the different values are to be counted. Please make use of the value help, which only displays valid fields of the selected table.
- “Value of Single Field” (with or without aggregate and with given filter criteria)
  a. As additional customizing at key figure level, you have to enter the name of the single field for which you want to read its value (“Selection Fieldname”). Ideally the data type of this field is an integer. Numeric fields with decimals will be rounded. For other types the result may be undefined or you would even get an error message in the alert text regarding a conversion error. Make sure that the where-clause defined by the filter
fields selects one single row only. Otherwise the result might be not unique, which will be indicated in the alert text.

b. In addition you can optionally enter an "Aggregate Function". If the select results in multiple rows, one single integer value is returned due to aggregation, for example MAX(ERRORCODE) would return the highest error code from the error table column. Please use the value help to select from possible aggregate functions:
   i. Default: None
   ii. MIN: Minimum value
   iii. MAX: Maximum value
   iv. AVG: Average value
   v. SUM: Sum of values

- For each key figure, you can customize an "Alternative Alert Text", using certain placeholder variables.
  a. The standard key figure alert texts return a rather technical alert message
     - “Number of entries in table <table> = <counter>” or
     - “Number of distinct values in field <field> of table <table> = <counter>”
     This is automatically proposed if you leave the parameter “Alternative Alerttext” empty.
  b. You can use any free text together with the following placeholders to overwrite the standard alert text with your own alert text:
     - $COUNT returns the measured value from the selection
     - $TABLE returns the database table name
     - $FIELD returns the table’s field name
  c. Example for our use-case in key figure “Number of Counted Entries”:
     - “$COUNT unprocessed sales items”

- For each key figure, the parameter "Detailcall Headertext" allows to store a screen title for the detail display, to explain the user what kind of output is shown.

![Picture 11: Detailcall headertext parameter](image)

- For each key figure you can enter a threshold for a YELLOW alert and a threshold for a RED alert as an absolute value, also supporting a four-step rating with minimum and maximum thresholds pairs (RED >= YLW >= Value >= YLW >= RED).

![Picture 12: Thresholds for NATABCNX](image)

25. At node 'Analysis & Monitoring Tools', do not select an ABAP-based analysis tool like a transaction code. Instead, later on inside the monitoring session you will have a button called “Detail Info” which will display the relevant table entries inside a list viewer according to the setup filters.
26. Once you have entered all relevant information for the monitoring objects you want to monitor, generate the monitoring customizing and activate the monitoring within the 'Business Process Monitoring' session.

**4.6 SQL Query Format**

The key figure "Number of Counted Entries" creates an SQL query in the following format:

```sql
SELECT COUNT(*)
FROM table
WHERE field1 IN filterrange1
  AND field2 IN filterrange2
  AND field3 IN ... 
```

The key figure "Number of Distinct Values" creates an SQL query in the following format:

```sql
SELECT COUNT( DISTINCT field )
FROM table
WHERE field1 IN filterrange1
  AND field2 IN filterrange2
  AND field3 IN ... 
```

The key figure "Value of Single Field" creates an SQL query in the following format:

```sql
SELECT field
or
SELECT aggregate(field)
FROM table
WHERE field1 IN filterrange1
  AND field2 IN filterrange2
  AND field3 IN ... 
```

**4.7 Result in Monitoring Session**

**4.7.1 Alert Display in Solution Manager**

![Alert display for "Number of counted entries"](image)

The column “Alert Message” shows the first 120 characters of the alert text, which was in this case assembled from the customized alternative alert text template. In addition there are some additional columns filled, like Database connection name, Table name and Runtime (which shows how long it took for the data collector to query the key figure).
Alert display for key figure “Number of Distinct Values”:
The column “Alert Message” shows the first 120 characters of the alert text, which might be assembled from your customized alternative alert text template as well. Furthermore there are some additional columns filled, like Database connection name, Table and Field name (just repeats which field was used for the distinction) and Runtime (which shows how long it took for the data collector to query the key figure).

Alert display for key figure “Value of Single Field”:
The column “Alert Message” shows the first 120 characters of the alert text, which might be assembled from your customized alternative alert text template as well. Furthermore there are some additional columns filled, like Database connection name, Table and Field name (just repeats which field was used for the distinction) and Runtime (which shows how long it took for the data collector to query the key figure).

4.7.2 Detail Display call into monitored system
Mark an alert and press push button “Detail Info”. This loads the relevant remote database table entries into an SAP-ABAP-based list viewer. The display might slightly differ from what you see with native database tools or within the UI of the external application. Not all table columns might find a matching ABAP type or might be represented in some other display format.
Please note that you always see the current selection result, even if you have selected an older alert. Reason is that the selection result for the regular key figure calculation is not stored, but rather re-selected during the detail display call.

![Picture 14: Detail information for key figure “Number of Counted Entries”](image)

The detailed info for the number of counted entries just displays the table rows with the affected entries (applying the same filtering as the data collector). Please note that a maximum of 10,000 rows is shown. You can also see a header text as defined in the corresponding setup parameter.

![Picture 15: Detail information for key figure “Number of Distinct Values”](image)
The detailed info for the number of distinct values displays an overview with each distinct field value and its number of occurrences (applying the same filtering as the data collector). In our example it shows all sales order numbers and the number of affected open items. Obviously the sum of all occurrences should be equal to the value of key figure “Number of Counted Entries”.

**Detail information for key figure “Value of Single Field”**
The detailed info for the value of a single field looks identical to the one from the first key figure “**Number of Counted Entries**”. Instead of just showing one row with one field containing its single or aggregated value, the display shows all affected rows for a better traceability of the aggregated result.
5 Custom-specific key figures for 3rd-party or legacy systems

5.1 Purpose
The BPMon framework allows storing pre-defined SQL-Statements in its repository, which act as data selection method for monitoring key figures. As long as the whole data collection (that is calculating the alert measured value) can be described by a single Select-Statement, there is no need to create additional special ABAP code. You can just define the key figure with its Select-Statement, together with a small program entry point, which delegates all the complex connection and remote database handling to general framework subroutines. The following procedure explains how to implement the BPMon customer-exit for additional Remote-DB Application Monitoring key figures, by using the framework of pre-defined Select-Statements.

5.2 Technical Prerequisites

5.3 Preparation
Prepare the required SQL-Select-Statements for
- **Counter** (calculating alert measured value; mandatory) => SELECT COUNT(*) FROM...
- **Display** (detail display of selection result; optional) => SELECT <fieldlist> FROM...

You can test your queries with program /SSA/ENA in processing mode ‘Free Select Query’ (see chapter “Test Environment for Remote DB Query (program /SSA/ENA)”).
Example for Counter:

**SELECT COUNT(*) FROM CSL71_BASELINE.CSL_MSG_MONITOR**

**COUNT(*)**

2016

Picture 16: Key figure Count simulation in Test Environment
Picture 17: Key figure Count simulation in Test Environment
5.4 Maintain Monitoring Object Repository

7. Run program /SSA/EXM for P_PROJID = ‘ECU’ in change mode.

8. Create the Monitoring Object. For ST-A/PI release 01M the name must start with an ‘S’ (like for the standard monitors in project ENA).

![Picture 18: Definition of Monitoring Object](image)

9. Go to “Customize Parameters”
   a. Set Data Retrieval = ‘Remote DB Query’.

![Picture 19: Definition of Monitoring Object Header Attributes](image)

   b. No Parameters on header level
   c. Set Application = CUSTOMER

10. Go to “Define Key Figures” and create all required key figures (one per each different query)

![Picture 20: Definition of Monitoring Object’s Key Figures](image)
11. Mark Key Figure and go to “Customize Parameters”

   a. On tab ‘Attributes’, set Additional Functionalities = ‘Detail Info’ if this needs to be supported.
      Validation Check and Default Value are not yet possible in customer-exit.
      Enter text labels for Threshold 1+2 (for 2-step rating) and optionally also Threshold 3+4 (for 4-step-rating).

   b. On tab ‘Parameters’ you can define user-customizable parameters or select-options, which will appear on the counter popup in BPMon Setup UI. You need to maintain the table name as Tech Name 1 and the field name as Tech Name 2 to get an automatic value help based on distinct values in that table column.

   c. On tab ‘ADBC Statement Data Collection’ maintain the SQL statement fragments for SELECT-, FROM-, WHERE- and GROUP-BY-clause. The SELECT-clause must return an integer value, e.g. with using a COUNT-aggregate. The FROM-clause is mandatory and may contain the placeholder /*SCHEMA*/, which will be replaced with the remote DB schema name at runtime. The WHERE-clause is optional and should contain all hard-coded filters (in contrast to user-customizable filters as above). The GROUP-BY-clause is optional and should
be used in conjunction with a DISTINCT-aggregate.

**Picture 23: Definition of SQL Statement for Counting**

d. On tab "ADB Statement Detail List" maintain the SQL statement fragments for SELECT-, FROM-, WHERE-, GROUP-BY- and ORDER-BY-clause. The SELECT-clause must contain the field list for the columns to be displayed, using a qualified column name ("TABLE"."FIELD"). The FROM-clause is mandatory and may contain the placeholder /*SCHEMA*/, which will be replaced with the remote DB schema name at runtime. The WHERE-clause is optional and should contain all hard-coded filters (in contrast to user-customizable filters as above). Typically the FROM- and WHERE-clause are identical to those used in the Data Collection query. The GROUP-BY-clause and ORDER-BY-clause are optional.

**Picture 24: Definition of SQL Statement for Detail Display**

12. Mark Key Figure and go to "Detail List"
a. For each key figure, you can define the text labels for the Short/Medium/Long column headings of the list viewer during detail display calls. The columns ‘DDIC Table’ and ‘DDIC Field’ must match with the identifiers used as field list in the SELECT-clause.

![Image of a list viewer interface]

Picture 25: Definition of Column List for Detail Display

5.5 Maintain Monitoring Object Coding


10. Make sure that the following **Includes** are included:

    INCLUDE /SSA/IET.
    INCLUDE /SSA/IES.
    INCLUDE /SSA/IEN.

11. At event **LOAD-OF-PROGRAM** call form routine `NABRDB_CONFIG_LOAD`:

    LOAD-OF-PROGRAM.
    PERFORM NABRDB_CONFIG_LOAD.

12. For each Monitoring Object, maintain the entry form routines to be called from BPMon framework

    a. Value Help
    b. Data Collector
    c. Detail Info

13. **Value Help** (VH):

    There is no way to derive **DBC+SCHEMA** from **SMSY/LMDB**, so you have to maintain the default values per Monitoring Object as described below. Besides that, you can simply route the Value Help call to the general form routine `IEN_VH_RDB` with the same interface.

    ```
    FORM VH_CUSUST33 TABLES PT_PARAMETER TYPE TT_PARAMETER  
    PT_VH_CONTENT TYPE TT_CUSTVH  
    PT_APPMONS TYPE TT_APPMONS  
    USING PF_PATTERN TYPE TF_PATTERN  
    PF_MONOBJ TYPE TS_APPMONO-MONOBJ  
    PF_KEYFIG TYPE TS_APPMONO-KEYFIG  
    PS_APPMONS TYPE TS_APPMONS.
    ```
DATA:
  LS_INPUTTAB TYPE TS_INPUTTAB.

* There is no chance in ST-A/PI 01M SP00 to get DBCON+SCHEMA from BPMon
* Framework or System Landscape. The only chance is the defaulting per
* Monitoring Object name, which can be applied automatically within
* Form IEN_VH_RDB. Using program /SSA/ENA in Expert Mode, you need to
* create the following configuration entry:
* Key 1 = DEFAULT_DBCON_MONOBJ
* Key 2 = <MonObj-Name>, e.g. CUSUST33
* Value 1 = <DBCON-Name>, e.g. ZTN_TPT_PwDF6567_ORA
* Value 2 = <Schema-Name>, e.g. CSL71_BASELINE

PERFORM IEN_VH_RDB
  TABLES PT_PARAMETER
         PT_VH_CONTENT
         PT_APPMONS
  USING PF_PATTERN
          PF_MONOBJ
          PF_KEYFIG
          PS_APPMONS
          LS_INPUTTAB.

ENDFORM.

14. Data Collector (DC):
   You can simply route the Data Collector call to the general form routine IEN_DC_RDB
   with the same interface.

FORM DC_CUSUST33
  TABLES PT_INPUTTAB TYPE TT_INPUTTAB
        PT_CUSTTABC TYPE TT_APPMONC
        PT_CUSTTABL TYPE TT_APPMONL
        PT_CUSTTABM TYPE TT_APPMONN
        PT_CUSTTABO TYPE TT_APPMONO
        PT_CUSTTABA TYPE TT_APPMONA
        PT_CUSTTABS TYPE TT_APPMONS
        PT_CUSTTABP TYPE TT_APPMONP
        PT_OUTPUTTAB TYPE TT_OUTPUTTAB
  USING PF_SOLUTION TYPE TF_SOLUTION
          PF_PARAMETER TYPE TF_PARAMETER
  CHANGING PF_SUBRC LIKE SY-SUBRC.

* You can directly route the Customer-Exit data collector call to the
* generic Form IEN_DC_RDB. The only limitation (in ST-A/PI 01M SP00) is
* that the name of the custom monitoring object contains an 's' at the
* third character, like CUST3S3 (instead of CUSUST33). This limitation
* will vanish in newer releases.

PERFORM IEN_DC_RDB
  TABLES PT_INPUTTAB
        PT_CUSTTABC
        PT_CUSTTABL
        PT_CUSTTABM
        PT_CUSTTABO
        PT_CUSTTABA
        PT_CUSTTABS
        PT_CUSTTABP
        PT_OUTPUTTAB
  USING PF_SOLUTION
        PF_PARAMETER
  CHANGING PF_SUBRC.

ENDFORM.
15. **Detail Info (DI):**

The general form routine IEN_DI_RDB has a different interface, and you need to do some preparation steps. Easiest way is to create the wrapper form routine Z_WRAP_DI_RDB (template see further below) inside program Z_BPM_ECU_COLLECTOR, and then you call it directly.

```
FORM DI_CUSUST33
  TABLES PT_APPMONI TYPE TT_APPMONI
  CHANGING PT_ERRMSG TYPE TF_ERRMSG
  PT_SY_SUBRC LIKE SY-SUBRC.

* Instead of calling IEN_DI_RDB directly, you need to copy the wrapper
* Form Z_WRAP_DI_RDB into Program Z_BPM_ECU_COLLECTOR, to apply some
* special preparation steps, e.g. retrieving DBCON+SCHEMA.

PERFORM Z_WRAP_DI_RDB
  TABLES PT_APPMONI
  CHANGING PT_ERRMSG
  PT_SY_SUBRC.

ENDFORM.
```

16. **General form routine to prepare Detail Info call:**

```
FORM Z_WRAP_DI_RDB
  TABLES PT_APPMONI TYPE TT_APPMONI
  CHANGING PT_ERRMSG TYPE TF_ERRMSG
  PT_SY_SUBRC LIKE SY-SUBRC.

DATA:
  LV_SOLUTION TYPE C LENGTH 50,
  LV_MONID TYPE C LENGTH 20,
  LV_OBJECT TYPE C LENGTH 8,
  LV_KEYFIG TYPE C LENGTH 2,
  LS_INPUTTAB TYPE TS_INPUTTAB,
  LS_CUSTTABA TYPE TS_APPMONA.

* Extract main alert attributes
  MOVE PT_APPMONI-SOLUTION TO LV_SOLUTION.
  MOVE PT_APPMONI-MONID TO LV_MONID.
  MOVE PT_APPMONI-ALERTTYPE(8) TO LV_OBJECT.
  MOVE PT_APPMONI-ALERTTYPE+8(2) TO LV_KEYFIG.

* Check whether all import parameters for monitoring object are filled
  IF LV_SOLUTION IS INITIAL OR
     LV_MONID IS INITIAL OR
     LV_OBJECT IS INITIAL OR
     LV_KEYFIG IS INITIAL.
     PT_ERRMSG = '90'.
     PT_SY_SUBRC = '90'.
     RETURN.
  ENDIF.

* Build dummy INPUTTAB, as this is required for Form IEN_DI_RDB
  LS_INPUTTAB-SOLUTION = LV_SOLUTION.
  LS_INPUTTAB-MONID = LV_MONID.
  LS_INPUTTAB-DATARETR = 'R'.

* Try to get DBCON+SCHEMA from APPMONI additional alert attributes
  IF PT_APPMONI-INFO2 IS NOT INITIAL AND
     PT_APPMONI-INFO3 IS NOT INITIAL.
     LS_INPUTTAB-DBCON_NAME = PT_APPMONI-INFO2.
     LS_INPUTTAB-DB_SCHEMA = PT_APPMONI-INFO3.
  ENDIF.
```
* Another try to get DBCON+SCHEMA from SMSY (works on SolMan only)

IF LS_INPUTTAB-DBCON_NAME IS INITIAL.
  TRY.
    CALL FUNCTION 'DSWP_BPM_RFC_DEST_GET'
      EXPORTING
        PF_SID = PT_APPMONI-SID
        PF_CLIENT = PT_APPMONI-MANDT
        PF_MONTY = 'APPMON'
        PF_DATARETR = 'R'
      IMPORTING
        PF_DBCON_NAME = LS_INPUTTAB-DBCON_NAME
        PF_DB_SCHEMA = LS_INPUTTAB-DB_SCHEMA
      EXCEPTIONS
        OTHERS = 1.
    IF SY-SUBRC <> 0.
      ENDIF.
    CATCH CX_SY_DYN_CALL_ILLEGAL_FUNC
      CX_SY_DYN_CALL_ILLEGAL_TYPE
      CX_SY_DYN_CALL_PARAM_MISSING
      CX_SY_DYN_CALL_PARAM_NOT_FOUND.
    ENDTry.
  ENDIF.
ENDIF.*

* Last try to get DBCON+SCHEMA as MONID/MONOBJ defaults from RDB config

IF LS_INPUTTAB-DBCON_NAME IS INITIAL.
  LS_CUSTTABA-SOLUTION = LV_SOLUTION.
  LS_CUSTTABA-MONID = LV_MONID.
  LS_CUSTTABA-MONOBJ = LV_OBJECT.
  PERFORM NABRDB_DEFAULT_DBCON_BPMON
    USING LS_CUSTTABA
    CHANGING LS_INPUTTAB-DBCON_NAME
              LS_INPUTTAB-DB_SCHEMA.
  ENDIF.

* Usecase 'S' => Predefined Select-Statement in virtual mon.obj.

PERFORM IEN_DI_RDB
  USING LV_SOLUTION
       LV_MONID
       LV_OBJECT
       LV_KEYFIG
       LS_INPUTTAB
  CHANGING PT_ERRMSG
           PT_SY_SUBRC.

ENDFORM.
6 The monitoring object “TriplePoint Commodity SL Trade Monitoring” (NASTPCT1)

6.1 Purpose
The application monitor “TriplePoint Commodity SL Trade Monitoring” provides the possibility to query a set of predefined key figures directly from a TriplePoint CSL database. Using ADBC (ABAP Database Connectivity) the SAP Solution Manager can establish a secondary database connection to the TriplePoint CSL Trade application database. This application monitor is part of the Business Process Monitoring framework within SAP Solution Manager. It serves as an example how to implement key figures by storing pre-defines Select-Statements inside the BPMon Repository.

6.2 Technical Prerequisites
For the ABAP-based data collector, all SAP Basis (NetWeaver) releases starting from release 7.00 and later are supported. ST-A/PI Add-On with Release 01M (2010.1) or higher needs to be installed. The technical name of the Monitoring Object is NASTPCT1. While the collector can run on any ABAP-based system, it is highly recommended to run the data collector directly on SAP Solution Manager. The SAP Solution Manager needs software component ST-SER to be on minimum release 2010.1 (part of Support Package Stack SPS 23). This allows running local data collectors, even if they are linked to a Logical Component that does not match the Solution Manager’s System-ID.

To monitor a remote (non-ABAP) database using ADBC, the corresponding secondary database connection must be maintained with transaction DBACOCKPIT. Please see appendix “Remote Database Connectivity” for more details.

Inside the Solution Manager, you must have a solution created with at least one business process maintained. The business process should have several business process steps. The actual system assigned to the business process step is not the system where the data collector runs, but the logical system you want to monitor and to which the remote database connection refers to. So you may want to create an own Logical Component for this non-ABAP/non-SAP system (instead of using the Solution Manager’s Logical Component), in order to have a unique assignment of the generated alert records.

6.3 Limitations
Because the data collector works on database tables, which are not defined inside the SAP ABAP Data Dictionary, there is no automatic support of certain build-in functionality, such as
- complex search helps on check tables or domain fix values,
- input and output conversion according to certain data types or user’s locale settings.

Search helps for defining filters are fed by selecting distinct field values from the corresponding table columns in the remote database, which might be a performance-intensive query. Within the BPMon Setup Session, all filter criteria are stored as character fields.

The data collector accesses the remote DB data dictionary to query table meta data like the column’s data types, in order to map the native data types into SAP dictionary data types and ABAP data types, for example to allow a detail display with the list viewer. However, it cannot be guaranteed that all native DBMS-dependent data types can be properly matched into a suitable ABAP data type. Therefore the output of certain data might look different
compared to the user interface of native database tools or to the actual TriplePoint CSL application UI. There might be restrictions especially for number types (like maximum amount of decimals) or date types (like other than expected formats for dates and timestamps).

**6.4 Setup Procedure in SAP Solution Manager**

1. Call the SAP Solution Manager (trx DSWP or SOLUTION_MANAGER) or the work center “Business Process Operation”.

2. Choose your solution.

3. Go to 'Operation Setup' and navigate to 'Solution Monitoring' => 'Business Process Monitoring'.

4. One-time action only:
   Check 'Basic Settings' => 'Update Central Application Monitoring Repository': Press Button 'Load Monitors' and select the System ID where the ABAP-part of the data collector shall run (which preferably is the System ID of Solution Manager itself).


6. Check for your chosen business process: choose process steps to monitor.
   Note: Typically you have multiple steps for a business process like TriplePoint CSL Trade. All key figures are contained in one single monitoring object. You can chose this monitoring object multiple times across the different business process steps, and then assign those key figures only that are suitable for monitoring a certain step. You also need to setup the monitoring object multiple times, if you want to use the key figures with different selection criteria, for example to separate data by organizational or geographical means together with separate alert notification recipients.

7. Each business process step is assigned to a logical component. It is mandatory the remote database connection name can be derived from the System Landscape Information stored for this Logical Component. Please see appendix “Derive Connection Entry” for more details.

8. Check for your chosen step: Choose type of monitor, here 'Application Monitors: Other area'.

9. Choose application monitor ‘**Triple Point Commodity SL Trade Monitoring**’ [NASTPCT1] via search help and name your monitoring object with a speaking name.
   This will create a sub-tree below 'Application Monitors' with the name of your monitoring object.

   ![Picture 26: Selecting the monitoring object](image)

10. Check ‘<your monitoring object>’ tab strip ‘Monitoring Schedule’: Enter the data for the scheduling of the collector job in the system where the collector is executed physically, i.e.
weekdays, collection period or fixed start time.

11. Check 'your monitoring object' tab strip 'Key Figures': choose the key figure(s) to focus on. The key figures shown in Picture 27 are available:

![List of key figures for TriplePoint Commodity SL Trade Monitoring](image)

After saving you see new sub-checks for the marked key figures where you can do additional customizing on key figure level. Press on Counter “001” to access a popup screen for defining filter criteria.

The available filters depend on the actual key figure. Typical filter types are:

- Fix values (only used for “EOD Phase” at first key figure)
  a. The search help shows a list of predefined fixed values.
- Variable values (such as Instrument Type, Location, Counterpart)
  a. The search help will query the remote database for distinct field values in the corresponding check table column. You can also use complex selection criteria like multiple values, patterns or ranges.
- Date ranges (such as Last Modify Date)
  a. Although it is possible to enter fixed dates, this is not useful as you would only get a static filter setup. Instead you would like to have relative date filters, like all records created yesterday, or all errors occurred during last week. For this purpose, please refer to appendix “Dynamic Date Filtering”.

![Example for a parameter set](image)

- For each key figure you can enter a threshold for a YELLOW alert and a threshold for a RED alert as an absolute value.

![Maintaining thresholds](image)

12. At node 'Analysis & Monitoring Tools', do not select an ABAP-based analysis tool like a transaction code. Instead, later on inside the monitoring session you will have a button
called “Detail Info” which will display the relevant table entries inside a list viewer according to the setup filters.

13. Once you have entered all relevant information for the monitoring objects you want to monitor, generate the monitoring customizing and activate the monitoring within the 'Business Process Monitoring' session.

6.5 List of Key Figures for CSL Trade

Available Key Figures

- EOD: Failed End-of-Day Process records
- Scheduling: Failed Goods Receipts / Goods Issues
- Scheduling: Failed Transfer Match records in CSL
- Master Data: Failed Reference Data Request records
- GTM Trades: Total number of Custom Events
- GTM Trades: Failed Trade Modification records
- GTM Trades: Failed Trade Reference Number records
- GTM Trades: Failed Trade Release records (no SO/PO)
- GTM Trades: Failed Trade Voiding records (no SAP update)
- TRM Trades: Failed TRM Clearing Broker records
- TRM Trades: Failed TRM Floor Broker records

Choose the key figures that you need to monitor the corresponding part of your business process or interfaces. For each key figure you will be able to maintain additional filter criteria. If you want to monitor multiple business process steps, you should set it up for each relevant step and only chose those key figures relevant to the specific step. You also need to setup the monitoring object multiple times, if you want to use the key figures with different selection criteria, for example to separate data by organizational or geographical means together with separate alert notification recipients.

6.5.1 Key Figure "EOD: Failed End-of-Day Process records"

This key figure counts the number of failed records during the End-of-Day Process.

Optional Parameters:
- EOD Phase*
- Run Date

*Possible EOD Phases are
- Mark Curves COB
- Generate / Update Obligations
- Storage / Inventory Valuation
- EOD Revaluation
- Mark Revaluation COB
- Execute EOD
6.5.2 Key Figure "Scheduling: Failed Goods Receipts / Goods Issues"

Once a trade gets saved in SAP system the next step is to create either an inbound delivery or an outbound delivery. An inbound delivery is always followed by GR which stands for goods receipt and an out bound delivery is always followed by GI which means Goods Issued. Both GR/GI are triggered in SAP and it is possible for them to fail in CSL. This key figure counts the number of failed GI/GR records during the End-of-Day Process, which means they could not be processed successfully in CSL.

Optional Parameters:
- Instrument Type
- Location
- Counterpart
- Last Modify Date

6.5.3 Key Figure "Scheduling: Failed Transfer Match records in CSL"

This key figure counts the number of failed Transfer Match records. It can be used to obtain details of all such Trades for which transfer/match got failed in CSL though they were successful in SAP. It should be noted that an inbound delivery is used with Buy Trade while an outbound delivery is used with Sell Trade.

Optional Parameters:
- Instrument Type
- Location
- Counterpart
- Last Modify Date

6.5.4 Key Figure "Master Data: Failed Reference Data Request records"

This key figure counts the number of failed Reference Data Request records, which flow from SAP to CSL. This Reference Data Request gets already intercepted when received from SAP at the Messaging Monitor Layer in CSL. In case of a problem encountered the status field is either set to pending, error or warning.

Optional Parameters:
- Operation Name
- Last Modification Date

6.5.5 Key Figure "GTM Trades: Total number of Custom Events"

While capturing a Trade in the CSL system, the user can add custom events to a Trade if required. There is one custom event called Title Transfer Date which is auto populated and need not be send to SAP. This is a kind of exception and apart from this all other custom events always flow to SAP. Every custom event is characterized by a code, a start date and an end date. This key figure counts the number of custom events which have been added on all the trades.

Optional Parameters:
- Instrument Type
6.5.6 Key Figure "GTM Trades: Failed Trade Modification records"

This KPI aims to track all records for which a Trade modification was unsuccessful in SAP. A Trade may be in sync on both of the systems when it is saved for the first time but further modifications to a Trade can lead to some kind of data discrepancy and it may not get saved in the SAP system. This key figure counts the number of failed Trade Modification records.

Optional Parameters:
- Instrument Type
- Location
- Counterpart
- Last Modify Date

6.5.7 Key Figure "GTM Trades: Failed Trade Reference Number records"

This key figure counts the number of all active Trades which have not flown to SAP, i.e. where these Trades cannot be found in SAP using the CSL Trade number.

Optional Parameters:
- Instrument Type
- Location
- Counterpart
- Last Modify Date (Header)
- Last Modify Date (Term)

6.5.8 Key Figure "GTM Trades: Failed Trade Release records (no SO/PO)"

This key figure counts the number of failed Trade Release records. It provides details of all such Trades or Contracts for which there was no SO/PO order generated in SAP. We may or may not have the reference number (DEAL_ID) generated for all such Trades.

Optional Parameters:
- Instrument Type
- Location
- Counterpart
- Last Modify Date

6.5.9 Key Figure "GTM Trades: Failed Trade Voiding records (no SAP update)"

This key figure counts the number of failed Trade Voiding records. Trade voiding is a feature which is supported by current CSL-SAP Integration framework. While voiding a trade it is possible that the update may not get reflected in SAP.
6.5.10 **Key Figure "TRM Trades: Failed TRM Clearing Broker records"**

This key figure is meant to be used for TRM Trades. It counts the number of failed TRM Clearing Broker records, during Add Flow and Change Flow.

**Optional Parameters:**
- Instrument Type
- Location
- Counterpart
- Last Modify Date

6.5.11 **Key Figure "TRM Trades: Failed TRM Floor Broker records"**

This key figure is meant to be used for TRM Trades. It counts the number of failed TRM Floor Broker records, during Add Flow and Change Flow.

**Optional Parameters:**
- Instrument Type
- Location
- Counterpart
- Last Modify Date

6.6 **Performance Warning**

Because some of the key figures need complex database queries, the data collector may cause some performance degradation on the monitored system. Please consider your setup carefully and follow these golden rules:

- Avoid running the data collection with a high monitoring frequency
- Avoid complex filtering, especially with patterns, ranges and exclusions
- Avoid filters on fields which are not supported by an index.

Monitor the data collection runtime which is logged into a special column of the alert list (see chapter “Alert Display”).
6.7 Result in Monitoring Session

6.7.1 Alert Display in Solution Manager

Picture 30: Sample alert overview for all key figures

Picture 31: Sample alert display for key figure “EOD: Failed End-of-Day Process records”

In the alert history list, the column Alert Message shows the alert text, which is assembled from the key figure text plus the Measured Value of the database query. Furthermore there are some additional columns filled, like Runtime (which shows how long it took for the data collector to query the key figure), the Database Connection name and the used Schema name. In case there are any problems in connecting the remote database or executing the query, you would see the error message as alert text.

6.7.2 Detail Display call into monitored system

Mark an alert and press push button “Detail Info”. This loads the relevant remote database table entries into an SAP-ABAP-based list viewer. The display might slightly differ from what you see with native database tools or within the UI of the external application. Not all table columns might find a matching ABAP type or might be represented in some other display format.

Please note that you always see the current selection result, even if you have selected an older alert. Reason is that the selection result for the regular key figure calculation is not stored, but rather re-selected during the detail display call.

For performance reasons, a maximum of 10,000 rows is shown.
Press the magnifier glass push button to call up an additional detail popup screen. In addition, you can use the common features of the list viewer, such as sorting, filtering and exporting.
7 Introduction to monitoring external systems with Web services

In order to monitor applications on non-SAP systems via Business Process Monitoring in SAP Solution Manager, SAP provides the possibility collect data for these external applications via Web services. In general, there are two different interaction mechanisms between SAP Solution Manager and an external application (see Picture 33):

The first option is that the external system initiates the data exchange and push data into SAP Solution Manager (right scenario in picture 1). A classical example would be an RFC-call from the external application using the RFC Software Development Kit provided by SAP. Using Web services, the external system would implement a consumer for a Web service provided by SAP. This scenario is realized in the Business Process Monitoring by providing Web service DSWP_BPM_PUSH_WS and monitoring object NAWSPUSH and is described in section “
Monitoring object “Dependent Data Collector for Push WS” (NAWSPUSH)”.

- The second option is that SAP Solution Manager initiates the data exchange and pulls data from the external system (left scenario in picture 1). This scenario can be realized as well and is described in chapter “Monitoring object “Data Collector for Pull WS” (NAWSPULL)”

Picture 33: Interaction with external system via Web services
8 Monitoring object “Dependent Data Collector for Push WS” (NAWSPUSH)

8.1 Purpose
The monitoring object is intended to evaluate data from non-SAP-systems which has been pushed into SAP Solution Manager using Web Service DSWP_BPM_PUSH_WS.

8.2 Technical Prerequisites
The “Dependent Data Collector for Push WS” is delivered with add-on ST-A/PI starting with release 01M. The technical name of the Monitoring Object is NAWSPUSH.

The data collector is executed directly on SAP Solution Manager so the ST-A/PI has to be implemented on the SAP Solution Manager system. The collector evaluates data received by Web Service “DSWP_BPM_PUSH_WS”. This Web Service is delivered as part of ST-SER 20101_1. Both the ST-A/PI and the ST-SER with the correct versions are part of SPS23, so this stack or a higher version has to be implemented.

Inside SAP Solution Manager, you must have a solution created with at least one business process maintained. The business process should contain several business process steps, at least one of which should be executed on the monitored system. The monitored system can be any system (including non-ABAP). Since the ABAP based data collection is executed on SAP Solution Manager, the “monitored system” is not the system on which the data collection runs physically, but the system to which the monitored data refers. You may want to create an own Logical Component for this non-ABAP/non-SAP system (instead of using the Solution Manager’s Logical Component), in order to have the alert records assigned to the system where the actual problem occurred.

8.3 Performance Warning
The Web service has to identify the relevant listeners NAWSPUSH for the received data. This will cause some additional load on SAP Solution Manager, especially if many listeners have been created and a large amount of data is received frequently. Therefore it is in the responsibility of the user to test the impact on the overall system load.

Please consider your setup carefully and follow these golden rules:

- Avoid pushing too many data sets into the Web Service.
- Avoid pushing data into SAP Solution Manager with a high frequency

8.4 Setup Procedure
8.4.1 Example use-case
To demonstrate the setup of the PUSH scenario we choose a fictive non-SAP legacy warehouse management system WMS where the performance and the number of transfer orders is evaluated by some in-house developments once per day. This data should be forwarded to SAP Solution Manager to utilize the BPMon alerting and reporting framework.
8.4.2 Setup in the managed non-SAP system

A consumer for the Web service “DSWP_BPM_PUSH_WS” provided by SAP Solution Manager has to be created in the legacy system. This consumer should either be included in the existing in-house developments, or some new programs have to be evaluated that obtain the evaluation result from an intermediate storage and forward it to the Web service. As the exact implementation depends on the used operating system, programming language etc. no detailed programming guideline can be provided but we would like to refer to the readily available literature\(^1\).

The consumer has to provide at least the intended measured value like the number of transfer orders created and up to two identifiers. These identifiers are used to describe the received data and link it to the relevant monitoring object. They can contain any information you would like.

Good identifiers for the example use case would be:
- Identifier 1: SystemID, e.g. WMS
- Identifier 2: Usages, e.g. PERFORMANCE or NUMBER_DOCUMENTS.

A Web Service Description Language (WSDL)-document describing the complete Web service can be found in the appendix WSDL-Document.

8.4.3 Setup Procedure in SAP Solution Manager

In short:

Create appropriate bindings according to your security needs for the Web service (see SAP Note 1318499)

Set up the BPMon configuration for application monitoring object NAWSPUSH, specifying the following selection criteria:
- Identifiers to link the received data to the monitoring object instance

Detailed instructions to set up the transport binding:

1. Start transaction SOAMANAGER
2. Navigate to Service Administration and choose Single Service Configuration
3. Search for a Service using the internal name DSWP_BPM_PUSH_WS

\(^1\) Some guidance for the utilization of Web Services in standard programming languages can for example be found in

- SAMS Teach Yourself Web Services in 24 hours, SAMS Publishing, 2003
- Java SOA Cookbook, O'Reilly Media; 2009
4. Mark the displayed service line and select *Apply Selection* (see Picture 34)

![Web Service Administration](image)

**Picture 34: Starting the administration for the Web service**

5. Move to *Configurations* and choose *Create Endpoint*.
6. Enter the desired data for communication security, authentication and transport settings (see Picture 35)

![Configuration of the service end point](image)

**Picture 35: Configuration of the service end point**

Detailed instructions to set up the BPMon configuration:


8. Within the BPMon Setup session execute the following one-time activities:
   Node 'Basic Settings' → 'Update Central Application Monitoring Repository': Press Button 'Load Monitors' and select the system ID of the SAP Solution Manager system
itself to load the monitoring objects running locally.

9. Within node 'Business Processes': Select the business process you want to monitor.

10. Within the sub-node for your selected business process: Choose the relevant process steps you want to monitor.

11. Within the node for your selected step: Choose the monitoring type 'Application Monitors: Other area' or 'Application Monitors: Cross Application'.

12. Node ‘Application Monitors ’ choose application monitor ‘Dependent Data Collector for Push WS’ [NAWSPUSH] via search help for field Monitor Name and name your monitoring object with a speaking name, e.g. “Transfer Orders via PUSH WS” if you want to evaluate the number of transfer orders created in the legacy WMS system. This will create a sub-node below 'Application Monitors' with the name of your monitoring object as shown in Picture 36.

![Picture 36: New node for the Monitoring Object NAWSPUSH](image)

13. Node '< monitoring object name >', tab 'Detail Information': Customize the counter by double-clicking on counter ‘001’ which will result in the selection screen shown in Picture 37.

![Picture 37: Available parameters for Object NAWSPUSH](image)

Parameter "Identifier 1" defines the first identifier provided by the external application to Web service DSWP_BPM_PUSH_WS. In the given example you would have to enter “WMS”.

For parameter "Identifier 2", enter the second identifier provided. In the given example, it would be “NUMBER_DOCS" if you want to monitor the number of transfer orders created and “PERFORMANCE” if you want to monitor the performance of the system as provided to the Web service.

No input help is available for these values as they depend on the values transferred from the external system.

27. Node '< monitoring object name >', tab 'Monitoring Schedule': Enter the data for the scheduling of the collector on SAP Solution Manager, i.e. select the relevant weekdays, the data collection period or the fixed start time at which the collector should check whether data has been received by the Web service. In case multiple results are received from the legacy system between two runs of the collector
individual alerts will be created for each measured value.

28. Node '<monitoring object name>' tab 'Key Figures': Choose the key figures you want to use. There are three key figures available as shown in Picture 38.

29. After saving you see a new sub-node for each selected key figure where you can configure additional customizing on key figure level.

- **“Status Keyfigure”**
  You should use this key figure if you want to react to an event independent of the measured value; e.g. if you have a severe failure in the WMS you are not interested in how many failures occurred, the important fact is that a failure occurred. Enter the rating of the intended alert:

  ![Picture 39: Example for key figure “Status Keyfigure”](image)

- **“2-Step-Rating”**
  You should use this key figure if you want to get an alert if the measured value exceeds certain thresholds. This is most common for error monitoring or performance monitoring where you can have too many errors or a response time to high but never too few errors or a performance too good. This key figure could therefore be chosen for the identifiers “WMS” and “PERFORMANCE”. Enter a threshold for a YELLOW alert and a threshold for a RED alert as absolute values:

  ![Picture 40: Example for key figure “2-Step-Rating”](image)

- **“4-Step-Rating”**
  You should use this key figure if you want to get an alert if the measured value has to be within some boundaries of expected values. This is most common for throughput measurements where the number of created documents can exceed your expectations or can be too small. In our example this key figure could therefore be chosen for the identifiers “WMS” and “NUMBER_DOCUMENTS”.
  For each key figure you can enter two sets of thresholds. The thresholds for a
YELLOW alert and for a RED alert have to be entered as absolute values., also supporting a four-step rating with minimum and maximum thresholds pairs:

<table>
<thead>
<tr>
<th>4-Step Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

**Picture 41: Example for key figure “4-Step-Rating”**

30. In node ‘Analysis & Monitoring Tools’, do not select an ABAP-based analysis tool like a transaction code. Instead, use the “Detail Info” functionality, which requires no configuration in this node. For details how to set up this “Detail Info” display refer to the section Detail Display for “Dependent Data Collector for Push WS” (NAWSPUSH) and “Data Collector for Pull WS” (NAWSPULL).

31. Once you have entered all relevant information for the objects you want to monitor, generate the monitoring customizing and activate the monitoring within the Business Process Monitoring session.

**8.5 Result in Monitoring Session**

**8.5.1 Alert Display in Solution Manager**

**Alert display for key figure “4-Step-Rating”**:  

The column Alert Message shows the alert text. You see the date and time of the data collection in SAP Solution Manager in columns Date and Time. In addition, the web service has transferred the measured value (6 in the first alert, 0 in the second one), the date used for data collection in the external system in column Info1, and the name of the observed object “Transfer Orders” in column Object1. No information has been returned in Object2 and Object3.

**Picture 42: Example for an alert caused by pushing data into BPMon**

**8.5.2 Detail Display call into monitored system**

Mark an alert and press push button “Detail Info”. This will trigger a call to a BAdI where you can implement the selection of the data to be shown and how it should be presented. For details refer to section Detail Display for “Dependent Data Collector for Push WS” (NAWSPUSH) and “Data Collector for Pull WS” (NAWSPULL).
9 Monitoring object “Data Collector for Pull WS” (NAWSPULL)

9.1 Purpose
The monitoring object is intended to pull monitoring data from non-SAP-systems via a Web Service interface.

9.2 Technical Prerequisites
The monitoring functionality “Data Collector for Pull WS” is delivered with the add-on ST-A/PI starting with release 01M. The technical name of the Monitoring Object is NAWSPULL.

The data collector is executed directly on SAP Solution Manager so ST-A/PI 01M or higher has to be implemented on SAP Solution Manager. The monitoring object receives data via a BAdI-implementation. The relevant BAdI BADI_DSWP_BPM_PULL_WS is delivered as part of ST-Ser 20101_1. Both the ST-A/PI and the ST-SER with the correct versions are part of SPS23, so this stack or a higher version has to be implemented.

In addition, SAP Notes 1478242 and 1475174 or the related support packages are recommended.

Inside SAP Solution Manager, you must have a solution created with at least one business process maintained. The business process should contain several business process steps, at least one of which should be executed on the monitored system. The monitored system can be any system (including non-ABAP). Since the ABAP based data collection is executed on SAP Solution Manager, the “monitored system” is not the system on which the data collection runs physically, but the system to which the monitored data refers. You may want to create an own Logical Component for this non-ABAP/non-SAP system (instead of using the Solution Manager’s Logical Component), in order to have the alert records assigned to the system where the actual problem occurred.

9.3 Performance Warning
The data collector can query any information object that is accessible via the exposed web service in the managed system. Therefore it is in the responsibility of the user to avoid frequent invocations of Web Service performing selections on huge tables returning large amounts of data. Also, it is in the responsibility of the Web Service developer to ensure a good performance.

Please consider your setup carefully and follow these golden rules:
- Avoid creating web services to query large data sets without a good access path.
- Avoid running the data collection with a high monitoring frequency

If you are in doubt, try to simulate the query in the managed system to get an idea about the possible amount of data evaluated. Furthermore, monitor the data collection runtime.

9.4 Setup Procedure

9.4.1 Example use-case
To demonstrate the setup of the Pull Web service we choose a simple fictive non-SAP warehouse management system where information about the transfer orders is stored in a simple table (see Picture 43)
The table will log the transport order number together with the corresponding material movement information like quantities, storage location and execution date. Now we want to measure the throughput of transfer orders using this table.

### 9.4.2 Setup in the managed system

First you have to create a Web Service that receives the relevant identifiers from SAP Solution Manager, determines from these identifiers that it should query the throughput for which day. The actual details depend on the programming environment available on the managed system like programming language, web server used etc. and cannot be covered here in detail. Instead, we’ll provide a typical example where we want to create the Web Service using Visual Studio.

You have to create a new web site that should be invoked by SAP Solution Manager. The coding has to declare the appropriate communication structures to be used in the data exchange with the Solution Manager system. Mainly we need structures to receive the identifiers and to return the appropriate measured value and alert context. If you would use VB.NET, the coding could look like Listing 1:

```vbnet
Public Structure BPM_TRANSFER
    Dim Ident1 As String
    Dim Ident2 As String
    Dim MeasValue As Integer
    Dim Object1 As String
    Dim Object2 As String
    Dim Object3 As String
    Dim Info1 As String
    Dim Info2 As String
    Dim Info3 As String
    Dim Info4 As String
End Structure

Public Structure BPM_RETURN
    Dim Type As Char
    Dim Message As String
End Structure

Public Structure BPM_PULL
    Dim Ident1 As String
    Dim Ident2 As String
End Structure

Public Structure BPM_PULL_RESPONSE
    Dim DATA As BPM_TRANSFER
    Dim RETURN1 As BPM_RETURN
End Structure
```

Listing 1: Example for definition of the communication structures
Here, the public structure BPM_PULL_RESPONSE would return the actual alert data and potential error messages as defined by BPM_TRANSFER and BPM_RETURN while structure BPM_PULL is intended to receive the trigger information as set up in the BPMon Setup session, e.g. the two identifiers maintained in that session.

Note:
It is recommended that you use the same field names in the structure definitions in your foreign application as used in the parameter definitions for the BAdl-interface in SAP Solution Manager as you can transfer the data in the BADI-implementation using MOVE-CORRESPONDING.

To be available as Web Service the project has to use the appropriate .NET-classes and to declare a public method that can be invoked by SAP Solution Manager. In VB.NET the coding could look like Listing 2:

```vbnet
Imports System.Web
Imports System.Web.Services
<WebService(Namespace:="Your Namespace")> _
<WebServiceBinding(ConformsTo:=WsIProfiles.BasicProfile1_1)> _
Public Class BPM_PULL_WS

    <WebMethod()> _
    Public Function Example_WS(ByVal Input As BPM_PULL, ByRef Response As BPM_PULL_RESPONSE) As String
        ...
        <Your code to evaluate Input (e.g. the two identifiers provided), trigger the appropriate selections and return the result in Response.>
        ...
    End Function

End Class
```

Listing 2: Example for a non-ABAP Web Service interface to SAP Solution Manager

Once you have deployed the project it is available in your Web Server and you can implement the invocation from SAP Solution Manager via the provided WSDL-document.

### 9.4.3 Setup Procedure in SAP Solution Manager

In short:
For application monitoring, implement a consumer proxy for the Web service created in section 9.4.2 in BADI BADI_DSWP_BPM_PULL_WS, set the appropriate filters to select the correct implementation, set up the monitoring object NAWS_PULL using the correct identifiers and maintain the intended scheduling to trigger the execution of the Web Service in the managed system.

Detailed instructions:

2 A good tutorial on web service consumption can be found in SDN under http://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/media/uuid/20eb3174-41ab-2a10-a383-907f9f60eed3
Start transaction SE80 and create a Consumer Proxy based on the WSDL of your Web Service. Follow the wizard to create the proxy class (see Picture 44). At the end you have to generate the class.

![Picture 44: Creating a consumer proxy](image)

Once you have created a consumer proxy you need to create the needed HTTP-Connection between SAP Solution Manager and your service provider. To do this, start transaction SOAMANAGER, switch to Application and Scenario Communication and select Single Service Administration. Search for your proxy, mark it in the result list and press Apply Selection. Then move to tab Configurations and press Create Logical Port. You can generate the logical port based on your WSDL or create it manually (see Picture 45).
Now that the logical port has been created you can use the consumer in your BAdI implementation. Create the BAdI implementation with corresponding filter criteria using transaction SE18, drag and drop the consumer proxy to the relevant coding of method CALL_PULL_WS and adopt the coding. A typical example can be found in Listing 3.

```abap
method EX_IF_CALL_WS~CALL_PULL_WS.
  DATA: IO_WS_PULL TYPE REF TO <Generated Proxy Class for your PULL web service>.
  DATA: ls_data TYPE DSWF_BPM_WS_TRANSFER.
  FIELD-SYMBOLS: <ls_item> TYPE LINE OF DSWF_BPM_WS_PULL_RESPONSE-DATA-ITEM.
  DATA: ls_return TYPE LINE OF DSWF_BPM_TT_WS_RETURN.
  DATA: o_system_fault TYPE REF TO CX_AI_SYSTEM_FAULT.
  DATA: o_appl_fault TYPE REF TO CX_AI_APPLICATION_FAULT.
  TRY.
    CREATE OBJECT IO_WS_PULL
    * EXPORTING
    * LOGICAL_PORT_NAME = .
    CATCH CX_AI_SYSTEM_FAULT .
  ENTDYN.
  data: LS_OUTPUT type <Generated Output Structure for your pull web service>.
  data: LS_INPUT type <Generated Input Structure for your pull web service>.
  ls_input-input-ident1 = iv_ident1.
  ls_input-input-ident2 = iv_ident2.
```
TRY.
   CALL METHOD IO_WS_PULL->DSWP_BPM_PULL
   EXPORTING
       INPUT  = LS_INPUT
   IMPORTING
       OUTPUT = LS_OUTPUT.
CATCH CX_AI_SYSTEM_FAULT INTO o_system_fault.
   ls_return-type = 'E'.
   ls_return-message = o_system_fault->GET_TEXT( ).
   APPEND ls_return TO et_return.
CATCH CX_AI_APPLICATION_FAULT INTO o_appl_fault.
   ls_return-type = 'I'.
   ls_return-message = o_appl_fault->GET_TEXT( ).
   APPEND ls_return TO et_return.
ENDTRY.

* LOOP AT ls_output-response-data assigning <ls_item>.
  MOVE-CORRESPONDING ls_output-response-data TO ls_data.
  APPEND ls_data TO et_data.
* ENDLOOP.

   MOVE-CORRESPONDING ls_output-response-return1 TO ls_return.
   APPEND ls_return TO et_return.
endmethod.

Listing 3: Example for a consumption of a web service

To trigger the execution of the Web Service interface you have to set up Business Process Monitoring for monitoring object NAWSPULL.


2. Within the BPMon Setup session execute the following one-time activities::
   Node 'Basic Settings' → 'Update Central Application Monitoring Repository': Press Button 'Load Monitors' and select the System ID of the SAP Solution Manager system itself to load the monitoring objects running locally.

3. Within the node 'Business Processes': Select the business process you want to monitor.

4. Within the sub-node for your selected business process: Choose the relevant process steps you want to monitor.

5. Within the node for your selected step: Choose the monitoring type 'Application Monitors: Other area' or 'Application Monitors: Cross Application’.

6. Node ‘Application Monitors’: Choose application monitor ‘Dependent Data Collector for Pull WS’ [NAWSPULL] via search help and name your monitoring object with a speaking name, e.g “Transfer Orders via PULL WS” if you want to evaluate the number of transfer orders created in the legacy WMS system. This will create a sub-node below
'Application Monitors' with the name of your monitoring object.

7. Node '<monitoring object name>', tab 'Detail Information': Customize the first counter line by double-clicking on counter ‘001’.

   ![Picture 46: Available parameters for Object NAWSPULL](image)

   Parameter "Identifier 1" defines the first identifier needed for the correct data selection in the external system. In the given example you would have to enter “WMS”.

   For parameter "Identifier 2", enter the second identifier you want to provide to the external system. In the given example, it could be “NUMBER_DOCUMENTS” if you want to monitor the number of transfer orders created.

   No input help is available for these values as they depend on the values evaluated in your Web Service coding on the external system.

8. Node '<monitoring object name>', tab 'Monitoring Schedule': Enter the data for the scheduling of the collector in the Solution Manager, i.e. select the relevant weekdays, the data collection period or the fixed start time. This frequency determines how often the Web Service in the non-SAP-System is called.

9. Node '<monitoring object name>', tab 'Key Figures': Choose the key figures you want to use. There are three key figures available as shown in Picture 38.

   ![Picture 47: Available key figures for the PULL-interaction](image)

10. After saving you see a new sub-node for each selected key figure where you can configure additional customizing on key figure level.

    - **“Status Keyfigure”**
      
      You should use this key figure if you want to react to an event independent of the measured value; e.g. if you have a severe failure in the WMS you are not interested in many failures occurred, the important fact is that a failure occurred.
Enter the rating of the intended alert:

**“2-Step-Rating”**

You should use this key figure if you want to get an alert if the measured value exceeds certain thresholds. This is most common for error monitoring or performance monitoring where you can have too many errors or a response time to high but never too few errors or a performance too good. This key figure could therefore be chosen for the identifiers “WMS” and “PERFORMANCE”. Enter a threshold for a YELLOW alert and a threshold for a RED alert as absolute values:

**“4-Step-Rating”**

You should use this key figure if you want to get an alert if the measured value has to be within some boundaries of expected values. This is most common for throughput measurements where the number of created documents can exceed your expectations or can be too small. In our example this key figure could therefore be chosen for the identifiers “WMS” and “NUMBER_DOCUMENTS”.

For each key figure you can enter a threshold for a YELLOW alert and a threshold for a RED alert as an absolute value, also supporting a four-step rating with minimum and maximum thresholds pairs:

11. In node ‘Analysis & Monitoring Tools’, do not select an ABAP-based analysis tool like a transaction code. Instead, use the “Detail Info” functionality, which requires no configuration in this node. For details how to set up this “Detail Info” display refer to the section Detail Display for “Dependent Data Collector for Push WS” (NAWSPUSH) and “Data Collector for Pull WS” (NAWSPULL).

12. Once you have entered all relevant information for the objects you want to monitor, generate the monitoring customizing and activate the monitoring within the ‘Business Process Monitoring’ session.
9.5 Result in Monitoring Session

9.5.1 Alert Display in Solution Manager

![Picture 51: Example for an alert caused by pulling data from an external system]

The column Alert Message shows the alert text which is currently only a standard alert text. You can see the date and time of the data collection in SAP Solution Manager in columns Date and Time. In addition, the web service has transferred the measured value 6 and the date used for data selection which has been placed into field Info1. Fields Object1 to Object3 have not been transferred.

9.5.2 Detail Display call into monitored system

Mark an alert and press push button “Detail Info”. This will trigger a call to a BAdI where you can implement the selection of the data to be shown and how it should be presented. For details refer to section 1.

10 Detail Display for “Dependent Data Collector for Push WS” (NAWSPUSH) and “Data Collector for Pull WS” (NAWSPULL)

10.1 Purpose

The Detail Display is intended to show the relevant data causing the selected alert together with some context information in order to start basic root cause investigations if desired. To do this two different means of displaying the data can be implemented in a BAdI which will both be described in detail in the next sections.

10.2 Technical Prerequisites

The main routines and BAdIs for the Detail Display of both Web Services are executed directly on SAP Solution Manager, so ST-A/PI01M or higher has to be implemented on SAP Solution Manager. The monitoring object receives data via a BADI-implementation. The relevant BADI is delivered as part of SPS23 so this stack or higher has to be implemented.

In addition, SAP Notes 1478242 and 1475174 or the related support packages are recommended to make use of the complete functional scope.

Inside SAP Solution Manager, you must have set up at least one monitoring instance for monitoring object NAWSPULL or NAWSPUSH.

10.3 Performance Warning

The Detail Display has to handle any information that is returned. Therefore, it is in the responsibility of the developer to optimize the access to huge tables returning large amounts
of data. Keep in mind that the data to be displayed has also to be transferred via the network and (depending on the chosen display method) be realized as internal tables in SAP Solution Manager.

10.4 Setup Procedure

10.4.1 Example use-case

To demonstrate the setup of the Detail Display we choose the same example used to measure the throughput of transfer orders, but this time we want to return not only the throughput number, but the individual transfer orders constituting the throughput. The example will be based on the same model used in chapter 9.

10.4.2 Setup in the managed system

There are two options you can use and which govern the detailed implementation on the non-SAP system. You could create a Web service that returns the selected data to SAP Solution Manager where it will be displayed in an ALV or you could display the data directly in a Web page on the external system. The latter method is especially useful if the external system provides a Web Interface as you could trigger further actions from the Web page via links if needed. We will first describe the setup of a Web page followed by the detailed setup for the Web service methodology.

To utilize a Web page you have to create a Web page that can received context parameters if needed, perform the needed data selections (for example via ASP.Net or other scripting languages) and then display the required data.

If you want to display the data in SAP Solution Manager you have to start the consumption of a Web service in BAdI BADI_DSWP_BPM_WS_DETAILS using coding similar to Listing 3. The Web Service has to return the data in an XML-like format as a STRING or XSTRING using tags shown in Table 1.

<table>
<thead>
<tr>
<th>TAG</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;TABLE&gt;…&lt;/TABLE&gt;</td>
<td>This tag starts and ends the transferred table data</td>
</tr>
<tr>
<td>&lt;HEADER&gt;…&lt;/HEADER&gt;</td>
<td>This tag encompasses the transferred header information.</td>
</tr>
<tr>
<td>&lt;LINE&gt;…&lt;/LINE&gt;</td>
<td>This tag encompasses each row of data</td>
</tr>
<tr>
<td>&lt;FD&gt;…&lt;/FD&gt;</td>
<td>This tag contains the actual data on line level and the column titles on header level. Each column has to be represented by a tag combination.</td>
</tr>
</tbody>
</table>

Table 1: Tags used to transfer detailed data

An example can be found in Picture 53.

10.4.3 Setup Procedure in SAP Solution Manager

Detailed instructions:

The trigger to the external system is implemented in BAdI BADI_DSWP_BPM_WS_DETAILS providing interface EX_IF_DISPLAY_WS.
You have to create an implementation for this BADI using transaction SE18 where you call the external Web Service in method DISPLAYDETAILS. The actual coding is similar to Listing 3, you just have to replace the proxy and generated DDIC-types to the PULL Web service with the proxy and DDIC-types of your Detail Display Web service implemented in the external system. The available filters for the BAdI-implementation are shown in Table 2.

<table>
<thead>
<tr>
<th>Filter</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILTERDETAIL_WS_IDENT1</td>
<td>Filter the BAdI-implementation according to the first identifier</td>
</tr>
<tr>
<td>FILTERDETAIL_WS_IDENT2</td>
<td>Filter the BAdI-implementation according to the second identifier</td>
</tr>
<tr>
<td>FILTERDETAIL_WS_TYPE</td>
<td>Filter BAdI-implementation for detail display according to PULL/PUSH WebService. Allowed filter values are PULL and PUSH which are set by the coding.</td>
</tr>
<tr>
<td>FILTERDETAIL_WS_SID</td>
<td>Filter the BAdI-implementation according to the system ID of the system you monitor</td>
</tr>
</tbody>
</table>

Table 2: Available filters for BAdI BADI_DSWP_BPMDETAILS

The data is transferred from the BAdI-implementation to the calling program using parameter ET_DATA which has to contain the table to be displayed as an XSTRING. If your web service cannot send the data in the correct encoding you can transmit it as a STRING or character field and convert it to a XSTRING within the BAdI-implementation using a call to function module “SCMS_STRING_TO_XSTRING” as shown in Listing 4

```
DATA: lv_string TYPE STRING.
DATA: lv_xstring TYPE XSTRING.
DATA: ls_data TYPE LINE OF DSWP_BPM TT WS DETAIL
  lv_string = <data returned from WebService>.
  CALL FUNCTION 'SCMS_STRING_TO_XSTRING'
    EXPORTING
      TEXT  = lv_string
    IMPORTING
      BUFFER = lv_xstring.
  ls_data-data = lv_xstring.
  APPEND ls_data TO et_data.
```

Listing 4: Conversion from STRING to XSTRING-format
10.5 Result in Monitoring Session

10.5.1 Detail Display for a Web Page
The Detail Display will simply be the Web Page you have called via function module CALL_BROWSER. For our given example the page could look like Picture 52.

![Detailed Display for Transfer Orders - Windows Internet Explorer](image)

Picture 52: Detailed Display for the throughput measurement via Web page

10.5.2 Detail Display call to receive the data
The Detail Display will simply be the table data you have returned to SAP Solution Manager based on the provided XML-String. For our given example the returned string could be similar to the one depicted in Picture 53.
Picture 53: The XML-String as returned from the external Web service

This data would be converted into an internal table and displayed as shown in Picture 54

<table>
<thead>
<tr>
<th>TNUM</th>
<th>MATNR</th>
<th>QTY</th>
<th>CREDAT</th>
<th>STORAGE_FROM</th>
<th>STORAGE_TO</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>12</td>
<td>1</td>
<td>15.06.2010</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>64</td>
<td>12</td>
<td>1</td>
<td>15.06.2010</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>65</td>
<td>12</td>
<td>1</td>
<td>15.08.2010</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>66</td>
<td>1254</td>
<td>2</td>
<td>15.08.2010</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>67</td>
<td>12</td>
<td>1</td>
<td>15.08.2010</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>68</td>
<td>13</td>
<td>1</td>
<td>215.08.2010</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Picture 54: Detail Display based on data returned by a Web Service
11 The monitoring object “Generic Infocube/DSO Reader for remote BW” (NABWICNX)

11.1 Purpose
The “Generic Infocube/DSO Reader for remote BW” provides the possibility to run generic queries to obtain the value of BW key figures in any BW information object using external BI read interface (RSDRI). It supports a flexible filtering based on select-options on up to three characteristics and allows an aggregation of the selected data. This monitoring object is therefore a substitute or workaround for areas, where no specific application monitor is available. Furthermore this monitor can help to avoid implementing the application monitoring customer-exit, as long as the collector result can be calculated by a simple query statement on one single information object. This application monitor is part of the Business Process Monitoring framework within SAP Solution Manager.

11.2 Technical Prerequisites
The Generic Infocube/DSO Reader for remote BW is delivered with the add-on ST-A/PI starting with release 01M. The technical name of the Monitoring Object is NABWICNX.

The data collector is executed directly on SAP Solution Manager for this purpose.

To monitor an external system via data collected into a BW system a RFC connection has to be established to the BW system. If the SAP Solution Manager’s integrated BW is used in the same client as the Business Process Monitoring, RFC-Destination “NONE” may be used.

Inside the Solution Manager, you must have a solution created with at least one business process maintained. The business process should have several business process steps. The monitored system can be any system, but the actual execution of the ABAP-based data collector is on the Solution Manager and SAP BW itself. However, the actual monitored system is not the system where the data collector runs physically, but the system to which the data in the information object refers to. So you may want to create an own Logical Component for this non-ABAP/non-SAP system (instead of using the Solution Manager’s Logical Component), in order to have a unique assignment of the generated alert records.

11.3 Limitations
Search helps for defining filters are fed by selecting distinct field values from the corresponding database tables where the information is stored physically, which might be a performance-intensive query. Within the BPMon Setup Session, all filter criteria are stored as character fields, so explicit conversions may have to be considered, e.g. for number types or dates.

The data collector accesses the BW system to query the data using a predefined interface and will map the native data types into SAP dictionary data types and ABAP data types, for example to allow a detail display with the list viewer. However, it cannot be guaranteed that all native data types can be properly matched into a suitable ABAP data type. Therefore the output of certain data might look different compared to the user interface of native database tools or other non-ABAP applications.
11.4 Performance Warning

The data collector can query any information object that is accessible by the login user maintained in the RFC-destination. Therefore it is in the responsibility of the user to avoid selections on huge tables returning large amounts of data. Keep in mind that also the necessary aggregations to calculate a single measured value can consume a lot of performance and memory resources especially when no or only not very selective filters are applied!

Using this application monitor without care could cause severe performance degradation on the BW system and the SAP Solution Manager.

Please consider your setup carefully and follow these golden rules:

- Avoid using this application monitor on large information objects returning many data sets
- Avoid running the data collection with a high monitoring frequency
- Avoid complex filtering, especially with patterns, ranges and exclusions.

If you are in doubt, try to simulate the query in the BW-system to get an idea about the possible amount of data being returned. Furthermore, monitor the data collection runtime which is logged into a special column of the alert list (see chapter “Alert Display”).

11.5 Setup Procedure

11.5.1 Example use-case

To demonstrate the setup of the Table Entry Counter we choose a simple information object PERF_DATA of a fictive non-SAP warehouse management system with the following structure:

<table>
<thead>
<tr>
<th>Metric Name THOS</th>
<th>Object Name THOS</th>
<th>THOS_MENA</th>
<th>CHAR</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric Type THOS</td>
<td></td>
<td>THOS_METY</td>
<td>CHAR</td>
<td>60</td>
</tr>
<tr>
<td>Object Name THOS</td>
<td></td>
<td>THOS_OBN</td>
<td>CHAR</td>
<td>60</td>
</tr>
<tr>
<td>Object Type THOS</td>
<td></td>
<td>THOS_OBYT</td>
<td>CHAR</td>
<td>60</td>
</tr>
<tr>
<td>Time Stamp THOS</td>
<td></td>
<td>THOS_TIKH</td>
<td>NUMC</td>
<td>12</td>
</tr>
<tr>
<td>SystemID THOS</td>
<td></td>
<td>THOS_SID</td>
<td>CHAR</td>
<td>08</td>
</tr>
</tbody>
</table>

| Measured Value THOS | THOS_MEAS | INT4 | 04 |

Picture 55: The InfoObject used for the example

The information object contains some sample performance and error data for the last days and hours:

<table>
<thead>
<tr>
<th>Metric Name THOS</th>
<th>Metric Type THOS</th>
<th>Object Name THOS</th>
<th>Object Type THOS</th>
<th>Time Stamp THOS</th>
<th>SystemID THOS</th>
<th>Measured Value THOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERRORS</td>
<td>LOGGING</td>
<td>TRANSFERORDER</td>
<td>STEP_NAME</td>
<td>201004121300 WMS</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>ERRORS</td>
<td>LOGGING</td>
<td>TRANSFERORDER</td>
<td>STEP_NAME</td>
<td>201004121310 WMS</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ERRORS</td>
<td>LOGGING</td>
<td>TRANSFERORDER</td>
<td>STEP_NAME</td>
<td>201004121410 WMS</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>RUN_TIME</td>
<td>PERFORMANCE</td>
<td>TRANSFERORDER</td>
<td>STEP_NAME</td>
<td>201004121300 WMS</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>RUN_TIME</td>
<td>PERFORMANCE</td>
<td>TRANSFERORDER</td>
<td>STEP_NAME</td>
<td>201004121310 WMS</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>RUN_TIME</td>
<td>PERFORMANCE</td>
<td>TRANSFERORDER</td>
<td>STEP_NAME</td>
<td>201004121410 WMS</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

Picture 56: Content of the example’s InfoObject

11.5.2 Simulate data collector result in monitored system

You can use transaction RSA1 in the BW-system to simulate the query and to clarify the selection parameters. Start the transaction and select the relevant InfoObject. Right-click it
and choose “Display Data”. This will show a selection screen where you may enter the needed filter conditions.

### Picture 57: Selecting data for display in transaction RSA1

Using the button “Fld Selectn for Output” to choose fields you want to use in BPM’s detail display functionality.

### Picture 58: Defining output fields in transaction RSA1
Execute the query to verify the result:

<table>
<thead>
<tr>
<th>THOS_MEND</th>
<th>THOS_METY</th>
<th>THOS_OBTY</th>
<th>THOS_SID</th>
<th>THOS_TIHM</th>
<th>THOS_MEAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERRORS</td>
<td>LOGGING</td>
<td>TRANSERORDER</td>
<td>STEP_NAME</td>
<td>WMS</td>
<td>201004121300</td>
</tr>
<tr>
<td>ERRORS</td>
<td>LOGGING</td>
<td>TRANSERORDER</td>
<td>STEP_NAME</td>
<td>WMS</td>
<td>201004121310</td>
</tr>
<tr>
<td>ERRORS</td>
<td>LOGGING</td>
<td>TRANSERORDER</td>
<td>STEP_NAME</td>
<td>WMS</td>
<td>201004121410</td>
</tr>
</tbody>
</table>

**Picture 59: Data display in transaction RSA1**

Now that we have proved that the table content can be accessed, we want to set up the following KPIs for application monitoring:
- Average number of errors for transfer orders
- Maximum response time

### 11.5.3 Setup Procedure in SAP Solution Manager

In short: For application monitoring, choose the monitoring object NABWICNX and specify the following selection criteria:
- RFC connection name (optional, only needed if a connection different to the one maintained in the system data should be used)
- Type of InfoObject
- Name of InfoObject
- Characteristic names for filtering and select-options for actual filter values
  - this is available three times and will be used to generate a filter for RSDRI
- Names of characteristics to be shown in the detail display (Attention: characteristics used as filter will be shown by default and should not be entered again)

**Detailed instructions:**

1. Start the BPM setup session via the work center “Business Process Operation” by calling the SAP Solution Manager (trx DSWP or SOLUTION_MANAGER).
2. Choose your solution.
3. Go to 'Operation Setup' and navigate to 'Solution Monitoring' => 'Business Process Monitoring'.
4. One-time action only:
   Check 'Basic Settings' => 'Update Central Application Monitoring Repository': Press Button 'Load Monitors' and select the System ID where the ABAP-part of the data collector shall run (which should be the System ID of Solution Manager itself).
5. Within the check 'Business Processes': Select a business process for application monitoring.
6. Within the check for your chosen business process: Choose process steps to monitor.
7. Within the check for your chosen step: Choose type of monitor, here 'Application Monitors: Other area' or 'Application Monitors: Cross Application'.
8. Choose application monitor ‘Generic Infocube/DSO Reader for remote BW’ [NABWICNX] via search help and name your monitoring object with a speaking name. This will create a sub-tree below 'Application Monitors' with the name of your monitoring object.

![Picture 60: Selecting the monitoring object NABWICNX](image)

9. Check '<your monitoring object>' tab strip 'Detail Information': Customize the first counter line by double-clicking on counter ‘001’ which will display the selection screen shown in Picture 61.

![Picture 61: Selection screen for NABWICNX](image)

The first parameter "RFC Connection to BW" defines the target server/client (using a normal RFC connection) to access the BW. The preferred way to maintain this attribute is inside the System Landscape definition (transaction SMSY/LMDB). The business process step refers to a logical component, which itself points to a system. You should enter the RFC connection name into the free attribute called "RFC Dest for BPMon BW-Alerting" at the maintenance screen of this database (see Appendix 2.1 for details). Here you have a central place to store this technical information, which would then be the default for all application monitors for this system using a remote BW connection. If the default should be used it is not necessary to maintain the monitoring object parameter "RFC Connection to BW". If you however do so, this will override the SMSY/LMDB attributes and the data collection and F4-helps will use your entered value instead.

As the second and third parameter, enter the type and name of the InfoObject into parameters "Type of infoobject" and “BW Infocube or DSO”. The search help loads the available names from the remote BW system. For performance reasons, make sure to enter a pattern (e.g. APP* for all tables prefixed by APP) to restrict the selection. The type and name are the only mandatory parameters, whereas the following filters are optional.

The optional fourth to eighth parameters called "FILTERX" and “Value for FilterX” where X equals 1, 2, or 3, allow restricting the data selection. Enter the names of characteristics and corresponding values, which obviously must be a valid characteristic of the selected InfoObject. A search help dialog is also shown here for the characteristics,
which shows all fields of the selected InfoObject based on the remote dictionary and BW naming conventions.

Currently no input help is available for the values. Please note to enter all values in the same format as stored in the InfoObject, for example a date 2008/09/24 might need to be entered as 20080924, or timestamps are usually entered in a 24-hour format without separators. Please see the appendix on how to implement a dynamic date filtering, using relative dates instead of absolute values. Repeat these steps for further filters if required.

The scope of the detail display screen is maintained by the ninth parameter. Instead of all characteristics and key figures only the specified column names will be loaded from the remote InfoObject, to allow hiding some technical fields.

![Selecting characteristics for detail display](image)

**Picture 62: Selecting characteristics for detail display**

Example: The KPIs regarding the number of errors for transfer orders would be defined by the number stored in the key figure “Measured Value” of the InfoObject using the following filter:

Filter1: THOS_MENA with value “ERRORS”
Filter2: THOS_OBNA with value “TRANSFERORDERS”.

![Defining a filter for NABWICNX](image)

**Picture 63: Defining a filter for NABWICNX**

10. Check `<your monitoring object>` tab strip ‘Monitoring Schedule’: Enter the data for the scheduling of the collector job in the Solution Manager, i.e. weekdays, collection period or fixed start time.

11. Check `<your monitoring object>` tab strip ‘Key Figures’: Choose the key figure “Aggregate value of Key Figures” which is actually the only key figure currently available.
After marking the key figure and saving you see a new sub-check where you can do additional customizing in the counter on key figure level.

- “Aggregate value of Key Figures” (with or without aggregate and with given filter criteria)
  
  a. At key figure level, you have to enter the name of the InfoObject’s key figure for which you want to read the value ("First Key Figure").
  
  b. Sometimes it is important not only to have the value of one key figure but to have the relationship between two key figures (for example the number of errors per time interval). In this case you can enter a second key figure. For details, how this is evaluated see the description for the key figure’s parameter "Aggregate Function".
  
  c. In addition you have to enter an "Aggregate Function". For this use case it is intended to select multiple rows, which then return one single integer value due to aggregation - for example MAX(ERRORS) would return the highest number of errors from the InfoObject. Please use the value help to select from possible aggregate functions:

  i. MIN: Minimum value
     The measured value will be the smallest value returned from the InfoObject.
     If one key figure is requested the aggregation MIN returns the smallest value of the key figure according to the filter criteria. In the given example this would be the value ‘0’.
     If two key figures are requested it would be the smallest ratio between the two key figures whereas the ratio is calculated per data row.
  
  ii. MAX: Maximum value
     The measured value will be the largest value returned from the InfoObject.
     If one key figure is requested the aggregation MAX returns the biggest value of the key figure according to the filter criteria. In the given example this would be the value ‘10’.
     If two key figures are requested it would be the largest ratio between the two key figures whereas the ratio is calculated per row.
  
  iii. AVG: Average value
     The measured value will be the average of all values returned from the InfoObject.
     If one key figure is requested it would be the mean value of all values according to the selection criteria. In the given example the value would be \((10 + 0 +5)/3 = 5\).
     If two key figures are requested the average value would be calculated by summing up key figure 1 and key figure 2 first and then dividing the two summarized values.
  
  iv. SUM: Sum of values
     The measured value will be the sum of all values returned (e.g. the total number of errors)
     This aggregation is not implemented if two key figures are selected as the result would be identical to the average calculation.
  
  v. LST: last observed value
     This will be the last value stored in the InfoObject. To use this key figure you have to define a filter on a time characteristic with a
relative date specification (see appendix 2.2).
In the given example it would be the value 5 as this value belongs to the latest time stamp.
If two key figures are selected it would be the ratio of the two key figures in the row with the newest time stamp.

d. You can customize an "Alternative Alert Text", using certain placeholder variables.
The standard key figure alert texts return a rather unspecific alert message - “Green rating for <InfoObject name>” (or "Yellow rating" or "Red rating") This is automatically proposed if you leave the parameter “Alternative Alerttext” empty.
You can use any free text together with the following placeholder to overwrite the standard alert text with your own alert text:
- &1 for the name of the InfoObject
Example for our use-case in key figure:
“Errors for transfer orders measured via &1”
e. The parameter "Detailcall Headertext" allows to store a screen title for the detail display, to explain the user what kind of output is shown.

![Image of counter 001 settings]

- Enter a threshold for a YELLOW alert and a threshold for a RED alert as an absolute value.

![Image of aggregate value of key figures and parameter value ranges]

12. At node 'Analysis & Monitoring Tools', do **not** select an ABAP-based analysis tool like a transaction code. Instead, inside the monitoring session you will have a button called “Detail Info” which will display the relevant table entries inside a list viewer according to the setup filters.

13. Once you have entered all relevant information for the monitoring objects you want to monitor, generate the monitoring customizing and activate the monitoring within the 'Business Process Monitoring' session.
11.6 Result in Monitoring Session

11.6.1 Alert Display in Solution Manager

An example for an alert display of NABWICNX’s key figure is shown in Picture 66.

![Alert display for key figure “Aggregate value of Key Figures”](image)

The column alert message shows the alert text, which in this case is the standard text.

11.6.2 Detail Display call into monitored system

Mark an alert and press push button “Detail Info”. This loads the relevant remote InfoObject entries into an SAP-ABAP-based list viewer. The display might slightly differ from what you see with native BW tools or within the UI of the external application. Some table columns might be represented in some other display format.

Please note that you always see the current selection result, even if you have selected an older alert. The reason is that the selection result for the regular key figure calculation is not stored, but rather re-selected during the detail display call.

![Detail information for key figure “Aggregate value of Key Figures”](image)

The detailed info for this key figure just displays those rows from the InfoObject with the affected entries (applying the same filtering as the data collector). You can also see a header text as defined in the corresponding setup parameter.
12 Appendix I: Technical Details

12.1 Remote Database Connectivity

Introduction
In contrast to the “classic” application monitoring, where the key figures are calculated based on local application data, the monitoring of non-ABAP applications or even non-SAP applications needs to query “remote” (external) databases to collect key figure data. To monitor a remote database using ADBC (ABAP Database Connectivity), the corresponding secondary database connection must be maintained with transaction DBACOCKPIT. The old method of using transaction DBCO is still supported but less user-friendly.

From a technical point of view, you need an installation of an appropriate Database Client and Database Shared Library (DBSL), for each type of SAP-supported DBMS, if not already installed on the system you run the data collector on. For example, if the Solution Manager runs on a MaxDB database, and the data collector wants to connect to some external application using also a MaxDB database, there is no additional installation necessary. However, if this external application runs on MS-SQL, you would need to install an additional DBSL and DB-Client from Microsoft.

Overview
To explain the technical system landscape for a remote DB access, please see the overview picture below.

- On the left-hand side, there is the standard database access for a “classic” BPMon data collector, running on the ABAP-stack of the monitored system. The data collector is an ABAP program, using the database-independent ABAP Open-SQL syntax. Within the SAP Kernel and SAP Database Interface, this is anyhow translated into the database-dependent native SQL syntax, but uses the default DBSL and DB Client for the local default connection to the Database Server with the SAP ABAP application data.

- On the right-hand side, there is a BPMon Data Collector for Remote DB Queries. It is also an ABAP program and therefore needs an ABAP-stack, typically the SAP Solution Manager. It directly works with native SQL syntax, keeping as close as possible to the ANSI SQL standard. If the Remote DB is of the same RDBMS type as the local DB, the connection can be established using the present default DBSL and DB Client. For other vendor’s RDBMS, the connection needs additional installations of suitable DBSL and DB Client, and then connects to the Remote DB Server using the connection parameters as defined in the DBCON entry (“multi-connect”).
The ADBC interface is an ABAP/OO-based API and supports all official SAP-supported Database Management Systems (DBMS). The mentioned SAP Notes provide additional information on how to setup the secondary database connections (also known as database multiconnect).

<table>
<thead>
<tr>
<th>DBMS</th>
<th>Description</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADA</td>
<td>SAP DB (ADABAS/D)</td>
<td>Note 955670 &quot;DB multiconnect with MaxDB as secondary database&quot;</td>
</tr>
<tr>
<td>DB2</td>
<td>DB2 UDB for OS/390</td>
<td>Note 160484 &quot;DB2/390: Database multiconnect with EXEC SQL&quot;</td>
</tr>
<tr>
<td>DB4</td>
<td>DB2 UDB for AS/400</td>
<td>Note 146624 &quot;i5/OS: Database multiconnect for IBM DB2 for i5/OS&quot;</td>
</tr>
<tr>
<td>DB6</td>
<td>DB2 UDB for Unix/Win.</td>
<td>Note 200164 &quot;DB multiconnect with DB6 as target database&quot;</td>
</tr>
<tr>
<td>HDB</td>
<td>SAP HANA database</td>
<td>Note 1597627 &quot;SAP HANA connection&quot;</td>
</tr>
<tr>
<td>MSS</td>
<td>Microsoft SQL Server</td>
<td>Note 178949 &quot;MSSQL: Database MultiConnect with EXEC SQL&quot;</td>
</tr>
<tr>
<td>ORA</td>
<td>Oracle</td>
<td>Note 339092 &quot;DB MultiConnect with Oracle as secondary database&quot;</td>
</tr>
<tr>
<td>SYB</td>
<td>Sybase ASE</td>
<td>Note 1507573 &quot;External DB connect to a Sybase ASE database&quot;</td>
</tr>
</tbody>
</table>

The ADBC interface needs some technical prerequisites in order to be able to connect to a remote database.

**Database Logon User**

On the remote database, there must be a database logon user available that has a read-only access to the database schema with the required application tables for monitoring. For setting up the database connection, the password of this user must be known. You may want to setup multiple database connections to the same database instance, which just differ by the used login name, in order to grant different authorizations on database level.

Regarding the authorization check on the SAP ABAP stack, please see further below.
**Database Client and DBSL**

On the monitoring system (where the data collector runs), so likely the Solution Manager itself, there must be

- a DBMS-specific **database client**, provided by the DBMS vendor.
- an SAP **DBSL** for the connected DBMS. A DBSL is a Database Shared Library, provided by SAP as a database-dependent "add-on" library for the database-independent SAP Kernel.

**Notes:**

- This prerequisite is not necessary (or more precise: already fulfilled), if the DBMS of the Solution Manager's SAP instance is identical to the DBMS of the remote database. Example: The Solution Manager runs on an IBM DB6 database. So a DB client and a DBSL are already installed for the local (=primary) database access. The SAP Database Interface (DBIF) can reuse these libraries for a remote database connection (=secondary connection) to another DB6 instance using ADBC.
- Ideally additional DB clients and DBMS are installed on all application servers of the SAP system, but minimum on at least one application server. If DB client and DBMS are not available on all application servers, the execution of the data collector must be forced on a specific application server. For that further system settings (e.g. additional RFC Destinations or RFC Server Groups) might be necessary.
- DB Clients are available from the respective database vendor company. However, not all vendors provide DB Clients for all operating systems (that are used as supported SAP platforms).

**DB Connection Entry**

For each connected secondary database, a connection entry must be available in table DBCON. This connection information table is maintained via transaction DBCO, or as of SAP Basis 7.0 via the more user-friendly transaction DBACOCKPIT. In DBACOCKPIT you need to switch the UI view to "DB Connections" by pressing the corresponding push button.
A connection entry defines the technical details (like database host, port number, connection parameters, etc.) and security information (like logon user and encrypted password). Make sure that the required remote database is available in transaction DBACOCKPIT. Perform a connection test to verify that the remote database is reachable.

**Authorizations**

There is an authorization check on the SAP ABAP stack for the remote database access, based on the authorization object S_DBCON. The monitoring users will need an authorization that has the name of the DBCON connection as “Server Name” and allowance for Activity 71 ("Analyze"). The rest of the fields can have wildcard "*". The “Database Name” and “Database User” have no meaning in this context, as the technical connection parameters and the login credentials are already defined in the corresponding DBCON entry.

Example for a database connection called MDM12:

![Authorization Object for a database connection](image)
This will prevent the misuse of other DB connections (where maybe more than read-only rights are available).

The authorization check happens during the following actions:
- Value help display in BPMon Setup, using the dialog user of the login RFC connection
- Validation check in BPMon Setup, using the system user of the read RFC connection
- Data collection for BPMon alert measurement, using the system user of the read RFC connection
- Detail display for a BPMon alert, using the dialog user of the login RFC connection

So the authorization profile as discussed above must be assigned to
- the BPMon background user (BPM_LOCAL) and
- all named users who either perform the BPMon Setup or need to execute the detail analysis.

Please note that for the detail analysis there is an additional general authorization check performed on the Solution Manager itself based on authorization object BPM_DETAIL.

**12.2 Derive ADBC Connection Entry**

This entry is split into two different entries. Depending on your release you derive the information either from SMSY or from the LMDB. Have a look at chapter 10.2.1 in case you have a Solution Manager with a release lower than 7.1 and Support Package 05. In the case you have a Solution Manager with release 7.1 Support Package 05 or higher, please have a look at chapter 10.2.2.

**12.2.1 Derive ADBC Connection Entry from System Landscape (SMSY) – Lower SolMan 7.1 SP05**

You can store the remote database connection entry (DBCON name) inside the System Landscape definition of SAP Solution Manager (transaction SMSY). This has the advantage, that this rather technical entry is separated from the application monitor definitions. So you have a central maintenance place in case the DBCON name needs to be changed, and you do not need to update the technical connection details within each application monitor. This can be compared with the storage of the RFC Destination names to define the technical connection to other ABAP-based SAP systems.

Note: On Solution Manager, you need software component ST-SER to be on release 2010.1 or later.

The landscape component “Database” has been extended with additional attributes that allow the storage of the technical connection details for application monitoring:
- SAP_DBCON_APP Remote DB Conn. for Appl.Mon.
- SAP_SCHEMA_APP DB Schema for Application Data

For general instructions on how to work with the Solution Manager System Landscape, please refer to the SAP Help Portal at [http://help.sap.com](http://help.sap.com) => SAP Solution Manager => SAP Solution Manager 7.0 EHP1 SP23 => Basic Settings => Solution Manager System Landscape.
For our purpose we need the following **landscape components**:

- **Server** (optional) to manage the host of the remote database
- **Database** to manage the database instance with its connection attributes
- **System** to manage the non-ABAP/non-SAP system with its database instance
- **System Components** to support non-ABAP main instances
- **Logical Component** to identify the main instance in its correct release and role

**Example for a TriplePoint non-ABAP Database:**

The logical component Z_TPT_CSL_SMJ_ORA refers to product TPT COMMODITY SL with main instance CSL in version 5.6.

![Picture 71: Example for a logical component](image)

Its production system is called TPTSMJOR and its relevant main instance is a database instance, but is no ABAP instance.

![Picture 72: Defining the system as a database instance](image)
Within the system definition there is a link to the database instance ZTN_TPTORA.

Here you find the additional attributes as mentioned above for storing the connection name and the application schema name.

The referenced DBCON entry ZTN_TPT_PWDF6567_ORA is maintained directly in SAP Solution Manager (transaction DBACOCKPIT).

So if there is an application monitor of type “Remote DB” for the logical component Z_TPT_CSL_SMJ_ORA, the system would derive the connection name.
The data collector will run on the ABAP-stack of SAP Solution Manager and establish a secondary database connection to the TriplePoint application database.

12.2.2 Derive ADBC Connection Entry from Landscape Management Database (LMDB) – SolMan 7.1 SP05 or higher

You can store the remote database connection entry (DBCON name) inside Landscape Management Database definition of SAP Solution Manager (transaction LMDB). This has the advantage, that this rather technical entry is separated from the application monitor definitions. So you have a central maintenance place in case the DBCON name needs to be changed, and you do not need to update the technical connection details within each application monitor. This can be compared with the storage of the RFC Destination names to define the technical connection to other ABAP-based SAP systems.

Note: On Solution Manager, you need software component ST-SER to be on release 2010.1 or later.

The landscape component “Database” has been extended with additional attributes that allow the storage of the technical connection details for application monitoring:

- SAP_DBCON_APP Remote DB Conn. for Appl.Mon.
- SAP_SCHEMA_APP DB Schema for Application Data

For general instructions on how to work with the Solution Manager Landscape Management Database (LMDB), please refer to the SAP Help Portal at http://help.sap.com => SAP Solution Manager => SAP Solution Manager 7.1 SP05 => SAP Solution Manager Operations => Managing System Landscape Information

For our purpose we need the following landscape components:

- **Server** (optional) to manage the host of the remote database
- **Database** to manage the database instance with its connection attributes
- **System** to manage the non-ABAP/non-SAP system with its database instance
  - Supported technical system types:
    - Database System
    - Unspecific Cluster System
    - Microsoft Internet information service
    - TREX System
    - SAP liveCache
    - SAP NetWeaver Master Data Management
    - SAP Web Dispatcher
    - SAP Business Object Cluster
- **System Components** to support non-ABAP main instances
- **Logical Component** to identify the main instance in its correct release and role (has to be maintained in SMSY and not LMDB!)
Example for a TriplePoint non-ABAP Database:

The logical component Z_TPT_CSL_SMJ_ORA (transaction SMSY) refers to product TPT COMMODITY SL with main instance CSL in version 5.6.

Its system is called TPTSMJOR (transaction LMDB), it is no ABAP System but has a database instance assigned.
Within the system definition there is a link to the database instance ZTN_TPTORA.

Here you find the additional attributes as mentioned above for storing the connection name and the application schema name.

<table>
<thead>
<tr>
<th>Custom Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Scheme for Application Data</td>
<td>CSL71_BASELINE</td>
</tr>
<tr>
<td>Remote DB Verb. for Appl. Mon.</td>
<td>ZTN_TPT_PWDF567_ORA</td>
</tr>
</tbody>
</table>
The referenced DBCON entry ZTN_TPT_PWDF6567_ORA is maintained directly in SAP Solution Manager (transaction DBACOCKPIT).

So if there is an application monitor of type “Remote DB” for the logical component Z_TPT_CSL_SMJ_ORA, the system would derive the connection name ZTN_TPT_PWDF6567_ORA. The data collector will run on the ABAP-stack of SAP Solution Manager and establish a secondary database connection to the TriplePoint application database.

12.3 Dynamic Date Filtering for remote database monitoring

There are many use cases, where the result of a table count is time-dependent. For example, we might be interested only in the amount of open sales order items, where the requested delivery date is older than 30 days. However, inside the BPMon Setup Session there are only static filter values, like a fixed day entry for a date field. To achieve a dynamic date filtering, which means calculating relative dates instead of absolute dates, the Table Counter provides a special syntax on how to define fix points and offsets for relative dates.

Syntax for Relative Dates

Instead of a fixed (absolute) date, you can enter a special keyword for the start date (prefixed by a $ character) and optionally an additional offset as difference in days.

Syntax = <StartDate>[<Difference>]
For \texttt{<StartDate>}, the keywords shown in Table \ref{tab:DATEFILTER} are available:

<table>
<thead>
<tr>
<th>Keyword for \texttt{&lt;StartDate&gt;}</th>
<th>Description</th>
<th>Example for 2008/09/24</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{TODAY}$</td>
<td>current date</td>
<td>2008/09/24</td>
</tr>
<tr>
<td>$\text{FDOCM}$</td>
<td>first day of current month</td>
<td>2008/09/01</td>
</tr>
<tr>
<td>$\text{LDOCM}$</td>
<td>last day of current month</td>
<td>2008/09/30</td>
</tr>
<tr>
<td>$\text{FDOCY}$</td>
<td>first day of current year</td>
<td>2008/01/01</td>
</tr>
<tr>
<td>$\text{LDOCY}$</td>
<td>last day of current year</td>
<td>2008/12/31</td>
</tr>
<tr>
<td>$\text{FDOPM}$</td>
<td>first day of previous month</td>
<td>2008/08/01</td>
</tr>
<tr>
<td>$\text{LDOPM}$</td>
<td>last day of previous month</td>
<td>2008/08/31</td>
</tr>
<tr>
<td>$\text{FDOPY}$</td>
<td>first day of previous year</td>
<td>2007/01/01</td>
</tr>
<tr>
<td>$\text{LDOPY}$</td>
<td>last day of previous year</td>
<td>2007/12/31</td>
</tr>
<tr>
<td>$\text{FDONM}$</td>
<td>first day of next month</td>
<td>2008/10/01</td>
</tr>
<tr>
<td>$\text{LDONM}$</td>
<td>last day of next month</td>
<td>2008/10/31</td>
</tr>
<tr>
<td>$\text{FDONY}$</td>
<td>first day of next year</td>
<td>2009/01/01</td>
</tr>
<tr>
<td>$\text{LDONY}$</td>
<td>last day of next year</td>
<td>2009/12/31</td>
</tr>
<tr>
<td>$\text{TIMES}$</td>
<td>current timestamp</td>
<td></td>
</tr>
<tr>
<td>$\text{DELTA}$</td>
<td>delta mode (NATAABCNX only)</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Key words for dynamic date filtering
\textit{(these keywords are also available as value help, if you enter a $ into the filter field)}

The optional \texttt{<Difference>} is entered as positive or negative offset in days
- using \texttt{+} increments days (= move start date into the future)
- using \texttt{-} decrements days (= move start date into the past)

\textit{Note: On certain frontend UIs, entering a \texttt{+} (plus sign) may be interpreted as a wildcard character. This can even change the selection option from \texttt{equal} to \texttt{contains pattern}. To avoid this, you can also use the \texttt{"#"} \text{(number/hash sign)} instead of \texttt{\textquoteright+\textquoteright}.}

Examples:
- \texttt{$\text{TODAY}$-2} = day before yesterday
- \texttt{$\text{TODAY}$+2} = day after tomorrow

The dynamic date selection should be combined with a relation symbol (such as greater than or less than) or used in ranges (using the “from” and “to” fields of the selection criteria), for example to define “between” intervals.
Example: You want to count the amount of documents which have a creation date in last month. The selection criteria would be \texttt{<FieldName> = $\text{FDOPM}$ to $\text{LDOPM}$}.

Back to our example: With the following setting you would select all sales order items, where the requested delivery date is older than 30 days only.

\begin{center}
\begin{tabular}{|c|c|}
\hline
Filter 2 Fieldname & REQ_DLV_DATE \\
Filter 2 Sel options & $\text{TODAY}$-30 \\
\hline
\end{tabular}
\end{center}

At runtime, this will be converted into an appropriate where-clause, like \texttt{REQ_DLV_DATE <= '2009-10-19 00:00:00'} (on execution date 2009-11-18).

For the key words \texttt{$\text{TIMES}$} and \texttt{$\text{DELTA}$} the Table Entry Counter calculates an exact time, not only a date (available from ST-A/PI 01P only).
\texttt{$\text{TIMES}$} just takes the current system timestamp at the time of data collection.
\texttt{$\text{DELTA}$} uses the stored timestamp of the previous run of the data collector.
Both also support an additional \texttt{<Difference>} offset, but please note that here the value is interpreted as minutes (instead of whole days).
12.4 WSDL-Document

You can display the WSDL for the PULL-Web service in transaction SOAMANAGER under Service Administration → Single Service Configuration by searching for the internal name DSWP_BPM_PUSH_WS. Select the service and use the link “Open WSDL document for selected binding” (see Picture 81).

Picture 81: Accessing the WSDL for a Web service

The current WSDL is shown in Listing 5.

```xml
<?xml version="1.0" encoding="utf-8" ?>
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
  xmlns:n1="urn:sap-com:document:sap:rfc:functions">
- <wsdl:documentation>
</wsdl:documentation>
- <wsp:UsingPolicy wsdl:required="true" />
- <wsp:Policy wsu:Id="BN_BN_DSWP_BPM_WS_PUSH">
  <saptrnbd:OptimizedXMLTransfer
  <saptrnbd:OptimizedXMLTransfer
    uri="http://www.w3.org/2004/08/soap/features/http-optimization"
</wsp:Policy>
```

   xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
   xmlns:wsa="http://www.w3.org/2005/08/addressing"
   xmlns:wsst="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
  - <sp:All>
    - <sp:TransportBinding>
      - <wsp:Policy>
        - <sp:TransportToken>
          - <wsp:Policy>
            - <sp:HttpsToken>
              - <wsp:Policy>
                - <sp:HttpBasicAuthentication />
                  </wsp:Policy>
                </sp:HttpsToken>
              </wsp:Policy>
            </sp:TransportToken>
          - <sp:AlgorithmSuite>
            - <wsp:Policy>
              - <sp:TripleDesRsa15 />
                </wsp:Policy>
            </sp:AlgorithmSuite>
          - <sp:Layout>
            - <wsp:Policy>
              <sp:Strict />
              </wsp:Policy>
            </sp:Layout>
          </wsp:Policy>
        </sp:TransportBinding>
      </wsp:Policy>
    </sp:All>
  - <wsp:ExactlyOne>
    - <wsp:Policy wsu:Id="IF_IF_DSWP_BPM_PUSH_WS">
        <sapsession:enableSession>false</sapsession:enableSession>
      </sapsession:Session>
    </wsp:Policy>
  - <wsp:Policy wsu:Id="OP_IF_OP_DswpBpmWsPush">
    <sapcomhnd:enableCommit xmlns:sapcomhnd="http://www.sap.com/NW05/soap/features/commit/" false
    </sapcomhnd:enableCommit>
    <sapblock:enableBlocking xmlns:sapblock="http://www.sap.com/NW05/soap/features/blocking/" true
    </sapblock:enableBlocking>
    <saptrhnw05:required xmlns:saptrhnw05="http://www.sap.com/NW05/soap/features/transaction/" no
    </saptrhnw05:required>
<saprmnw05:enableWSRM xmlns:saprmnw05="http://www.sap.com/NW05/soap/features/wsrm/">false</saprmnw05:enableWSRM>
</wsp:Policy>
- <xsd:types>
  - <xsd:schema attributeFormDefault="qualified" targetNamespace="urn:sap-com:document:sap:rfc:functions">
    - <xsd:simpleType name="char1">
      - <xsd:restriction base="xsd:string">
        <xsd:maxLength value="1" />
      </xsd:restriction>
    </xsd:simpleType>
    - <xsd:simpleType name="char12">
      - <xsd:restriction base="xsd:string">
        <xsd:maxLength value="12" />
      </xsd:restriction>
    </xsd:simpleType>
    - <xsd:simpleType name="char20">
      - <xsd:restriction base="xsd:string">
        <xsd:maxLength value="20" />
      </xsd:restriction>
    </xsd:simpleType>
    - <xsd:simpleType name="char200">
      - <xsd:restriction base="xsd:string">
        <xsd:maxLength value="200" />
      </xsd:restriction>
    </xsd:simpleType>
    - <xsd:simpleType name="char60">
      - <xsd:restriction base="xsd:string">
        <xsd:maxLength value="60" />
      </xsd:restriction>
    </xsd:simpleType>
    - <xsd:simpleType name="numeric6">
      - <xsd:restriction base="xsd:string">
        <xsd:maxLength value="6" />
        <xsd:pattern value="\d*" />
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:schema>
</xsd:types>
- <xsd:complexType name="DswpBpmWsTransfer">
  - <xsd:sequence>
    - <xsd:element name="Ident1" type="n0:char60" />
    - <xsd:element name="Ident2" type="n0:char60" />
    - <xsd:element name="MeasValue" type="xsd:int" />
    - <xsd:element name="Object1" type="n0:char60" />
    - <xsd:element name="Object2" type="n0:char60" />
    - <xsd:element name="Object3" type="n0:char60" />
    - <xsd:element name="Info1" type="n0:char60" />
    - <xsd:element name="Info2" type="n0:char60" />
    - <xsd:element name="Info3" type="n0:char60" />
    - <xsd:element name="Info4" type="n0:char60" />
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
<xsd:element name="RemoteUser" type="n0:char12" /> 
</xsd:sequence> 
</xsd:complexType> 
- <xsd:complexType name="DswpBpmSWsReturn"> 
- <xsd:sequence> 
  <xsd:element name="Type" type="n0:char1" /> 
  <xsd:element name="Message" type="n0:char200" /> 
  <xsd:element name="LogNo" type="n0:char20" /> 
  <xsd:element name="LogMsgNo" type="n0:numeric6" /> 
</xsd:sequence> 
</xsd:complexType> 
- <xsd:complexType name="DswpBpmTtWsTransfer"> 
- <xsd:sequence> 
  <xsd:element name="item" type="tns:DswpBpmWsTransfer" minOccurs="0" maxOccurs="unbounded" /> 
</xsd:sequence> 
</xsd:complexType> 
- <xsd:complexType name="DswpBpmTtWsReturn"> 
- <xsd:sequence> 
  <xsd:element name="item" type="tns:DswpBpmSWsReturn" minOccurs="0" maxOccurs="unbounded" /> 
</xsd:sequence> 
</xsd:complexType> 
- <xsd:element name="DswpBpmWsPush"> 
- <xsd:complexType> 
- <xsd:sequence> 
  <xsd:element name="Input" type="tns:DswpBpmTtWsTransfer" /> 
</xsd:sequence> 
</xsd:complexType> 
- <xsd:element name="DswpBpmWsPushResponse"> 
- <xsd:complexType> 
- <xsd:sequence> 
  <xsd:element name="Result" type="tns:DswpBpmTtWsReturn" /> 
</xsd:sequence> 
</xsd:complexType> 
- <xsd:message name="DswpBpmWsPush"> 
  <xsd:part name="parameters" element="tns:DswpBpmWsPush" /> 
</xsd:message> 
- <xsd:message name="DswpBpmWsPushResponse"> 
  <xsd:part name="parameters" element="tns:DswpBpmWsPushResponse" /> 
</xsd:message> 
- <xsd:portType name="DSWP_BPM_PUSH_WS"> 
- <wsp:Policy> 
  <wsp:PolicyReference URI="#IF_IF_DSWP_BPM_PUSH_WS" /> 
</wsp:Policy> 
- <xsd:operation name="DswpBpmWsPush"> 
- <wsp:Policy> 
  <wsp:PolicyReference URI="#OP_IF_OP_DswpBpmWsPush" /> 
</wsp:Policy> 
  <xsd:input message="tns:DswpBpmWsPush" />
Listing 5: WSDL document for the provided Web service to PUSH data into SAP Solution Manager

12.5 Derive BW Connection Entry
This entry is split into two different entries. Depending on your release you derive the information either from SMSY or from the LMDB. Have a look at 9.5.1 in case you have a Solution Manager with a release lower than 7.1 and Support Package 05. In the case you have a Solution Manager with release 7.1 Support Package 05 or higher, please have a look at 9.5.2.

12.5.1 Derive BW Connection Entry from System Landscape (SMSY)
You can store the RFC connection entry to the BW-system inside the System Landscape definition of SAP Solution Manager (transaction SMSY). This has the advantage, that this rather technical entry is separated from the application monitor definitions. So you have a central maintenance place in case the RFC connection needs to be changed, and you do not need to update the technical connection details for each application monitor NABWICNX accessing the same system. This can be compared with the storage of the standard RFC Destination names to define the technical connection to monitor ABAP-based SAP systems.

Note: As Solution Manager, you need to have Solution Manager Version 7.0 with SPS 23 or a later SPS installed.
The landscape component “**System**” has been extended with additional attributes that allow the storage of the technical connection details for application monitoring:

- SAP_BW_RFC_BPM RFC Dest. For BPMon BW Alerting

For general instructions on how to work with the **Solution Manager System Landscape**, please refer to the SAP Help Portal at [http://help.sap.com](http://help.sap.com) => SAP Solution Manager => Help for SAP Solution Manager 7.0 EHP1 SP23 => Basic Settings => Solution Manager System Landscape.

For our purpose we need the following **landscape components**:

- **System** to manage the non-ABAP/non-SAP system with its database instance
- **System Components** to support non-ABAP main instances
- **Logical Component** to identify the main instance in its correct release and role

**Example for a connection to BW for a non-SAP system:**

The logical component NON_SAP refers to product ZNONSAP with main instance ZNONSAP.

```
<table>
<thead>
<tr>
<th>Logical Component</th>
<th>NON_SAP</th>
<th>A Active</th>
<th>Saved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product/Proc. Inst</td>
<td>ZNONSAP [ZNONSAP]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>NON_SAP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delivered by SAP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component Group</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last Changed On/By</td>
<td>11.05.2010 13:29:58</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Picture 82: A logical component**

Its production system is called NONSAP and its relevant main instance is no ABAP instance.

<table>
<thead>
<tr>
<th>Product System</th>
<th>NONSAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Version</td>
<td>ZNONSAP 01</td>
</tr>
</tbody>
</table>

**Picture 83: Defining a main instance for non-ABAP-monitoring**
You find the additional attributes as mentioned above for storing the connection name in the system definition on tab “Other Attributes”.

### Picture 84: The SMSY-parameters for the BW-connection

So if there is an application monitor of type “Generic Infocube/DSO Reader for remote BW” for the logical component NONSAP, the Solution Manager would derive the connection name NONE. The data collector will run on the ABAP-stack of SAP Solution Manager and establish a secondary RFC connection to the relevant BW system (which in this case resides on the monitoring client).

### 12.5.2 Derive BW Connection Entry from Landscape Management Database (LMDB)

You can store the RFC connection entry to the BW-system inside the Landscape Management Database definition of SAP Solution Manager (transaction LMDB). This has the advantage, that this rather technical entry is separated from the application monitor definitions. So you have a central maintenance place in case the RFC connection needs to be changed, and you do not need to update the technical connection details for each application monitor NABWICNX accessing the same system. This can be compared with the storage of the standard RFC Destination names to define the technical connection to monitor ABAP-based SAP systems.

Note: As Solution Manager, you need to have Solution Manager Version 7.1 with SPS 05 or a later SPS installed.

The landscape component “System” has been extended with additional attributes that allow the storage of the technical connection details for application monitoring:

- SAP_BW_RFC_BPM RFC Dest. For BPMon BW Alerting

For general instructions on how to work with the Solution Manager Landscape Management Database (LMDB), please refer to the SAP Help Portal at http://help.sap.com => SAP Solution Manager => SAP Solution Manager 7.1 SP05 => SAP Solution Manager Operations => Managing System Landscape Information

For our purpose we need the following **landscape components**:

- **System** to manage the non-ABAP/non-SAP system with its database instance
- **System Components** to support non-ABAP main instances
- ** Logical Component** to identify the main instance in its correct release and role
Example for a connection to BW for a non-SAP system:

The logical component NON_SAP refers to product ZNONSAP with main instance ZNONSAP.

<table>
<thead>
<tr>
<th>Logical Component</th>
<th>NON_SAP</th>
<th>A Active</th>
<th>Saved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Prod. Instance</td>
<td>ZNONSAP [ZNONSAP]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>NON_SAP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delivered by SAP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component Group</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last Changed On/By</td>
<td>11.05.2018 13:29:55</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Picture 85: A logical component

Its production system is called NONSAP and its relevant main instance is no ABAP instance.

Picture 86: Defining a technical system instance for non-ABAP-monitoring
You find the additional attributes as mentioned above for storing the connection name in the system definition on tab “Custom Attributes”.

So if there is an application monitor of type “Generic Infocube/DSO Reader for remote BW” for the logical component NONSAP, the Solution Manager would derive the connection name NONE. The data collector will run on the ABAP-stack of SAP Solution Manager and establish a secondary RFC connection to the relevant BW system (which in this case resides on the monitoring client).

12.6 Dynamic Date Filtering for BW-queries

There are many use cases, where the desired result of a query is time-dependent. For example, we might be interested only in the amount of errors yesterday or within the last minutes. However, inside the BPMon Setup Session there are only static filter values, like a fixed day entry for a date field. To achieve a dynamic date filtering, which means calculating relative dates instead of absolute dates, the monitoring object NABWICNX provides a special syntax on how to define fix points and offsets for relative dates.

Syntax for Relative Dates

Instead of a fixed (absolute) date, you can enter a special keyword for the start date (prefixed by a $ character) and optionally an additional offset as difference in days.

Syntax = <StartDate>[<Difference>]
For \(<\text{StartDate}\)\> the keywords shown in Table 4 are available:

<table>
<thead>
<tr>
<th>Keyword for (&lt;\text{StartDate})&gt;</th>
<th>Description</th>
<th>Example for 2008/09/24</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TODAY</td>
<td>current date</td>
<td>2008/09/24</td>
</tr>
<tr>
<td>$TIMES</td>
<td>Current date and time</td>
<td>2008/09/24 14:00:00</td>
</tr>
</tbody>
</table>

Table 4: Key words for dynamic date filtering for BW

The optional \(<\text{Difference}\)\> is entered as positive or negative offset in days
- using \('+'\) increments days (= move start date into the future)
- using \('-'\) decrements days (= move start date into the past)

*Note: On certain frontend UIs, entering a ‘+’ (plus sign) may be interpreted as a wildcard character. This can even change the selection option from ‘equal’ to ‘contains pattern’. To avoid this, you can also use the ‘#’ (number/hash sign) instead of ‘+’.*

Examples:
- $TODAY-2 = day before yesterday
- $TODAY+2 = day after tomorrow
- $TIMES-30 = The last 30 minutes before the time of data collection

The system will identify the characteristics to be used to determine the newest row for aggregation ‘LST’ based on the use of dynamic date filtering. It will use the characteristics with the dynamic date and will evaluate the row with the newest value in this characteristic.

### 13 Appendix II: Test Environment for Remote DB Query (program /SSA/ENA)

#### 13.1 Purpose

The “Test Environment for Remote DB Query” provides the possibility to run generic database queries to any database table of a remote database. It supports a flexible filtering based on select-options on five fields of a table. Using ADBC (ABAP Database Connectivity) it can access a remote database table on an SAP-supported DBMS (database management system).

The “Test Environment” is part of the Business Process Monitoring framework within SAP Solution Manager. It can be used to prepare queries for the “Generic Table Entry Counter for Remote DB” (monitoring object NATABCNX) or for generic monitoring objects with free SQL queries towards Remote Databases.

#### 13.2 Technical Prerequisites

The Test Environment is delivered with the add-on ST-A/PI starting with release 01M. It can be started by running program /SSA/ENA with transaction SA38.

All SAP Basis (NetWeaver) releases starting from release 7.00 and later are supported.

To monitor an external (non-ABAP) database using ADBC, the corresponding secondary database connection must be maintained with transaction DBACOCKPIT. Please see appendix “Remote Database Connectivity” for more details.
This Test Environment shall not be used to query databases of SAP ABAP-based systems. For that purpose you can for example use transaction SE16, build especially to support database tables defined inside the SAP ABAP Data Dictionary.

13.3 Limitations
Because the Test Environment queries database tables, which are not defined inside the SAP ABAP Data Dictionary, there is no automatic support of certain build-in functionality, such as
- multi-lingual text labels for table columns,
- complex search helps on check tables or domain fix values,
- input and output conversion according to certain data types or user’s locale settings.

Search helps for defining filters are fed by selecting distinct field values from the corresponding table columns, which might be a performance-intensive query. On the selection screen, all filter criteria are stored as character fields, so there may occur implicit conversions, e.g. for number types or dates.

The Test Environment accesses the remote DB data dictionary to query the table meta data like column’s data types, in order to map the native data types into SAP dictionary data types and ABAP data types, for example to allow a detail display with the list viewer. However, it cannot be guaranteed that all native DBMS-dependent data types can be properly matched into a suitable ABAP data type. Therefore the output of certain data might look different compared to the user interface of native database tools or other non-ABAP applications. There might be restrictions especially for number types (like maximum amount of decimals) or date types (like other than expected formats for dates and timestamps).

13.4 Performance Warning
The Test Environment can query any table that is accessible by the login user. Therefore it is in the responsibility of the user to avoid frequent selections on large tables which might result in a performance degradation of the database. Keep in mind that the necessary SQL selection aggregates for COUNT and DISTINCT create expensive database selects, including full table scans!

Using the Test Environment without care can cause severe performance degradation on the monitored system.

Please consider your selection criteria carefully and follow these golden rules:
- Avoid using queries on large database tables
- Avoid running the queries as a background job with a high frequency
- Avoid complex filtering, especially with patterns, ranges and exclusions
- Avoid filters on fields which are not supported by an index.
13.5 Main Selection Screen

The main selection screen of the program (see Picture 88) consists of four areas:

1. **Remote Database Table** to define the connection target
2. **Processing Mode** to choose the type of database query
3. **Filter Criteria for Where-Clause** to define selection filter fields and values
4. **Display Parameters** to set some special output options
13.5.1 Processing Mode

The effect and usage of the different processing modes are explained in more detail further below. The settings for “Remote Database Table” and “Filter Criteria for Where-Clause” are identical for those processing mode. However, the available “Display Parameters” may differ per processing mode.

13.5.2 Remote Database Table

To identify the remote database table, you need to enter three attributes:

- The “Database Connection Name” is the name of a connection entry as defined in transaction DBACOCKPIT (stored in table DBCON). Please see appendix “Remote Database Connectivity” for more details. The search help shows the available entries. By pressing the test button, you can perform a connection test to the remote database.

- As “Database Schema Name” enter the name of the schema, where the table belongs to. The search help queries the remote database for available schemas. You can also press button to search for the default schema (attribute in transaction DBACOCKPIT) and take it as a proposal.

- In field “Table Name”, enter the name of the remote database table. You can use tables, views and synonyms. The search help queries the remote database for available tables in the given schema. Please avoid running this query for all tables. If you know
the first characters of the table name, you can enter it as a pattern to restrict the search help result.

![Remote Database Table](image)

**Picture 89: Search help for table names**

![Search help for table name](image)

By pressing button ✉ you can view the table’s meta data from the remote database dictionary, such as table columns, keys, indexes, or view definitions.

### 13.5.3 Filter Criteria for Where-Clause

You can use up to five pairs of field name and select-options, which are used to create a where-clause for your select query. Please note that the selection screen is organized like a standard ABAP report making use of the multiple select-options to support ranges and patterns. However, during runtime this will be converted into an ANSI SQL-conform where-clause.

In the first column, enter the table **field name**. You can use the search help to show a popup with the table’s column meta-data, including information about key fields and data types.

![Search help for field names](image)

**Picture 91: Search help for field names**

The columns “Field type”, “DB length”, “Precision” and “Scale” refer to the native data type in the remote database. The columns “DTyp”, “Length” and “Decimals” show the corresponding data type as it would be used in an SAP ABAP Dictionary. If the “DTyp” is empty, the native data type cannot be mapped into a suitable SAP Dictionary type.
In the from/to-columns on the right-hand side, enter the actual **selection criteria**. As mentioned, this can be entered using the known multiple-selection screen, using all available select-options, such as single and multiple values, patterns with wildcards, ranges with intervals, all combined either as inclusions or exclusions. Use a double-click on the filter value field, or press button ➡, to access the multiple-selection screen. All filter values are stored in a character format and should be entered according to the internal format of the native data type.

An example for a filter on all material types except those starting with prefix ‘Access’ is shown in Picture 92.

![Picture 92: Example for a filter](image)

At runtime it gets converted into the following where-clause:

```
WHERE (MATERIAL_TYPE NOT LIKE 'Access%')
```

Multiple filter values on the same field name are connected with a logical ‘OR’ and filter values on different field names are connected with a logical ‘AND’.

Please see appendix on how to define variable filters for **dynamic date selections**.

### 13.5.4 Display Parameters

At “**Free Text as Display Header**” you can enter any text that will be shown as a header line in the list output. This is helpful when storing the selection screen as a variant for a later re-use.

The parameter **“Distinct/Single Field Name”** is needed for the following use-cases:

- For processing mode “Count number of distinct values” you have to enter the name of the table column on which you want to apply the COUNT DISTINCT aggregate.
- For processing mode “Get single field value” you have to enter the name of the table column from which you want to select a single field value in order to convert it to an integer number.
- For processing mode “Display Table Content” you can optionally enter the name of a table column on which you want to apply the SELECT DISTINCT aggregate.

The parameter **“Maximum number of Rows”** is available for all selections, which do not perform a simple count, but rather load table entries for a list display, e.g. at “Display Table Content”. The default number is set to 1,000. Avoid increasing it to more than 10,000 rows for performance reasons on the remote database as well as on the SAP GUI Frontend.

The parameter **“Display Columns”** is available for all selections, which do not perform a simple count, but rather load table entries for a list display. If you do not enter any values in this parameter, all columns are shown that can be mapped from the native data type into an ABAP data type. Using the multiple-selection, you can maintain a list of column names for a restricted display. Using push button ➡️, you can easily pre-fill the multiple-selection with all available column names. Furthermore you can define column headings, if you postfix a double-dash plus the heading text to the column name as shown in Picture 93.
This will lead to a display as shown in Picture 94.

The parameter “Order Columns” is also available for all selections, which do not perform a simple count, but rather load table entries for a list display. It allows to sort the database and selection and screen output by one or more columns, e.g. ORDER BY "MATERIAL_NAME", "SALES_UOM". If you leave this parameter empty, you get the default sorting, usually by primary key.

13.6 Processing Mode “Count number of entries”
This processing mode does a simple SELECT COUNT(*) query to the target table with the given filter criteria. The result is an integer value.

You can use this query to simulate the calculation of the “alert measured value” of the BPMon Data Collector, e.g. for monitoring object “Generic Table Entry Counter” (NATABCNX, 1st key figure).

Example query:
```
SELECT COUNT(*)
FROM APP_MATERIAL
WHERE (APP_MATERIAL.MATERIAL_TYPE NOT LIKE 'Access%')
```
13.7 Processing Mode “Count number of distinct values”

This processing mode does a query of style SELECT COUNT(DISTINCT <fieldname>) on the target table with the given filter criteria. You need to enter the <fieldname> in parameter “Distinct/Single Field Name”. The result is an integer value.

You can use this query to simulate the calculation of the “alert measured value” of the BPMon Data Collector, e.g. for monitoring object “Generic Table Entry Counter” (NATABCNX, 2nd key figure).

Example query:
```
SELECT COUNT( DISTINCT MATERIAL_TYPE )
FROM APP_MATERIAL
WHERE ( APP_MATERIAL.MATERIAL_TYPE NOT LIKE 'Access%' )
```

13.8 Processing Mode “Get single field value”

This processing mode does a query of style SELECT SINGLE <fieldname> on the target table with the given filter criteria. You need to enter the <fieldname> in parameter “Distinct/Single Field Name”. It needs to be an integer data type or at least some number data type that can be converted into an integer by rounding. The final result will always be an integer value. You should make sure that the filter criteria restrict the selection to one single row of the table, otherwise you would get the first row that matches the filter criteria and thus the result may not be accurate or unique.

Optionally you can apply one of the following aggregates:
- MIN = Minimum value
- MAX = Maximum value
- AVG = Average value
- SUM = Sum of values
This would change the query to the style `SELECT <aggregate>(<fieldname>). Also here the result is a (rounded) integer value.

You can use this query to simulate the calculation of the “alert measured value” of the BPMon Data Collector, e.g. for monitoring object “Generic Table Entry Counter” (NATABCNX, 3rd key figure).

**Example query without aggregate:**
```sql
SELECT MATERIAL_NUMBER
FROM APP_MATERIAL
WHERE (APP_MATERIAL.MATERIAL_TYPE NOT LIKE 'Access%')
```

![Example query without aggregate](image)

**Example query with aggregate:**
```sql
SELECT MAX(MATERIAL_NUMBER)
FROM APP_MATERIAL
WHERE (APP_MATERIAL.MATERIAL_TYPE NOT LIKE 'Access%')
```

![Example query with aggregate](image)

**13.9 Processing Mode “Display Table Content”**

This processing mode does a query of style `SELECT <fieldlist>` or `SELECT *` on the target table with the given filter criteria, depending on whether you have or not have maintained the parameter “Display Columns”.

As a variant, you can enter a `<fieldname>` in parameter “Distinct/Single Field Name”. This would change the query to the style `SELECT DISTINCT <fieldname>, COUNT(*) ... GROUP BY <fieldname>`.

The result of the queries is displayed as a list of found rows.

You can use this query to simulate the “Detail Info” of the BPMon Data Collector, e.g. for monitoring object “Generic Table Entry Counter” (NATABCNX).
Example query without distinct field:
```sql
SELECT "MATERIAL_NUMBER" , "MATERIAL_NAME" , "BASE_UOM" , "SALES_UOM" , "MATERIAL_TYPE"
FROM APP_MATERIAL
WHERE ( APP_MATERIAL.MATERIAL_TYPE NOT LIKE 'Access%' )
ORDER BY "MATERIAL_NAME" , "SALES_UOM"
```

![Database Connection](image1.png)

**Picture 99: Example for display of table content**

Example query with using the distinct field MATERIAL_TYPE for grouping:
```sql
SELECT DISTINCT MATERIAL_TYPE , COUNT(*)
FROM APP_MATERIAL
WHERE ( APP_MATERIAL.MATERIAL_TYPE NOT LIKE 'Access%' )
GROUP BY MATERIAL_TYPE
ORDER BY MATERIAL_TYPE
```

![Database Connection](image2.png)

**Picture 100: Example using grouping**

13.10 **Processing Mode “Compare Table with second connection”**

This processing mode is an experimental feature. It allows a comparison of two tables that are on different databases, by using data fetches from two connections. The table design needs to be identical, i.e. having the same column names with the same data types. The name of the tables may differ though.
You can enter two triples of “Database Connection Name”, “Database Schema Name” and “Table Name”, to identify the two target tables.

There are three additional parameters you can use for this processing mode:

1. “Key field(s)” allows entering the number of key fields. The key is important to match identical rows from the two tables. If they have more than one primary key field (such as order number plus item number), the amount must be entered. This would treat the first X columns from the meta data as combined key.
2. “Hide equal rows” suppresses the display of all fully identical rows.
3. “Hide equal columns” suppresses the display of all fully identical columns.

In the list output you see the comparison result with the following structure:

- On the left-hand side the table columns of the first table
- On the right-hand side the table columns of the second table
- In the middle a column called “Compare” that shows an icon for the comparison result:
  
  - `=>` the entry (with the same key(s)) exist in both tables and has identical content
  - `=>` the entry (with the same key(s)) exist in both tables but has different content
  - `=>` the entry is present in the left-hand table but is missing in the right-hand table
  - `=>` the entry is present in the right-hand table but is missing in the left-hand table

![Sample output without options “Hide equal rows” and “Hide equal columns”](image1)

![Sample output with option “Hide equal rows”](image2)

![Sample output with option “Hide equal columns”](image3)
By clicking on the button ![side-by-side view](image), you get a side-by-side view to compare the single table columns. Columns with different field content are highlighted in red.

**13.11 Processing Mode “Free Select Query”**

The actual SQL Select Statement is always assembled automatically for all the previously described processing modes to provide suitable query syntax for the specific use-case. Although this is very convenient and easy to use, one important restriction to observe is that you are not able to query multiple tables within one selection (unless you enter a view as table name, which internally joins several tables). With mode “Free Select Query” you can enter your own SQL Select Statement, for example to define a JOIN of several database bases, or to allow sub queries.

An example for a simple selection of the sales order items (table `APP_SALES_ORDER_ITEM`), using processing mode “Display Table Content” is shown in Picture 106.

**Picture 104: Sample output with both options “Hide equal rows” and “Hide equal columns”**

**Picture 105: Example for side-by-side view**

**Picture 106: Example for table contents**
Now we would like to see the material description, instead just the material number. So we need to join the field MATERIAL_NAME from table APP_MATERIAL.

```
SELECT
    APP_SALES_ORDER_ITEM.ORDER_NUMBER /*Order*/,
    APP_SALES_ORDER_ITEM.ITEM_NUMBER /*Item*/,
    APP_SALES_ORDER_ITEM.QUANTITY /*Quantity*/,
    APP_SALES_ORDER_ITEM.MATERIAL /*Mat.No.*/,
    APP_MATERIAL.MATERIAL_NAME /*Material*/,
    APP_SALES_ORDER_ITEM.ITEM_STATUS /*Status*/
FROM
    APP.APP_SALES_ORDER_ITEM
JOIN
    APP.APP_MATERIAL
ON
    APP_MATERIAL.MATERIAL_NUMBER = APP_SALES_ORDER_ITEM.MATERIAL
```

Chose processing mode “Free Select Query” and press button to start an editor for the SQL query which is shown in Picture 107.

![Edit SQL Command](Picture 107: Example for a free SQL entry)

Press the save button to return to the selection screen.

In order that your statement can be interpreted, you need to follow a couple of rules:

1. All SQL keywords (such as SELECT, FROM, JOIN, WHERE) must be in uppercase.
2. The SQL statement must start with a SELECT keyword.
3. The SELECT-clause must be either an aggregate that returns an integer value, for example SELECT COUNT(*) or SELECT COUNT( DISTINCT... ), or must contain a comma-separated field list that is subject to a list display, such as SELECT table1.field, table2.field2, and so on. In the latter case, please note that you have to use qualified field names (table designator plus dot plus column designator). This is important because for each field the column meta-data needs to be read per table. Aliases are not allowed. For each selection field you can add a text label, enclosed in SQL in-line comments (/*Label*/). This text label will be used as column header in the list display. Without that, you would get the technical table and field name only.
4. In the FROM-clause you can add any standard or native SQL syntax that can be interpreted by the remote database. This allows using sub queries, joins, and so on. There are no special checks done by the test environment. However, you should work with qualified table names (prefixed by the schema designator). You can also use the pseudo comment /*SCHEMA*/ which will be replaced by the schema name from the selection screen during runtime.
   Example:
   ```sql
   FROM /*SCHEMA*/APP_SALES_ORDER_ITEM
   JOIN /*SCHEMA*/APP_MATERIAL
   ON APP_MATERIAL.MATERIAL_NUMBER = APP_SALES_ORDER_ITEM.MATERIAL
   ```
5. In the WHERE-clause you can add the fixed part of your filters. Also here you should work with table-qualified field names. Additional filters as maintained on the selection screen will be added to the WHERE-clause as well. This is handy for defining variable filters like dynamic dates (see appendix).
6. Last but not least, add further clauses like GROUP BY and ORDER BY.
13.12 Further Diagnostics Features

13.12.1 Connection Test

A connection test is useful to check whether the remote database can be reached using the technical settings of the DBCON entry. It may happen that the connection cannot be established from all application servers of the calling system, e.g. when the DBSL or DB Client is not available. Please see appendix “Remote Database Connectivity” for more details.

There are two options for a connection test.

**Single Connection Test**

You can press button (next to the parameter “Database Connection Name”) to perform a single connection test to the entered remote database.

![Testing a DB connection](image108.png)

This will perform a connection test from the current application server to the remote database base (as defined by DBCON entry):

![Result of a connection test](image109.png)

Mark a line and press on the “Details” button to see a popup with all details (Picture 110)

![Details for a connection test](image110.png)

- “Database Connection Name” is the name of the DBCON entry
- “Remote Database System” is the type of DBMS for this connection
- “Application Server RFC Destination” is the name of the internal RFC destination
- “Application Server Host Name” is the host name of the tested application server
- “App.Server Local Operating System” is the type of the local OS on the application server
- “App.Server Local Database System” is the type of local DBMS (primary connection)
- “Available DBMS-DBSL on App.Server” shows for which DBMS a DBSL is available
- “Ping Time (seconds)” is the pure remote database connection test time
- “Total Time with RFC (seconds)” is the total test time that may include internal RFCs
- “Ping Status” is the result of the connection test
- “Error Message” would show the message text in case of failures
Mass Connection Test
You can press button (on top of the selection screen) to perform a mass connection test for several DBCON entries and even via several application servers.

First you have to fill out an additional selection screen (Picture 111) that will pop up.

![Remote Database Connection Test](Picture 111)

**Picture 111: Mass connection test**

- The parameter “Database Connection” allows a multiple-selection of several DBCON entries, including patterns and ranges.
- If the checkbox “Test via all Application Servers” is initial, the connection attempt is made from the application server where you are currently logged on to. If you activate the checkbox, there will be internal RFC calls to each application server to determine from which application server the database connection can be established successfully.
- By increasing the “Number of repetitive Pings” you can force more than one connection test to the same remote database.

Example for a mass connection test with three DBCON entries on current application server:

![Connection Test Result](Picture 112)

**Picture 112: Example 1 for a mass connection test**

- As you can see there are two application servers with host names lu0099 and vmw3193.
- The connections ZTN_DB6 and ZTN_DB6_2 could be tested successfully via application server lu0099, but failed to connect via application server vmw3193.
- The connection ZTN_DB6_COPY did not succeed to connect via either application server.

Example for a mass connection test with three DBCON entries on all application servers:

![Connection Test Result](Picture 113)

**Picture 113: Example 2 for a mass connection test**

- As you can see there are two application servers with host names lu0099 and vmw3193.
- The connections ZTN_DB6 and ZTN_DB6_2 could be tested successfully via application server lu0099, but failed to connect via application server vmw3193.
- The connection ZTN_DB6_COPY did not succeed to connect via either application server.

**Trouble-shooting:**
In above example, although each application server has a DBSL available for the DBMS “DB6”, on the second server we always get a “DBI error”. This indicates a problem detected by SAP’s Database Interface during establishing the secondary database connection.
In brackets you see the number of the work process and the application server name. This is helpful when searching for the correct work process trace to find further information on the root cause of the problem.
An extract from the corresponding work process trace file "dev_w2" is shown in Picture 114.

**Picture 114: Example for a work process trace file**

To connect to the secondary database, the Database Interface needs to load a certain DB2 library, called “db2app64.dll”, which is not available on this application server. So the installation of the DB Client seems to be incomplete.

### 13.12.12 Display Work Process Trace

The standard way to display a work process trace file would be using transaction SM50, mark the work process number, and then press button “Display File”. Furthermore you typically would also restrict the scope of display by choosing “Display Components” and marking only “Database” and “Database (DBSL)” as trace components.

You can reach that easily from the Test Environment, given that you are logged on to the same application server.

Press button ![Expert Mode](image) to switch on the advanced options.

Press button ![Display WP Trace](image) to call an additional selection screen (see Picture 115) to choose the work process number and the application server name.

**Picture 115: Selecting the work process to be displayed**

This will load the corresponding work process trace file restricted to the interesting trace components “Database” and “Database (DBSL)”. Scroll down to very bottom of the file to see the most recent entries. You can also search by the name of the DBCON entry.

### 13.12.2 Remote Database Dictionary Management

For processing remote database data correctly, such as type-conform list output, the framework needs to load certain dictionary information. This includes schema names, table names, table columns, key fields, primary and secondary index, or view definitions. To speed up processing, this kind of metadata is buffered inside the cross-transaction application buffers of the shared memory on the application server. Assuming that the remote data dictionary would not change so often, the buffer is kept valid for 24 hours. Afterwards the next dictionary query would read the requested remote database dictionary information again to refresh the buffer information.

**Display Remote Database Dictionary for one table**

To display the remote database dictionary information for one single table or view, simply press the button ![Table Name](image) next to the parameter “Table Name”. More information will be shown
(see Picture 116) than possibly already available in the buffer from previous queries, so there might be additional dictionary accesses to load all information types into the buffer.

### Remote Dictionary Information

<table>
<thead>
<tr>
<th>Connection</th>
<th>Schema</th>
<th>Database Table</th>
<th>Buffer Type</th>
<th>Columns</th>
<th>Buffer Content</th>
<th>Last Update Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZTN_DB5</td>
<td>APP</td>
<td>APP_MATERAL</td>
<td>Table Name</td>
<td>5</td>
<td>MATERIAL_NUMBER</td>
<td>2010-02-19 13:01:59 UTC</td>
</tr>
<tr>
<td>ZTN_DB5</td>
<td>APP</td>
<td>APP_MATERIAL</td>
<td>Table Columns</td>
<td>1</td>
<td>MATERIAL_NUMBER</td>
<td>2010-02-19 12:24:10 UTC</td>
</tr>
<tr>
<td>ZTN_DB5</td>
<td>APP</td>
<td>APP_MATERIAL</td>
<td>Primary Key</td>
<td>1</td>
<td>PK_MATERIAL</td>
<td>2010-02-19 12:24:10 UTC</td>
</tr>
<tr>
<td>ZTN_DB5</td>
<td>APP</td>
<td>APP_MATERIAL</td>
<td>Index PK</td>
<td>1</td>
<td>MATERIAL_NAME</td>
<td>2010-02-19 13:01:59 UTC</td>
</tr>
</tbody>
</table>

**Picture 116: Remote dictionary information for a table**

Double-click on column “Buffer Content” to get a popup with additional information, for example a list of all table columns, key fields, or index fields (see Picture 117).

### List of field names for a remote table

<table>
<thead>
<tr>
<th>Field Name</th>
<th>TabPos</th>
<th>Key</th>
<th>Field type</th>
<th>DB length</th>
<th>Pr</th>
<th>Scale</th>
<th>DTyp</th>
<th>Length</th>
<th>Decimals</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATERIAL_NUMBER</td>
<td>0000</td>
<td>X</td>
<td>INTEGER</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>INT4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>MATERIAL_NAME</td>
<td>0001</td>
<td>X</td>
<td>VARCHAR</td>
<td>40</td>
<td>0</td>
<td>0</td>
<td>CHAR</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>BASE_UOM</td>
<td>0002</td>
<td>X</td>
<td>VARCHAR</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>CHAR</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>SALES_UOM</td>
<td>0003</td>
<td>X</td>
<td>VARCHAR</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>CHAR</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>MATERIAL_TYPE</td>
<td>0004</td>
<td>X</td>
<td>VARCHAR</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>CHAR</td>
<td>20</td>
<td>0</td>
</tr>
</tbody>
</table>

**Picture 117: List of field names for a remote table**

In case of a view, double-click on the “Buffer Content” of type “View Definition” to display the definition of the view (see Picture 118) as stored in the remote database.

### Displaying the definition of view

**Picture 118: Displaying the definition of view**
Display the entire Remote Database Dictionary Buffer
In order to display the entire list of buffered remote database dictionary information, first press button Expert Mode to switch on the advanced options.

Then press button Display Buffer to list all buffered dictionary information as shown in Picture 119.

<table>
<thead>
<tr>
<th>Connection</th>
<th>Database Table Name</th>
<th>Buffer Type</th>
<th>Columns</th>
<th>Buffer Content / List of Columns</th>
<th>Last Update Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>SX</td>
<td>SAP/SX</td>
<td>Table Name</td>
<td></td>
<td></td>
<td>2010-02-19 10:02:16 UTC</td>
</tr>
<tr>
<td>SX</td>
<td>SAP/SX</td>
<td>Table Name</td>
<td></td>
<td></td>
<td>2010-02-19 10:02:16 UTC</td>
</tr>
<tr>
<td>SX</td>
<td>SAP/SX</td>
<td>Table Name</td>
<td></td>
<td></td>
<td>2010-02-19 10:02:16 UTC</td>
</tr>
<tr>
<td>ZTN_D65</td>
<td>APP</td>
<td>Table Name</td>
<td></td>
<td></td>
<td>2010-02-19 10:02:16 UTC</td>
</tr>
<tr>
<td>ZTN_D65</td>
<td>APP</td>
<td>Table Name</td>
<td></td>
<td></td>
<td>2010-02-19 10:02:16 UTC</td>
</tr>
<tr>
<td>ZTN_D65</td>
<td>APP</td>
<td>Table Name</td>
<td></td>
<td></td>
<td>2010-02-19 10:02:16 UTC</td>
</tr>
<tr>
<td>ZTN_D65</td>
<td>APP</td>
<td>Table Name</td>
<td></td>
<td></td>
<td>2010-02-19 10:02:16 UTC</td>
</tr>
<tr>
<td>ZTN_D65</td>
<td>APP</td>
<td>Table Name</td>
<td></td>
<td></td>
<td>2010-02-19 10:02:16 UTC</td>
</tr>
<tr>
<td>ZTN_D65</td>
<td>APP</td>
<td>Table Name</td>
<td></td>
<td></td>
<td>2010-02-19 10:02:16 UTC</td>
</tr>
<tr>
<td>ZTN_D65</td>
<td>APP</td>
<td>Table Name</td>
<td></td>
<td></td>
<td>2010-02-19 10:02:16 UTC</td>
</tr>
<tr>
<td>ZTN_D65</td>
<td>APP</td>
<td>Table Name</td>
<td></td>
<td></td>
<td>2010-02-19 10:02:16 UTC</td>
</tr>
<tr>
<td>ZTN_D65</td>
<td>APP</td>
<td>Table Name</td>
<td></td>
<td></td>
<td>2010-02-19 10:02:16 UTC</td>
</tr>
</tbody>
</table>

Picture 119: List of buffered entries

At the column “Last Update Timestamp” you can see when the dictionary buffer was updated with the actual remote dictionary information for the last time.

Clear the entire Remote Database Dictionary Buffer
If you are aware of dictionary changes in the remote database, that have not yet been updated in the dictionary buffer, you can press button Clear entire Buffer. This will clear the entire Remote Database Dictionary Buffer and each first request per dictionary object would go directly to the remote database to query the most recent dictionary information and update that into the buffer for subsequent requests.

Remove all outdated entries from the Remote Database Dictionary Buffer
If you just want to clear the remote database buffer for all outdated objects, that is information older than 24 hours, press button Remove all outdated entries. All newer information will stay inside the buffer. Removed buffer entries will force a new remote dictionary access upon next request anyway.
## 13.13 Table of figures

<table>
<thead>
<tr>
<th>Picture</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picture 1</td>
<td>Example data model</td>
<td>7</td>
</tr>
<tr>
<td>Picture 2</td>
<td>Example data</td>
<td>8</td>
</tr>
<tr>
<td>Picture 3</td>
<td>Maintaining connection details for the test environment</td>
<td>8</td>
</tr>
<tr>
<td>Picture 4</td>
<td>Selected example data</td>
<td>8</td>
</tr>
<tr>
<td>Picture 5</td>
<td>Selecting the Generic Table Entry Counter for Remote DB</td>
<td>9</td>
</tr>
<tr>
<td>Picture 6</td>
<td>Parameter entry for NATA BCNX</td>
<td>10</td>
</tr>
<tr>
<td>Picture 7</td>
<td>Defining display columns</td>
<td>10</td>
</tr>
<tr>
<td>Picture 8</td>
<td>Example parameters</td>
<td>11</td>
</tr>
<tr>
<td>Picture 9</td>
<td>Entering multiple values for a filter</td>
<td>11</td>
</tr>
<tr>
<td>Picture 10</td>
<td>Available key figures for monitoring object NATA BCNX</td>
<td>12</td>
</tr>
<tr>
<td>Picture 11</td>
<td>Detailcall headertext parameter</td>
<td>13</td>
</tr>
<tr>
<td>Picture 12</td>
<td>Thresholds for NATA BCNX</td>
<td>13</td>
</tr>
<tr>
<td>Picture 13</td>
<td>Alert display for &quot;Number of counted entries&quot;</td>
<td>14</td>
</tr>
<tr>
<td>Picture 14</td>
<td>Detail information for key figure “Number of Counted Entries”</td>
<td>15</td>
</tr>
<tr>
<td>Picture 15</td>
<td>Detail information for key figure “Number of Distinct Values”</td>
<td>15</td>
</tr>
<tr>
<td>Picture 16</td>
<td>Key figure Count simulation in Test Environment</td>
<td>17</td>
</tr>
<tr>
<td>Picture 17</td>
<td>Key figure Count simulation in Test Environment</td>
<td>18</td>
</tr>
<tr>
<td>Picture 18</td>
<td>Definition of Monitoring Object</td>
<td>19</td>
</tr>
<tr>
<td>Picture 19</td>
<td>Definition of Monitoring Object Header Attributes</td>
<td>19</td>
</tr>
<tr>
<td>Picture 20</td>
<td>Definition of Monitoring Object’s Key Figures</td>
<td>19</td>
</tr>
<tr>
<td>Picture 21</td>
<td>Definition of Key Figure Attributes</td>
<td>20</td>
</tr>
<tr>
<td>Picture 22</td>
<td>Definition of Key Figure Parameter</td>
<td>20</td>
</tr>
<tr>
<td>Picture 23</td>
<td>Definition of SQL Statement for Counting</td>
<td>21</td>
</tr>
<tr>
<td>Picture 24</td>
<td>Definition of SQL Statement for Detail Display</td>
<td>21</td>
</tr>
<tr>
<td>Picture 25</td>
<td>Definition of Column List for Detail Display</td>
<td>22</td>
</tr>
<tr>
<td>Picture 26</td>
<td>Selecting the monitoring object</td>
<td>27</td>
</tr>
<tr>
<td>Picture 27</td>
<td>List of key figures for TriplePoint Commodity SL Trade Monitoring</td>
<td>28</td>
</tr>
<tr>
<td>Picture 28</td>
<td>Example for a parameter set</td>
<td>28</td>
</tr>
<tr>
<td>Picture 29</td>
<td>Maintaining thresholds</td>
<td>28</td>
</tr>
<tr>
<td>Picture 30</td>
<td>Sample alert overview for all key figures</td>
<td>33</td>
</tr>
<tr>
<td>Picture 31</td>
<td>Sample alert display for key figure “EOD: Failed End-of-Day Process records”</td>
<td>33</td>
</tr>
<tr>
<td>Picture 32</td>
<td>Detail information for key figure “EOD: Failed End-of-Day Process records”</td>
<td>34</td>
</tr>
<tr>
<td>Picture 33</td>
<td>Interaction with external system via Web services</td>
<td>35</td>
</tr>
<tr>
<td>Picture 34</td>
<td>Starting the administration for the Web service</td>
<td>38</td>
</tr>
<tr>
<td>Picture 35</td>
<td>Configuration of the service end point</td>
<td>38</td>
</tr>
<tr>
<td>Picture 36</td>
<td>New node for the Monitoring Object NAWSPUSH</td>
<td>39</td>
</tr>
<tr>
<td>Picture 37</td>
<td>Available parameters for Object NAWSPUSH</td>
<td>39</td>
</tr>
<tr>
<td>Picture 38</td>
<td>Available key figures for the PUSH-NAWPUSH</td>
<td>40</td>
</tr>
<tr>
<td>Picture 39</td>
<td>Example for key figure “Status Keyfigure”</td>
<td>40</td>
</tr>
<tr>
<td>Picture 40</td>
<td>Example for key figure “2-Step-Rating”</td>
<td>40</td>
</tr>
<tr>
<td>Picture 41</td>
<td>Example for key figure “4-Step-Rating”</td>
<td>41</td>
</tr>
<tr>
<td>Picture 42</td>
<td>Example for an alert caused by pushing data into BPMon</td>
<td>41</td>
</tr>
<tr>
<td>Picture 43</td>
<td>The data model</td>
<td>43</td>
</tr>
<tr>
<td>Picture 44</td>
<td>Creating a consumer proxy</td>
<td>45</td>
</tr>
<tr>
<td>Picture 45</td>
<td>Creating a logical port</td>
<td>46</td>
</tr>
<tr>
<td>Picture 46</td>
<td>Available parameters for Object NAWSPULL</td>
<td>48</td>
</tr>
<tr>
<td>Picture 47</td>
<td>Available key figures for the PULL-interaction</td>
<td>48</td>
</tr>
<tr>
<td>Picture 48</td>
<td>Example for key figure “Status Keyfigure”</td>
<td>49</td>
</tr>
</tbody>
</table>
Picture 49: Example for key figure “2-Step-Rating” ................................................................. 49
Picture 50: Example for key figure “4-Step-Rating” ................................................................. 49
Picture 51: Example for an alert caused by pulling data from an external system ..................... 50
Picture 52: Detailed Display for the throughput measurement via Web page ................................ 53
Picture 53: The SMSY as returned from the external Web service ............................................ 54
Picture 54: Detail Display based on data returned by a Web Service ........................................... 54
Picture 55: The InfoObject used for the example .................................................................. 56
Picture 56: Content of the example’s InfoObject ................................................................ 56
Picture 57: Selecting data for display in transaction RSA1 ....................................................... 57
Picture 58: Defining output fields in transaction RSA1 ............................................................ 57
Picture 59: Data display in transaction RSA1 ......................................................................... 58
Picture 60: Selecting the monitoring object NABWICNX ....................................................... 59
Picture 61: Selection screen for NABWICNX .......................................................... ................. 59
Picture 62: Selecting characteristics for detail display ....................................................... 60
Picture 63: Defining a filter for NABWICNX ....................................................................... 60
Picture 64: Maintaining the header text for the detail display .................................................. 62
Picture 65: Maintaining thresholds for the key figure of monitoring object NABWICNX .......... 62
Picture 66: Alert display for key figure “Aggregate value of Key Figures” ............................. 63
Picture 67: Detail information for key figure “Aggregate value of Key Figures” ....................... 63
Picture 68: SAP Database Connection Layers ...................................................................... 65
Picture 69: Connection details in transaction DBACOCKPIT ................................................... 67
Picture 70: Authorization Object for a database connection ..................................................... 67
Picture 71: Example for a logical component ...................................................................... 69
Picture 72: Defining the system as a database instance ........................................................... 69
Picture 73: Link to the database ........................................................................................... 70
Picture 74: SMSY-Attributes to maintain the connection data .................................................. 70
Picture 75: A database connection entry ............................................................................. 70
Picture 76: Example for a logical component ...................................................................... 72
Picture 77: Defining the system as a database instance ........................................................... 72
Picture 78: Link to the database ........................................................................................... 73
Picture 79: LMDB-Attributes to maintain the connection data .................................................. 73
Picture 80: A database connection entry ............................................................................. 74
Picture 81: Accessing the WSDL for a Web service ............................................................... 76
Picture 82: A logical component ........................................................................................ 81
Picture 83: Defining a main instance for non-ABAP-monitoring ............................................... 81
Picture 84: The SMSY-parameters for the BW-connection ...................................................... 82
Picture 85: A logical component ........................................................................................ 83
Picture 86: Defining a technical system instance for non-ABAP-monitoring ............................ 83
Picture 87: The LMDB-parameters for the BW-connection ...................................................... 84
Picture 88: Main selection screen for test environment ........................................................... 87
Picture 89: Search help for table names ................................................................................. 88
Picture 90: Search help for table name .................................................................................. 88
Picture 91: Search help for field names .................................................................................. 88
Picture 92: Example for a filter ............................................................................................ 89
Picture 93: Example for display field selection ....................................................................... 90
Picture 94: Resulting display screen ..................................................................................... 90
Picture 95: Example for "Count number of entries" .................................................................. 91
Picture 96: Example for "Count number of distinct values" ................................................... 91
Picture 97: Example for “Get single field value” ................................................................... 92
Picture 98: Example for "Get single field value" with aggregation ........................................... 92
Picture 99: Example for display of table content .................................................................... 93