Advanced features that can help you customize your reports

This appendix includes the following topics:

About customizing your reports

Similar to predefined reports (Profile, Exceptions, and so on), customized reports can have property definitions set through the Properties dialog box. However, this is currently not supported by the Report Manager User Interface and must be implemented manually.

Retrieving over time information from the Performance Management Database may result in gaps (missing information). These are specific timestamps for which no data is available. To fill in these gaps, Report Manager FocalPoint supports holes filling. However, the current version of the Report Manager User Interface does not support the holes filling definition.

To implement these features, a few manual changes must be made to the tables and XML files of Report Manager FocalPoint. These changes should only be carried out by a person who is experienced with SQL and familiar with XML.

> These two features are supported by Report Manager FocalPoint, but are not supported by the Report Manager User Interface.

About defining properties for a customized report

Defining properties for a Customized report enables the user to rerun a report with its preferred property values. This procedure enables you to add new properties to the General, Filtering, and Time Frame tabs in the Properties dialog box.

After you complete this procedure, do not edit the Customized report by clicking Edit, because Report Manager will override the newly defined properties. If you have to edit the report, you will have to carry out this procedure again to restore these properties.

Defining new properties for a Customized report includes the following stages:

About creating a customized report

Set a filter condition that includes a fixed valid value for each new property, (the value should be distinguishable, so that you can easily recognize it later when you replace it with the new property name in the generated report's .xml file).

When you complete the filter condition settings for all the new properties, execute the customized report and then save it. Report Manager creates two files, the report's Microsoft Excel file and .xml file. See Creating customized reports.

About updating Report Manager tables in Performance Management Database

The following changes should be made to the Report Manager tables of the Performance Management Database, in order that the new properties you defined for your customized report can be supported:

> Check with your Precise administrator about the location of the Performance Management Database and user permission details.

About adding a new property to the Report Parameters table

Open the Report Parameters Table, named `PS_FSRA_REPORT_PARAMETERS`, using a database tool, such as SQL*Plus or Microsoft SQL Server Enterprise Manager (depending on your Performance Management Database type).

Add the new property to the Report Parameters Table as described in the following table. Verify that the property identifier (`FSRA_ID`) and name (`FSRA_NAME`) are unique in this table.

The following table describes the properties that can be added to the Report Parameters table.

Table 1 Additional properties that can be added to the Report Parameters table

Column	Description
FSRA_ID	Specifies the property identifier. Set a unique and consecutive number, starting from 100000 (enables you to easily distinguish between the new properties you added and the existing pre-defined properties).
FSRA_NAME	Specifies a unique property name.
FSRA_IS_DISPLAYABLE	Specifies whether or not to display the property through the Properties dialog box. Valid values are: • T - True • F - False Set this value to 'T.'
FSRA_IS_CONVERTIBLE	Specifies whether or not to convert the % (wildcard sign) to * in the Properties dialog box. The valid values include: • T - True • F - False If the property value contains wildcards, set this value to 'T.'
FSRA_DEFAULT_VALUE	Specifies the default value of the property. Set this value to NULL, since you will later define the default value when you associate the new property to the customized report.
FSRA_PROMPT_TEXT	Specifies the text that is displayed with the property box in the Properties dialog box. Enter a description.
FSRA_TAB_ID	Specifies the tab ID - of the report Properties dialog box - in which this property is displayed. Set to one of the following valid values according to your property: • G - General • F - Filters • T - Time Frame
FSRA_INDEX	Specifies the property index in the tab, indicating its order.

If you use a database tool such as Microsoft SQL Server Enterprise Manager, you can use the following INSERT statement as an example, to add a new parameter to the PS_FSRA_REPORT_PARAMETERS table.

```
INSERT INTO PS_FSRA_REPORT_PARAMETERS
(FSRA_ID, FSRA_NAME, FSRA_IS_DISPLAYABLE,
FSRA_IS_CONVERTABLE, FSRA_DEFAULT_VALUE,
FSRA_PROMPT_TEXT, FSRA_TAB_ID, FSRA_INDEX)
VALUES (100000, 'ProgramFilter', 'T', 'T', NULL, '
Analyze only the following programs:', 'F', 10)
```

About locating the report identifier

Open the Reports Table (named PS_FSRP_REPORTS) and search for the Report Identifier of the new report that you created in the FSRP_ID column. You can search for it according to the FSRP_NAME column, which contains the report name you specified when saving the report, or according to the FSRP_GENERATOR_NAME column that contains the string ‘excel’.

About associating a new property to the customized report

To associate the new property to the Customized report, set the property values in the Parameter Values Table (named PS_FSPV_PARAMETER_VALUES).

The following table describes the parameters values which can be customized.

Table 2 Parameter Values

Column	Description
FSPV_FSRA_ID	Specifies the property identifier. Set the number as you specified in the FSRA_ID column of the Report Parameters Table. See About adding a new property to the Report Parameters table.

FSPV_FSPG_ID	Specifies the group identifier. Set to NULL.
FSPV_FSRP_ID	Specifies the report identifier. Set the number that you found in the Reports Table. See About locating the report identifier.

FSPV_CTX_ID	Specifies the context identifier.
	
	To find the context identifier of the new property, run the following query:
	
	`SELECT FSPV_CTX_ID`
	`FROM PS_FSPV_PARAMETER_VALUES`
	`WHERE (FSPV_FSRP_ID = report_id) AND`
	`(FSPV_FSRA_ID = parameter_id)`
	
	Where `report_id` is the report identifier that you found in the Reports Table and `parameter_id` is the report parameter that you added in the Report Parameters Table. See About adding a new property to the Report Parameters table.
	
	The query result includes the required value for `FSPV_CTX_ID`. Set the resulted value accordingly.

FSPV_VALUE	Specifies the default value of the new property.
FSPV_PROMPT_TEXT	Specifies the text that is displayed with the property box in the Properties dialog box. You can set to NULL to apply the text that you already set in the `FSRA_PROMPT_TEXT` column of the Report Parameters Table, or you enter a new text to override it. See About adding a new property to the Report Parameters table.

If you use a database tool such as Microsoft SQL Server Enterprise Manager, you can use the following example to add a new parameter to the `PS_FSPV_PARAMETER_VALUES` Table:

```
INSERT INTO PS_FSPV_PARAMETER_VALUES
(FSPV_FSRA_ID, FSPV_FSPG_ID, FSPV_FSRP_ID,
FSPV_CTX_ID, FSPV_VALUE, FSPV_PROMPT_TEXT) VALUES
(100000, NULL, 100086, 10, 'Scheduler', NULL)
```

Updating the .xml file with the report

When you create a Customized report, Report Manager creates two files: a Microsoft Excel file that contains the report details, and a .xml file that contains the report definition. This procedure provides step-by-step instructions for updating the definitions of the query tag in the .xml file. See About creating a customized report and About adding a new property to the Report Parameters table.

The following example shows the query tag before modification:

```
<query updated_manually="false">SELECT SQPN_STRING_VALUE "SQProgram"
FROM (SELECT SQSS_PROGRAM_ID FROM PW_SQSS_SESSIONS_STATS_D,
PS_INAP_APP_TIER, PW_SQPN_PROGRAM_NAMES_N, PS_INCE_INSTANCE,
 PS_INEN_ENVIRONMENT

WHERE SQSS_PWII_INSTANCE_ID=INCE_ID AND INCE_INAP_ID=INAP_ID AND
INAP_INEN_ID=INEN_ID AND SQSS_PROGRAM_ID = SQPN_ID AND SQPN_ID
<>0 AND SQPN_STRING_VALUE LIKE 'My Program' AND INEN_ID = 10
GROUP BY SQPN_STRING_VALUE, SQSS_PROGRAM_ID) intab,
PW_SQPN_PROGRAM_NAMES_N WHERE SQSS_PROGRAM_ID = SQPN_ID AND SQPN_ID
<>0</query>
```

The following example shows the query tag after modification:

```
<query updated_manually="true">SELECT SQPN_STRING_VALUE "SQProgram"
FROM (SELECT SQSS_PROGRAM_ID FROM PW_SQSS_SESSIONS_STATS_D,
PS_INAP_APP_TIER, PW_SQPN_PROGRAM_NAMES_N, PS_INCE_INSTANCE,
PS_INEN_ENVIRONMENT

WHERE SQSS_PWII_INSTANCE_ID=INCE_ID AND INCE_INAP_ID=INAP_ID AND
INAP_INEN_ID=INEN_ID AND SQSS_PROGRAM_ID = SQPN_ID AND SQPN_ID
<>0 AND SQPN_STRING_VALUE LIKE '$P{ProgramFilter}' AND INEN_ID = 10
GROUP BY SQPN_STRING_VALUE, SQSS_PROGRAM_ID) intab,
PW_SQPN_PROGRAM_NAMES_N WHERE SQSS_PROGRAM_ID = SQPN_ID AND SQPN_ID
<>0</query>
```

To update the report's .xml file

1. Open the .xml file of your Customized report from the following folder in the server where Report Manager FocalPoint is installed:
   <precise_root>`/products/foresight/etc/reports/udr/storage`

The file name format consists of the Context Identifier (FSPV_CTX_ID) concatenated with the Report Identifier (FSRP_ID). For example, in the 10_100086.xml Customized report file, the context identifier is 10 and the report identifier is 100086 (you can also check the file's modification date to identify your report's .xml file).

2. In the .xml file, search for the fixed value that you have specified in the filter conditions when creating the report.
   The fixed value can be found in both the filter and query tags. Update the query tag as follows:
   - Change the value of the updated_manually attribute to true.
   - Replace the fixed value of the filter condition with the string '$P{parameter_name}', where parameter_name is the new property name (FSRA_NAME) that you defined in the Report Parameters Table.
3. Save the .xml file.
4. Click **Scheduling** or **View** followed by Rerun to rerun the report.
   Do not rerun the report by clicking **Edit**.

Filling in gaps in a customized report

Retrieving information collected over time from the Performance Management Database may result in missing information. These are specific timestamps for which no data is available. For example, an Oracle program may not include data for a specific timestamp, if the program was not invoked during the interval that started with this timestamp. In this case, you may want, for example, to fill in the number of executions counter with a zero value.

To fill in gaps in a customized report

1. Create a customized report
2. Modify the report's .xml file

After filling in gaps in a customized report, don't click **Edit** to modify the report. If after filling in gaps in a customized report you mistakenly click **Edit**, Report Manager will override your modifications. If you must edit the Customized report, fill in the gaps again so as to restore the properties you modified.

About creating a customized report

The selected data fields in the report must contain one or more identifiers and a single timestamp data field, which must be a full timestamp data field and not a partial time data field, such as Date and Month.

You should also sort the identifiers followed by the timestamp. The timestamp must be sorted in ascending order, while the identifiers can be sorted in ascending or descending order.

After completing report settings, execute the customized report and then save it. Report Manager creates two files, the report's Microsoft Excel file and .xml file. See Creating customized reports.

Modifying the .xml file with the report

When you create a Customized report, Report Manager creates two files: a Microsoft Excel file that contains the report details, and a .xml file that contains the report definition.

The following example shows the report's .xml file before modification. The bold text indicates the value that should be replaced.

```
<report>
     <subreport name="default_query" sum_level="Daily" id="47">
          <selection>
               <object id="4000" name="Instance" />
               <object id="4003" name="User" />
               <object id="21" name="Timestamp" />
               <object id="4181" name="Sessions" />
          </selection>
          <order>
               <object id="4000" name="Instance" order="asc" />
               <object id="4003" name="User" order="asc" />
               <object id="21" name="Timestamp" order="asc" />
          </order>
          <filter />
          <query firstLoad="true" updated_manually="false">SELECT intab.c2
          "SQInstance", SQUN_STRING_VALUE "SQUser", intab.c3 "Timestamp",
          intab.c0 "SQSessions" FROM (SELECT SQSS_TIMESTAMP c3, SQSS_USER_ID ,
          SUM(SQSS_NUM_OF_ENDED_SESSIONS_SUM) c0, INCE_NAME c2
          FROM PW_SQSS_SESSIONS_STATS_D, PS_INAP_APP_TIER, PS_INCE_INSTANCE,
          PS_INEN_ENVIRONMENT WHERE SQSS_PWII_INSTANCE_ID=INCE_ID AND
          INCE_INAP_ID=INAP_ID AND INAP_INEN_ID=INEN_ID AND
          SQSS_PWII_INSTANCE_ID=INCE_ID AND INCE_INTE_CODE='SQ' AND
          INCE_DELETED='F' AND INEN_ID = 10 GROUP BY INCE_NAME,
          SQSS_USER_ID, SQSS_TIMESTAMP) intab, PW_SQUN_USER_NAMES_N
          WHERE SQSS_USER_ID = SQUN_ID AND SQUN_ID <>0 ORDER BY
          "SQInstance" asc, "SQUser" asc, "Timestamp" asc
          </query>
     </subreport>
</report>
```

The following example shows the report's .xml file after modification:

```
<report>
     <subreport name="default_query" sum_level="Daily" id="47">
          <selection>
               <object id="4000" name="Instance" />
               <object id="4003" name="User" />
               <object id="21" name="Timestamp" />
               <object id="4181" name="Sessions" />
          </selection>
          <order>
               <object id="4000" name="Instance" order="asc" />
               <object id="4003" name="User" order="asc" />
               <object id="21" name="Timestamp" order="asc" />
          </order>
          <filter />
          <query firstLoad="true" updated_manually="true">SELECT intab.c2
          "SQInstance", SQUN_STRING_VALUE "SQUser", intab.c3 "Timestamp",
          intab.c0 "SQSessions"
          FROM (SELECT SQSS_TIMESTAMP c3, SQSS_USER_ID ,
          SUM(SQSS_NUM_OF_ENDED_SESSIONS_SUM)
          c0, INCE_NAME c2 FROM PW_SQSS_SESSIONS_STATS_D, PS_INAP_APP_TIER,
          PS_INCE_INSTANCE, PS_INEN_ENVIRONMENT WHERE
          SQSS_PWII_INSTANCE_ID=INCE_ID AND
          INCE_INAP_ID=INAP_ID AND INAP_INEN_ID=INEN_ID AND
          SQSS_PWII_INSTANCE_ID=INCE_ID AND INCE_INTE_CODE='SQ' AND
          INCE_DELETED='F' AND INEN_ID = 10 GROUP BY INCE_NAME, SQSS_USER_ID,
          SQSS_TIMESTAMP) intab, PW_SQUN_USER_NAMES_N
          WHERE SQSS_USER_ID = SQUN_ID AND SQUN_ID
          <>0 ORDER BY "SQInstance" asc, "SQUser" asc, "Timestamp" asc
          </query>
          <fill_holes fill="true">
               <object id="4181" name="Sessions" fill="const" const="666" />
               <timeframe start="2004-05-20 00:15:23.11" end="2004-06-10 22:35:23.11" />
          </fill_holes>
     </subreport>
</report>
```

This procedure provides step-by-step instructions for updating the report's .xml file, including updating the query tag and setting the `fill_holes` tag.

To modify the report's .xml file

1. Open the .xml file of your Customized report from the following folder in the server, where Report Manager FocalPoint is installed:
   `< precise_root>/products/foresight/etc/reports/udr/storage`
   The file name format consists of the Context Identifier (`FSPV_CTX_ID`) concatenated with the Report Identifier (`FSRP_ID`).
   For example, in the 10_100086.xml file, the context identifier is 10 and the report identifier is 100086. (You can also check the file's modification date to identify your report's .xml file.)
2. In the .xml file, do the following:
   - Search for the query tag.
   - Change the value of the updated_manually attribute to True.
   - Add the `fill_holes` tag after the query tag at the same level.
3. Set the fill attribute of the `fill_holes` tag to True to enable the data completion feature.
4. Add the object tag under the `fill_holes` tag level and set its attributes.
   For example, `<object id="4000" name="Instance" fill="const" const="666"/>`
   - **id**. Specifies the object identifier.
     Set to one of the objects ID defined in the selection tag at the beginning of the report's .xml file.
   - **name**. Specifies the data field name.
     This attribute is optional. You may set this attribute to add clarity to the report.
   - **fill**. Specifies the method of data completion.
     Possible values include:
     - **const**. The data is completed with a constant value. In this case, you must also set the const attribute.
     - **prev**. The data is completed with the value of the previous timestamp.
     - **next**. The data is completed with the value of the next timestamp.
   - **const**. Specifies the constant value that completes the missing data.
     Set the const attribute if you set the fill attribute to const.
     By default, Report Manager fills in counters with zeros, and strings with nulls. You can set the const attribute, together with the prev or next fill types, to specify a different value other than the default value (zero or null), so that Report Manager also completes the margins, where there is no previous or next value.
5. Add the timeframe tag under the fill_holes tag level and set its attributes.
   The timeframe tag defines the time frame for which the data completion applies. You can set the time frame using one of the following methods:
   - **Absolute date range**. Set the start and end attributes. For example, `<timeframe start="2004-05-20 00:15:23.11" end="2004-06-10 22:35:23.11" />`.
   - **Relative date time**. Set the lastNType, lastNStart, and lastNLen attributes.
     - **start**. Specifies the start date and time.
       The format is: yyyy-mm-dd hh24:mi:ss.ms
       You must apply the entire format, except the ms (milliseconds), which is optional. For example:
       `<timeframe start="2004-05-15 18:30:15.00" end="2004-05-20 23:59:59.59"/>`
     - **end**. Specifies the end date and time. See the format of the start attribute.
     - **lastNType**. Specifies the time units for the lastNStart and lastNLen attributes.
       Possible values include:

        - H - Hours
        - D - Days
        - W - Weeks
        - M - Months

          Set a value that is not lower than the report summary level resolution (that is, the report summary level must be at least as the lastNType value).

          For example: lastNType="D" can be used with a report summary level of H (hours).
    - **lastNStart**. Specifies the start of the time frame counting back from the current time (current time is always 0).

      For example, if lastNType="D", then a lastNStart="0" signifies that the start time of the time frame is today.

      If lastNStart="10" and lastNLen = "5", this signifies that the start time of the time frame was 15 days ago.
    - **lastNLen**. Specifies the length of the time frame counting back from the lastNStart time.

      The following example defines a start time of 15 days ago and end time of 10 days ago.

      ```
      <timeframe lastNType="D" lastNStart="10" lastNLen="5"/>
      ```
6. Save the .xml file.
7. Rerun the report by doing one of the following:
    a. Click **Scheduling**.
    b. Click **View** and then **Rerun**.

       Do not click **Edit** to rerun the report

About importing and exporting customized reports

The following section describes how to export a customized report from one Precise system and import it to another Precise system.

> Before you begin, find the source Tier ID and the destination Tier ID in the `PS_INAP_APP_TIER` or `PS_FSEC_ENV_CONTENTS` tables.

To export/import a customized report

1. In the source Precise system, create the ForesightExportRequest.xml file in the Precise root folder. For example:
   ```
   <root name="Export">
        <report-name>In Sybase Breakdown</report-name>
        <apptier>1201</apptier>
   </root>
   ```

   > The report-name value is case-sensitive.

2. Run the following command:
   ```
   infra\bin\psin_http_requestor -dp -x fs-export-import FS < ForesightExportRequest.xml > out.xml
   ```
3. Check the out.xml file in the <*precise_root*>, to view the status.
4. Copy the exported report's .zip file from the source Precise root folder to the destination Precise installation in:
   <*precise_root*>`\distribution`

   > The .zip file name consists of `<technology code>_<report name>` (SY_In Sybase Breakdown.zip). For cross-Tiers the Tech code is ALL (ALL_CA_Rep.zip).

5. In the destination Precise system, create the ForesightImportRequest.xml file in the Precise root folder. For example:
   ```
   <root name="Import">
        <report-name>SY_In Sybase Breakdown.zip</report-name>
        <apptier>1174</apptier>
   </root>
   ```
6. Run the following command:
   ```
   infra\bin\psin_http_requestor -dp -x fs-export-import FS < ForesightImportRequest.xml > out.xml
   ```
7. Check the out.xml file in the Precise root folder, to view the status.

About copying customized reports

The following section describes how to copy customized reports from one Tier to another with the CLI utility. To copy a customized report

1. Run the following commands from the Precise root folder:
    - **Windows**
        - ```
          infra\bin\psin_cli.bat -i3-user <user> {-i3-clear-password <i3_clear_password> |
          -i3-encrypted-password <i3_encrypted_password>}
          -action fs-copy-foresight-custom-report
          -parametersfile <parameters-file.xml>
          Example:
          infra\bin\psin_cli.bat -i3-user admin -i3-clear-password admin -action fs-copy-foresight-
          custom-report -parametersfile my_parameters_file.xml
          ```
    - **UNIX**
          ```

- infra/bin/psin_cli.sh -i3-user <user> {-i3-clear-password <i3_clear_password> | -i3-encrypted-password <i3_encrypted_password>} -action fs-copy-foresight-custom-report -parametersfile <parameters-file.xml>

2. Create the parameters-file.xml file in the Precise root folder and verify that it contains the following:

```
<parameters>
     <parameter name="source-environment" value="Default"/>
     <parameter name="source-apptier" value="SQL Server"/>
     <parameter name="source-report" value="My_report"/>
     <parameter name="destination-environment" value="Default"/>
     <parameter name="destination-apptier" value="SQL Server"/>
</parameters>
```

and verify that the information in the following notes is applied.

The parameter values are case-sensitive.

The "source-report" name must not contain blanks.

The new report name will have a suffix with a serial number (for example: "My_report_1").

To copy a Cross-Tier report, you should use the "Cross-Tier" string for both "source-apptier" and "destination-apptier" parameters.