ARIS Process Performance Manager

QUERY INTERFACE

Version 10.2

April 2018

This document applies to PPM Version 10.2 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Contents

1 Text conventions

Menu items, file names, etc. are indicated in texts as follows:

- Menu items, key combinations, dialogs, file names, entries, etc. are displayed in **bold**.
- User-defined entries are shown in **<bold and in angle brackets>**.
- Single-line example texts (for example, a long directory path that covers several lines) are separated by ↵ at the end of the line.
- File extracts are shown in this font format:

  ```
  This paragraph contains a file extract.
  ```

2 General

The manual describes how to use the PPM Query interface.

Please note that this manual is not intended to replace user or customizing training. It is a source of reference containing information that supplements the information provided in the online help.

3 Query interface (query API)

A programmable interface (API) can be used to automatically query and process analysis results and measured values from PPM in other applications. Tabular data in the form of list tables or process instance lists from PPM is queried. Charts or EPC views cannot be evaluated using the query interface.

Like the Report component, the data determined using the query interface is based on favorites. The PPM user interface is first used to create a query with a table view and the required filtering and save it as a favorite.

A query can be executed directly by entering a URL in the browser using the HTTP protocol. During the call, the favorites path is transferred as a parameter.

The result of the query is returned as an XML data flow. This contains a **Row** XML element for each row and a corresponding sub-element for each measure or dimension column. The result is not localized. Time stamps and numerical values have fixed formats.

Alternatively, you can display the result in a formatted HTML table.

See also chapter **Output formats** (Page 9).

4 Submitting a query

The data is queried by entering a favorite. As an option, you can also specify filters to further restrict the data volume.

Queries can be submitted in two different ways - using **HTTP request** (**GET** and **POST** http protocols) or the **SOAP** service.

HTTP PATTERN

`http://host:<load balancer port>/ppm/contextname/urlquery/query.do`

Example

http://localhost:4080/ppm/API_umg_en/urlquery/query.do?favoritename=<favorite name>

SOAP PATTERN

`http://host:port/ppm/contextname/services/queryApiService`

This interface is provided for the implementation of client applications in Java, PHP, etc. It is also easy to embed it in service-oriented architectures (SOA).

4.1 Query parameters

Regardless of whether the query is submitted using HTTP Request or the SOAP service, the query interface supports the following parameters.

Parameter	Description
favoritename	Name of the favorite including path, for example, My Favorites/Test favorite
favoritefolder	Defines whether the favorite specified using favoritename is a private or shared favorite. Valid values: **FAVORITES_SHARED** **FAVORITES_PRIVATE**. Default value: **FAVORITES_SHARED**
language (optional)	The language in which the data is to be supplied, for example, **EN**. Unless specified, data is output in the language in which the PPM server was started.
Filter list in the form of key value pairs (optional)	Specifies additional filters (key value pairs), linked to the filters set in the favorite with AND. For example: **TIME=2008**

Parameter	Description
filterfavoritename (optional)	The filters set in the specified favorite are linked with the specified filter list with AND.
filterfavoritefolder (optional)	Defines whether the favorite specified using **filterfavorite** is a private or shared favorite. Valid values: **FAVORITES_SHARED** **FAVORITES_PRIVATE**. Default value: **FAVORITES_SHARED**

4.2 Supported favorite types

The query interface only supports favorites for which the output values are displayed as a table. If other favorites are queried, a corresponding error is returned.

LIST TABLE

With the **List table** view type, the data is output in a similar way to its representation in the analysis in PPM.

- The number and sequence of the columns is identical.
- The sequence of the rows is identical.
- The color highlighting of the table cell corresponding to the planned or threshold values, in which the measure is displayed in the analysis, is displayed as text.
- Depending on the query settings for the favorite, planned values are shown by highlighting only, in a separate value column only or in both ways. By default, they are displayed in both ways.

The highlighting resulting from a top-flop analysis (different intensities of red and green) is not displayed.

CROSSTAB

A crosstab set as the view type for the query is displayed as a list table.

INSTANCE LISTS

An instance list is displayed as a list table. The instance lists include:

- Process instance search
- Alarm value deviations
- Early alert system
- Outlier analysis

- Process instance list

Additional information, such as that displayed in the analysis with the **Alarm value deviations** view type (process type, alarm value), is not displayed by the query interface.

If multiple instance lists are shown on individual tabs in a query, only the table for the active tab is returned in the foreground. The active tab is saved when creating the favorite.

EPCS

If the favorite is displayed in the **EPC** view, the process instance list of the EPC is returned as if the process instance list view type had been set.

CHARTS

If the favorite is displayed in the **Chart** view, the process instance list of the EPC is returned as if the **List table** view type had been set. This includes:

- All default charts
- Flow chart
- Assessment chart

COMMUNICATION NETWORK AND RELATION EXPLORER

If the favorite is displayed in the **Communication network** or **Relation Explorer** view, the process instance list of the EPC is returned as if the **List table** view type had been set.

4.3 Set filter

Filters can be set for dimensions and measures. The individual filters are specified as key value pairs.

4.4 URL query

For a URL query, the URL is composed using the following pattern:

Query pattern

`http://host:port/kontextname/urlquery/query.do?parameterlist`

The context name is specified when installing PPM and the suggested name is composed as follows: **API_<client name>**, for example, **API_umg_en**

The URL is extended by particular parameters, which are entered at the end of the URL, separated by a question mark. The parameters (including value) are separated by the **&** symbol. The name and value are separated by the **equals** sign.

?<paramname_1>=<paramvalue_1>&<paramnameN=paramvalueN.

The possible parameters are described in the chapter on **Query parameters** (Page 4).

Example

```
http://pcwas:8080/API_default/urlquery/query.do?favoritename=02%20Verteilung%20d
er%20Durchlaufzeit&favoritefolder=FAVORITES_PRIVATE&language=de
```

If you have enabled the query interface, in PPM you can use the **URL for query interface** option in the pop-up menu for a particular favorite to copy the corresponding URL to the clipboard. The URL created is already correctly coded and contains the language specified when logging into the PPM front-end in the **language** argument. Additional filter settings specified for the favorite are not included.

4.4.1 Coding

By default, the coding of the query interface Web application is set to **UTF-8**. To use a different coding, enter the name of the coding you want to use in the **encoding** URL parameter, for example, **encoding=ISO-8859-1**.

There are various options for the URL coding on the client side.

Within Java programs, you can use the encode method (parameter value, encoding) from the **URLEncoder** class, and within JavaScript you can use the standard encodeURIComponent(parameter value) method.

4.4.2 Authentication

If a query is submitted via the browser, a dialog box opens in which users log into PPM with their **User name** and **Password**. The data is only extracted from PPM if the correct user ID is entered.

Only data for which the logged in user has the corresponding privileges is returned.

4.4.3 Logout

After the user has been authenticated once via the browser, login is not required for further queries. By default, the session length is set to 5 minutes. If no query is submitted in this time, the login must be repeated for another query.

To retrieve data for different users within the session duration, you can exit the active session early by logging out. To do this, enter the **logout.do** page in the URL. This page has no parameters. Then log in using different user access data.

Example

http://pcwas:8080/API_default/urlquery/logout.do

4.5 SOAP Web service

For calls via the SOAP Web service, the URL must be structured as follows:

URL pattern

http://host:<load balancer port>/ppm/containername/services/queryApiService

Example

http://pcwas:4080/ppm/API_umg_en/services/queryApiService

The **getData** method in the Web service can be used to query the data. The method has the following syntax:

getData(

String favoritename,
String favoritefolder,
String "Filter list in the form of key value pairs",
String filterfavorite,
String filterfavoritefolder
String language

)

The semantics of the parameters correspond to the parameters described in the chapter on **Query parameters** (Page 4).

The sequence of the query depends on the framework used.

4.5.1 WSDL

WSDL (Web service description language) describes the methods of a Web service.

The query interface supports the **getData** method with the parameters described in the chapter on **Query parameters** (Page 4).

You can determine the description of the Web service as follows.

`http://pcwas:8080/ppm/API_umg_en/services/queryApiService?wsdl`

The call can be made using generated stubs. Java classes are generated using WSDL, which can be used to access the Web service directly. In addition, the call can also be made using special frameworks, for example, axis.

4.5.2 Authentication

The Web service is authenticated using the same mechanism as the URL query. Unlike the http query, the Web service servlet does not use a session.

If the data is queried using the Web service, for example, via a Java client, the user name and password for each call must be specified in the http header. If the login fails, the Web service responds with the error code 401. This error code is evaluated by the client framework used.

4.5.3 Coding

The Web service always interprets the transferred character strings in UTF-8 format. For coding, within Java programs, you can use the encode method (parameter value, encoding) from the **URLEncoder** class, and within JavaScript you can use the standard encodeURlComponent(parameter value) method.

5 Output formats

5.1 XML data stream

The query interface returns the data in XML format.

The root element is called **queryresult** and contains the two child elements **header** and **data**.

THE HEADER XML ELEMENT

This XML element has further child elements containing the following meta and data structure information:

XML element	Description
query	**PCDATA** specifies the name of the favorite. The **Type** attribute contains the type of the favorite, **PRIVATE** or **SHARED**. The **Title** attribute contains the title of the favorite.
session	The **Client**, **User** and **Language** attributes contain the corresponding data used for the login.
timestamp	**PCDATA** specifies the execution time of the query to the second.
filter	**PCDATA** outputs all filters contained in the URL in URL notation. Each filter is described in more detail in a separate **item** child element.
Child element **item**	**PCDATA** contains the filter as a key value pair. The **Key** attribute specifies the key and the **Value** attribute the value of the filter.
column	Each element describes an individual data column. **PCDATA** contains the interface name of the measure or dimension. The **Name** attribute specifies the internal name and the **Datatype** attribute the PPM data type of the values. The **Usage** attribute specifies whether the data column contains values for a dimension (**ITERATION**) or a measure (**KEYINDICATOR**).

THE DATA XML ELEMENT

For each data row returned, a **row** XML element is output and further child elements contain the dimension values (**dc** XML element) and measure values (**kc** XML element) for the corresponding columns.

XML element	Description
dc	**PCDATA** and the **Value** attribute contain the dimension value for this data column. The **Name** attribute specifies the internal name of the dimension, corresponding to the **Name** attribute of a **Column** element. The **Type** attribute specifies the data type, which may differ from the PPM data type specified in the corresponding **Column** element. The **Value2** element specifies the scaling for numerical values, the descriptions - separated by **** for multi-level dimensions - for text dimensions and the format string for times.
kc	**PCDATA** and the **Vvalue** attribute contain the measure value for this data column. The **Name** attribute indicates the internal name of the measure, corresponding to the **Name** attribute of a **Column** element. The **value2** attribute (optional) specifies the scaling of the measure value. The zone attribute (optional) specifies the planned value range for the measure value, valid values: **_GREEN**, **_YELLOW** (adverse deviation), **_RED** (critical deviation)

Example

If you enter the URL **http://<Basic URL>/API_demo41_en/urlquery/query.do?favoritename=%5CMVs%5CHelpdesk%5CFavorite_Table&WERKS=3000&language=en&TIME=2007** for the demo database in Internet Explorer, the following result is obtained:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE queryresult (View Source for full doctype...)>
- <queryresult>
  - <header>
      <query type="SHARED" title="30 - Response time to customer speedometer">\MVs\Helpdesk\Favorite_Table</query>
      <session client="demo41_en" user="SYSTEM" language="en" />
      <timestamp>2008-09-02T07:56:16</timestamp>
    - <filter>
        TIME=2007&WERKS=3000
        <item key="TIME" value="2007">TIME=2007</item>
        <item key="WERKS" value="3000">WERKS=3000</item>
      </filter>
      <column name="D_PRIORITY" datatype="TEXT" usage="ITERATION">Priority</column>
      <column name="PNUM" datatype="LONG" usage="KEYINDICATOR">Number of processes</column>
    </header>
  - <data>
    - <row>
        <dc name="D_PRIORITY" type="TEXT" value="Very urgent" value2="">Very urgent</dc>
        <kc name="PNUM" value="54">54</kc>
      </row>
    - <row>
        <dc name="D_PRIORITY" type="TEXT" value="Urgent" value2="">Urgent</dc>
        <kc name="PNUM" value="164">164</kc>
      </row>
    - <row>
        <dc name="D_PRIORITY" type="TEXT" value="somewhat urgent" value2="">somewhat urgent</dc>
        <kc name="PNUM" value="959">959</kc>
      </row>
    - <row>
        <dc name="D_PRIORITY" type="TEXT" value="Not urgent" value2="">Not urgent</dc>
        <kc name="PNUM" value="677">677</kc>
      </row>
    - <row>
        <dc name="D_PRIORITY" type="TEXT" value="Not maintained" value2="">Not maintained</dc>
        <kc name="PNUM" value="4">4</kc>
      </row>
    </data>
</queryresult>
```

ERROR MESSAGES

If a query cannot be executed correctly, an **error** XML element is output instead of the **data** XML element. The text of the error is specified in PCDATA and the identifier in the **id** attribute.

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE queryresult (View Source for full doctype...)>
- <queryresult>
  + <header>
    <error id="0">Error determining filter values. Details:
      Unknown "TIMES" dimension/KPI.</error>
  </queryresult>
```

5.1.1 Query favorites

You can query a list of favorites from the PPM server. The list is returned as an XML file containing the following information for each exported favorite:

- Name of favorite
- Favorites path
- Full path (favorites path and name)
- URL-coded full path
- Query type
- Type of favorite

Only favorites and favorites folders in the currently queried folder are included. Favorites and favorites folders in subfolders are not included. A query returns all favorites based on any diagram type, EPC view, communication analysis, or Relation Explorer analysis.

An XML data stream is returned with the following structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE favoritequeryresult SYSTEM
"http://localhost:8080/API_umg_en/dtd/queryapifavorites.dtd">
<favoritequeryresult>
  <data>
    <favoritenode favoritetype="FOLDER" name="..." path="..."↵
                full-path="..." urlencoded-path="..."/>
    ...
    <favoritenode favoritetype="FAVORITE" name="..." querytype="..."↵
                path="..." full-path="..." urlencoded-path="..."/>
    ...
  </data>
</favoritequeryresult>
```

The information is specified in the form of XML attribute values.

XML attribute	Description
path	Path of the favorite. It is specified without masking characters.
name	Name of the favorite or favorites folder. It is specified without masking characters.
full-path	Full path of the favorite that also contains the favorite's name. The characters **/** or **** are used as separators between the path elements. Within the path element names, the characters **/** and **** are masked, that is, **/** is output as **/ /** and **** is output as **\ **.
urlencoded-path	Path and name of the favorite. Both are masked (see **full-path**) and URL-coded, that is, special characters are replaced by particular character sequences (for example, **** is replaced by **%5C**).
querytype	Query type of the favorite. Valid values: **CHART**: Favorite based on charts (2D, 3D, assessment, Gantt) **GRAPH**: Favorite based on EPC view, communication analysis, and Relation Explorer **TABLE**: Favorite based on tables and Process mining. **INSTANCELIST**: Favorites based on instance list, alarm value analysis, process instance search, outlier analysis, and Early alert system. Is only output if the result is a favorite.
favoritetype	The result is a favorites folder (value **FOLDER**) or a favorite (value **FAVORITE**).

The query considers only normal favorites and configuration favorites. Template and multi-analysis favorites are not included.

QUERY PATTERN

`http://host:port/contextname/urlquery/favorites/query.do?parameterliste`

Example

http://localhost:8080/API_demo41_en/urlquery/favorites/query.do?path=<favorites path>

If you have enabled the query interface, you can use in PPM the **URL for query interface** option in the pop-up menu of a particular favorites folder to copy the corresponding URL to the clipboard.

The URL created is already correctly coded and contains the language specified when logging into the PPM front-end in the **language** argument.

The following optional parameters are supported:

Parameter	Description
path (optional)	Path of the start node favorites are searched from. Depending on the **favoritefolder** value, all private or public favorites are searched if this information is missing.
querytype (optional)	Query type. Valid values: **CHART**: Favorites based on charts (2D, 3D, assessment, Gantt) **GRAPH**: Favorites based on EPC view, communication analysis, and Relation Explorer **TABLE**: Favorites based on tables and Process mining. **INSTANCELIST**: Favorites based on instance list, alarm value analysis, process instance search, outlier analysis, and Early alert system. If **querytype** is not specified, all query types are returned. You can search for multiple query types by specifying a semicolon-separated list of query types.
favoritetype (optional)	Specifies if only favorites folders (value **FOLDER**) or only favorites (value **FAVORITE**) will be returned. If this information is missing, favorites and folders will be returned.
favoritefolder (optional)	Determines whether private (**FAVORITES_PRIVATE**) or public (**FAVORITES_SHARED**) favorites are to be queried. Default value is **FAVORITES_SHARED**
language (optional)	Specifies the language of error messages. If this information is missing, error messages are output in the language in which the PPM server was started.

The data for **querytype**, **favoritetype**, and **favoritefolder** must be written in capital letters.

ERROR MESSAGES

If a query cannot be executed correctly, an **error** XML element is output instead of the **data** XML element. The text of the error is specified in PCDATA and the identifier in the **id** attribute.

5.2 HTML table

As an alternative to the previously described XML format you can display the result of your query in a formatted HTML table.

The URL is composed as follows.

QUERY PATTERN

`http://host:port/kontextname/urlquery/html/query.do?parameterliste`
Configuration and transferred parameter list are the same as described in chapters URL query (Page 6) and Submitting a query (Page 4).

FORMATTING

You can customize selected display properties for the HTML table by adjusting various styles. The styles are saved in a document template and can be edited in the **Report definition** component of the PPM user interface.

Tip

For further information on editing a document template, please refer to the PPM online help.

The standard styles are saved in the document template **_default_queryapi_html** (file name: _default_queryapi_html.ret).

If you want to use a specific document template for representing your HTML table you can specify the name of that template in the optional URL argument **reporttemplate**. The document templates are saved in the directory **config\<client name>\report\reportdefinitions** of your PPM installation.

STYLE FOR LIST TABLES

Style	Description
table-header	Specifies the appearance of the table's column headers
table-content	Specifies the appearance of the table's data cells
table-color-properties	The font color determines the color of the table borders and grids, and the background color determines the color of the table background.

STYLE FOR CROSSTABS

Style	Description
crosstable-header	Specifies the appearance of the upper left cell of the crosstab
crosstable-iteration-header	Specifies the appearance of the crosstab's iteration headers
crosstable-iteration-content	Specifies the appearance of the crosstab's iteration steps
crosstable-ki-content	Specifies the appearance of the measure names in the crosstab (if multiple measures are displayed)
crosstable-result-header	Specifies the appearance of the headers of sum rows and columns in the crosstab
crosstable-result-content	Specifies the appearance of the content of sum rows and columns in the crosstab
crosstable-content	Specifies the appearance of the crosstab's measure values
table-color-properties	The font color determines the color of the table borders and grids, and the background color determines the color of the table background.

The selected background color determines the background color of the corresponding table cells in all styles (except for **table-color-properties**).

5.3 Display analysis results as an image

You can display an analysis dynamically created by the PPM server as a graphic in your HTML page. The data is queried by specifying a favorite. As an option, you can also specify filters to further restrict the data volume.

QUERY PATTERN

`http://host:port/contextname/urlquery/image/query.do?parameterliste`

Example

http://localhost:8080/API_demo41_en/urlquery/image/query.do?favoritename=<favorite name>

FAVORITE TYPES

You can use all favorites based on any diagram type, EPC view, communication analysis, or Relation Explorer analysis.

If you enabled the query interface the pop-up menu entry **URL for query interface** is available in PPM for certain favorites based on a supported favorite type. You can use the pop-up menu entry to copy the relevant URL to the clipboard. The URL created is already correctly coded and contains the language specified when logging into the PPM front-end in the **language** argument. Filter settings specified in addition to the favorite will not be considered (see chapter URL query (Page 6)).

The query returns the binary data stream of the analysis graphic in the format **PNG** (Portable Network Graphics) or **JPG**.

In addition to the default parameters **favoritename**, **favoritefolder**, **language**, **filterfavorite**, and a filter list (see chapter URL query (Page 6) and Query parameters (Page 4)) you can specify further parameters for the query:

Parameter	Description
imageheigth (optional)	Height of the graphic to be created Maximum value 2000 pixels
imagewidth (optional)	Width of the graphic to be created Maximum value 2500 pixels
imageformat (optional)	Format of the graphic to be created Valid values: **PNG** (default value), **JPG**.

GRAPHIC SIZE

Width and height of the graphic to be created must always be specified together. If one information is missing or invalid, the other will be ignored. In this case, the default values are applied. A value is identified as invalid if the specified value is less than or equal to zero, or if it is larger than the maximum value.

DEFAULT SIZES

For graphics based on a chart, a default size of 640 pixels (width) by 480 pixels (height) applies. For graphics based on EPC and interaction analyses, the default size and thus the aspect ratio of the created graphic is determined by the number of objects the graphic contains.

ERROR BEHAVIOR

If an error occurs during the creation of the graphic, a graphic including the reason for the error in text form is created. Possible causes of errors are:

- The specified favorite does not exist.
- The specified favorite type is not supported.
- The specified filter(s) include errors
- The specified favorite does not return any data

6 Examples

6.1 JavaScript (http query)

This simple example shows an html page that displays input boxes for User, Password, and Favorite, where the required values can be entered. Clicking the **Create table** button imports the data via the query interface and displays it as a table below the form.

Example

To follow the example for Internet Explorer, create the file **test_JS.html** in the directory **<installation directory>\ppm\server\bin\work\data_ppm**webappQueryApi with the following content:

```
<html>
<head/>
<body>
<script type="text/javascript" language="JavaScript">
var g_QueryBaseURL = "http://localhost:16350/API_umg_en";
function makeRequest(usr,pass,favorit,folder) {
  var sfolder = (folder)?"FAVORITES_PRIVATE":"FAVORITES_SHARED";
  var url = g_QueryBaseURL+"/urlquery/query.do?favoritename="
    +encodeURIComponent(favorit)+"&favoritefolder="+sfolder;
  if (window.XMLHttpRequest) { // Mozilla, Safari,...
    http_request = new XMLHttpRequest();
    logout_request = new XMLHttpRequest();
    if (http_request.overrideMimeType) {
      http_request.overrideMimeType('text/xml');
      logout_request.overrideMimeType('text/xml');
    }
  } else if (window.ActiveXObject) { // IE
    http_request = new ActiveXObject("Microsoft.XMLHTTP");
    logout_request = new ActiveXObject("Microsoft.XMLHTTP");
  }
  if (!http_request) {
    alert('Giving up :( Cannot create an XMLHTTP instance');
    return false;
  }
  http_request.onreadystatechange = renderResults;
  http_request.open('POST', url, true,usr,pass); //GET can also be used here
  http_request.send(null);
}

function renderResults()
{
  if (http_request.readyState == 4) {
    if (http_request.status == 200) {
      var srcTree = new ActiveXObject("Msxml2.DOMDocument.4.0");
      srcTree.async=false;
      srcTree.loadXML(http_request.responseXML.xml);
      var dd = srcTree.selectSingleNode("//queryresult");
      var x = dd.getElementsByTagName("column") ;
      var newEl = document.createElement('TABLE');
      newEl.setAttribute('cellPadding',5);
      var tmp = document.createElement('TBODY');
      newEl.appendChild(tmp);
      var row = document.createElement('TR');
      for (j=0;j<x.length;j++) {
        var container = document.createElement('TH');
```

```
      var theData = document.createTextNode(x[j].text);
      container.appendChild(theData);
      row.appendChild(container);
     }
     tmp.appendChild(row);
     var z = dd.getElementsByTagName('row');
     for (var i=0;i<z.length;i++) {
      var row = document.createElement('TR');
      for (var j=0;j<z[i].childNodes.length;j++) {
       if (z[i].childNodes[j].nodeType != 1) continue;
       var container = document.createElement('TD');
       var theData =
document.createTextNode(z[i].childNodes[j].firstChild.nodeValue);
        container.appendChild(theData);
        row.appendChild(container);
      }
      tmp.appendChild(row);
     }
     document.getElementById('writeroot').appendChild(newEl);
     logout_request.open('POST', g_QueryBaseURL+"/urlquery/logout.do", true);
     logout_request.send(null);
    } else if(http_request.status == 401) {
     alert('Login error! User/password combination does not exist!');
    } else {
     alert('There is a problem with the request. Status:'+http_request.status);
    }
  }
}
</script>

<form action="" method="post" name="myform">
<table>
  <tr><td>User:</td><td><input type="text" name="user" size="20"
value="system"></td></tr>
  <tr><td>Password:</td><td><input type="password" name="password" size="20"
value="manager"></td></tr>
  <tr><td>Favorite:</td><td><input type="text" name="favorit" size="40"
value="\MVs\Helpdesk\Favorite_Table"></td></tr>
  <tr><td>Private favorite:</td><td><input type="checkbox" name="folder"
value="1"></td></tr>
</table>
<input type="button" value="Create table"
onclick="javascript:makeRequest(document.forms['myform'].elements['user'].value,
document.forms['myform'].elements['password'].value,document.forms['myform'].ele
ments['favorit'].value,document.forms['myform'].elements['folder'].checked);retu
rn true"/>
</form>
<div id="writeroot"></div>
</body>
</html>
```

7 Manage the query interface cache

To retrieve data, the PPM server supplies a cache that provides the data more quickly for repeated queries.

You can adjust the query cache in the **keyindicator_settings.properties** file in <installation directory>\ppm\server\bin\work\data_ppm\config\<client>.

Extract from the keyindicator_settings.properties file:

```
#---------------------------------
# Options for scorecard cache
#---------------------------------
# Memory based scorecard results LRU cache size. A value of zero disables this
# cache!
# Valid values: '0' or greater
SCORECARD_LRU_CACHE_SIZE=100

# Memory based history of scorecard queries LRU cache size. This cache will be used
# for precalculation of scorecard results after a data import into PPM system.
# A value of zero disables this cache!
# Valid values: '0' or greater
SCORECARD_HISTORY_LRU_CACHE_SIZE=250

# Specifies whether memory scorecard cache accesses are logged. (Only
# relevant if LOG_CACHE is set to 'true'.
LOG_SCORECARD_LRU_CACHE=false
```

The **SCORECARD_LRU_CACHE_SIZE** setting determines the number of query results held in the memory-based cache. If there is a large number of queries, the value for this setting should be increased to guarantee that the cache functions efficiently.
The history cache stores the last queries. The **SCORECARD_HISTORY_LRU_CACHE_SIZE** setting specifies the number of query results to be included in cache pre-calculation. This kind of cache precalculation is, for example, triggered after data import via **runppmimport**. It is advantageous to have a history cache larger than the LRU cache because also the PPM server accesses cache pre-calculation. The cache size of the PPM server is not limited.

DATABASE-BASED CACHE

The history cache is memory-based, that is, after restarting the PPM server the cache is empty and the PPM server calculates all queries again. By specifying the **-savepchistory** option of the **runppmadmin** command line program, you can save the history cache in the database before terminating the PPM server. Already existing cache content will be overwritten or deleted. Alternatively, you can use the **-addpchistory** option. In this case, the database is updated or extended with the history cache contents.

By default, the PPM server is configured for using caches (both **USE_KI_CACHE** and **AUTO_FILL_CACHE** keys of the client-specific configuration file **Keyindicator_settings.properties** with the value **true**), so that after starting the PPM server, the history cache is automatically filled with the content previously saved in the database when queries occur You can load the database-based cache into the history cache by explicitly specifying the **-fillcache** option of the **runppmadmin** command line program.

To delete the database-based cache, call up the **runppmadmin** command line program using the **-clearpchistory** option.