Developer's Guide - QPR Reporting Add-on

Version 2016.1.0

Table of Contents

1 Introduction

QPR Reporting Add-on is a combined installation package containing the following parts:

- QPR Web Views
- QPR Word Reports
- Expression Engine for QPR Suite
- QPR Reports Menu

All parts except QPR Reports Menu are .Net 4.5 web services hosted in IIS, and they interact with QPR Suite using QPR Web Services interface. All parts also contain the common expression language.

In addition to the general installation instruction described by this document, the individual parts contain additional settings which are described in their own documentation.

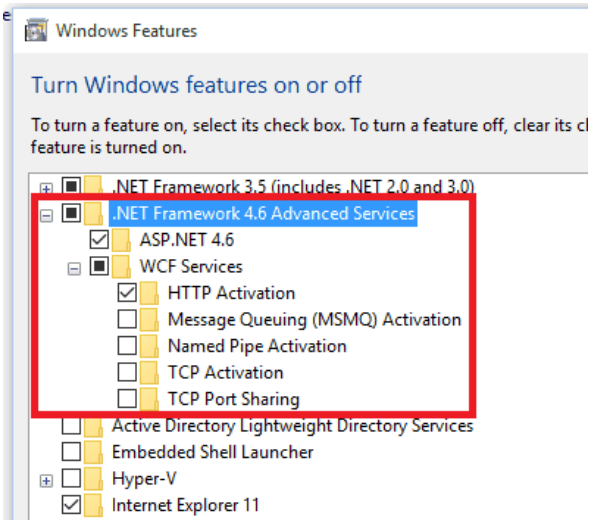
QPR Reporting Add-on doesn't have any compatibility defined for QPR Suite. See the compatibility of individual accelerators.

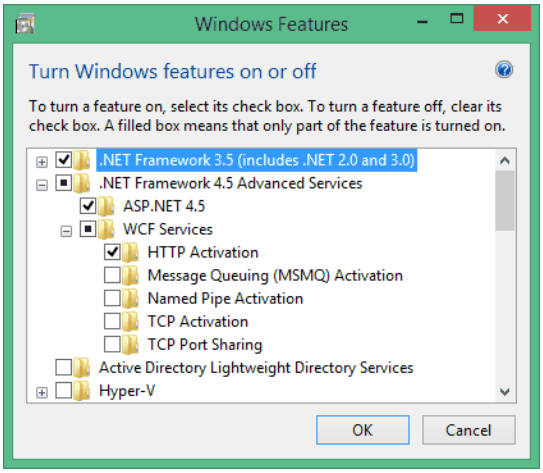
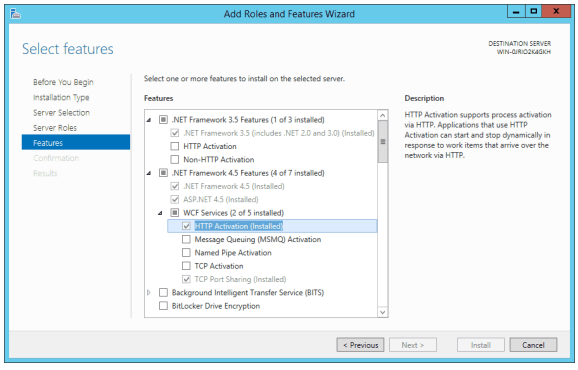
2 Installation

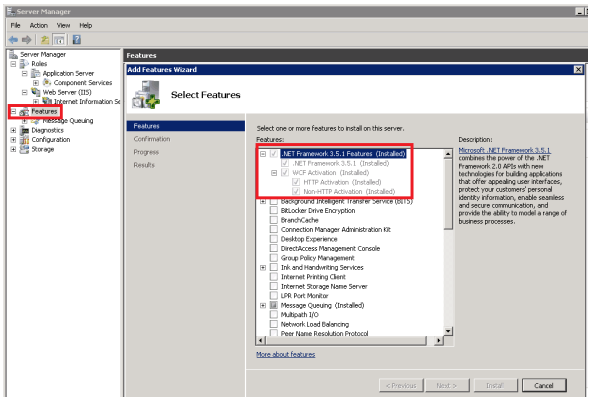
Follow these steps to install QPR Reporting Add-on. There are different paths for each QPR Suite version. These instructions use X in the folder names, that is replaced by the number of the corresponding QPR Suite version.

1. Check whether the QPR environment uses **Windows authentication** (IWA) and/or **HTTPS connection**. Windows authentication is used when QPR system is connected to LDAP/AD and Windows user accounts are used to login to QPR. HTTPS connection is in use when QPR Portal url starts with "https".

2. Open **Programs and Features** (in Windows **Control Panel)** and click **Turn Windows features on or off**. Check that components listed in the following table are installed. The installation procedure depends on Windows version. QPR Reporting Add-on needs **.Net Framework 4.5.1** or later version.

Windows Version	Required Components
Windows 10 (already includes .NET Framework 4.6)	• All components in **Internet Information Services** (except **FTP Server**) • Following Windows features: (see image below) **.Net Framework 4.6 Advanced Services > ASP.NET 4.6** **.Net Framework 4.5 Advanced Services > WCF Services** > **HTTP Activation** 
Windows 8 (already includes .NET Framework 4.5.1)	• All components in **Internet Information Services** (except **FTP Server**) • Following Windows features: (see image below) **.Net Framework 4.5 Advanced Services > ASP.NET 4.5**

<table>
<tr><td></td><td>.Net Framework 4.5 Advanced Services > WCF Services > HTTP Activation
 
</td></tr>
<tr><td>Windows Server 2012 R2
(already includes .NET Framework 4.5.1)</td><td>• All components in Internet Information Services (except FTP Server)
• Following Windows features: (see image below)
.Net Framework 4.5 Features > .Net Framework 4.5
.Net Framework 4.5 Features > ASP.NET 4.5
.Net Framework 4.5 Features > WCF Services > HTTP Activation
 
</td></tr>
<tr><td>Windows Server 2012
(needed .Net Framework is not installed by default)</td><td>• All components in Internet Information Services (except FTP Server)
• Following Windows features:
.Net Framework 4.5 Advanced Services > .Net Framework 4.5
.Net Framework 4.5 Advanced Services</td></tr>
</table>

	> **ASP.NET 4.5** **.Net Framework 4.5 Advanced Services > WCF Services > HTTP Activation** • Install .Net Framework 4.5.2: Online installer: http://www.microsoft.com/en-us/download/details.aspx?id=42643 Offline installer: http://www.microsoft.com/en-us/download/details.aspx?id=42642
Windows Server 2008 R2 (needed .Net Framework is not installed by default)	• All components in **Internet Information Services** (except **FTP Server**) • All components in **.Net Framework 3.5.1** (see image below) • Install .Net Framework 4.5.2: Online installer: http://www.microsoft.com/en-us/download/details.aspx?id=42643 Offline installer: http://www.microsoft.com/en-us/download/details.aspx?id=42642 

3. Check that QPR Web Services web.config file is a proper one (located in **C:\Program Files\QPR Software Plc\QPR 201X.1 Servers\WebServices**). In the default QPR installation there are files **web.config** and **web.config.IWA**. If QPR environment uses Windows authentication (refer to step 1), the latter file must be taken into use by renaming

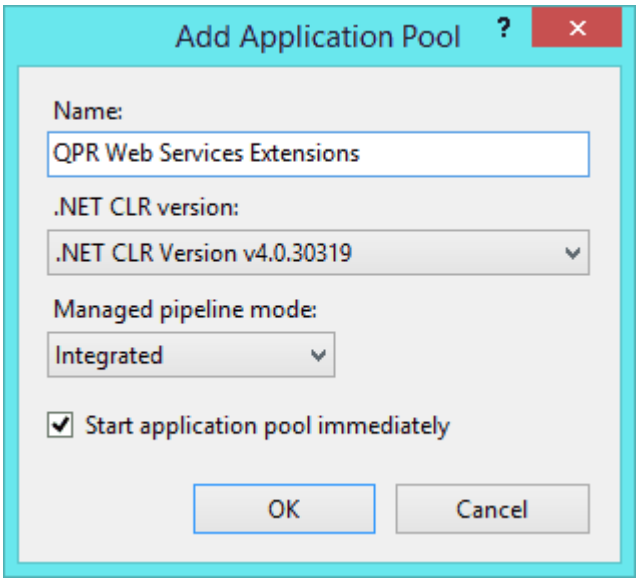
   a. **web.config** to **web.config.noIWA**, and

   b. **web.config.IWA** to **web.config**.

4. If using Windows authentication (refer to step 1), make sure **C:\ProgramData\QPR Software\QPR 201X\201X.1\Servers\Settings\QPR_Servers.ini** has setting **IWACGIBinaryHost=127.0.0.1** and **CGIBinaryHost=127.0.0.1**. The file already contains these setting, but they may be empty. QPR service restart is needed if you need to change this setting.

5. Copy **QPRWebServicesExtensions** folder to IIS published files in **C:\inetpub\wwwroot** (QPR Reporting Add-on can be deployed in any location in IIS, though).
6. There are following preconfigured files available to be used QPR Reporting Add-on **web.config** file:
    a) **web.config**: for HTTP connection and Anonymous authentication
    b) **IWA.web.config**: for HTTP connection and Windows authentication
    c) **HTTPS.web.config**: for HTTPS connection and Anonymous authentication
    d) **HTTPS+IWA.web.config**: for HTTPS connection and Windows authentication

    Copy a suitable config file to QPRWebServicesExtensions folder and rename as **web.config** (**C:\inetpub\wwwroot\QPRWebServicesExtensions\web.config**). Please do not mix up QPR Reporting Add-on web.config file (in **C:\inetpub\wwwroot\QPRWebServicesExtensions\web.config**) with QPR Web Services web.config file (in **C:\Program Files\QPR Software Plc\QPR 201X.1 Servers\WebServices\web.config).**
7. Make sure QPR Web Services Server is running: See in the **Windows Task Manager** (in **Details** tab) that there is a process **Qpr.WebServices.Server.exe**. The installation cannot be continued until QPR Web Services Server is running properly.
8. Set the parameters in the QPRWebServicesExtensions web.config's **appSettings** section. Parameters are listed in Settings.
9. In **IIS Management Console**, go to **Application Pools** (in left side hierarchy). Create a new application pool by clicking **Add Application Pool...**. Use the settings in the image below (settings **v.4.0.x** and **Integrated**). Please do not change settings for existing application pools, if they are used by other web applications because then the other applications may stop working. Especially QPR Suite has an application pool that is **v.2.0.x** and **Integrated**.



10. Find the QPRWebServicesExtensions folder in **IIS Management Console** and click **Convert to Application** (secondary mouse button). Select the previously created application pool **QPR**

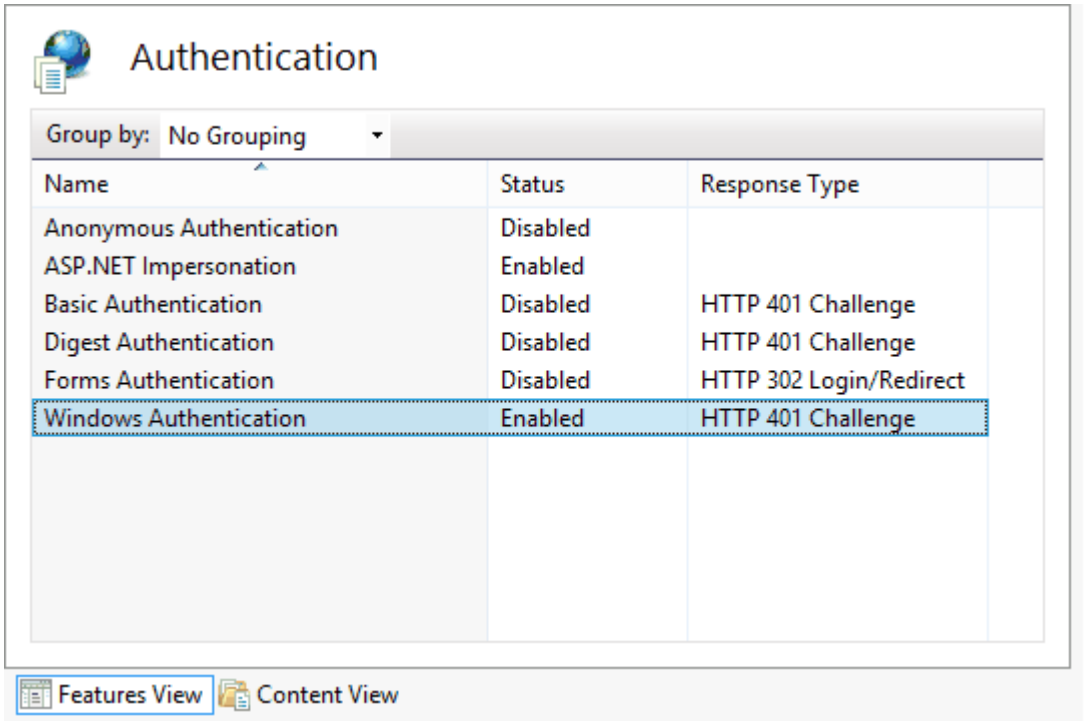
Reporting Add-on.

11. Check the IIS authentication settings by clicking QPRWebServicesExtensions web application is IIS Management Console (on the left side hierarchy). Check that **Features View** is opened (in bottom), and doubleclick **Authentication**. The authentication settings must match with the web.config file, which was set in step 3:

    a. For Windows authentication: **Anonymous Authentication** must be **Disabled** and **Windows Authentication** must be **Enabled**. (see the image below)

    b. For Anonymous authentication: **Anonymous Authentication** must be **Enabled** and **Windows Authentication** must be **Disabled**.

ASP.NET Impersonation must be **Enabled** in both cases.



12. Replace the file **C:\Program Files\QPR Software Plc\QPR 201X.1 Servers\WebServices \servicetester.aspx** with the new **servicetester.aspx**. Make a backup copy of the original file with name **servicetester.aspx.original**. (Alternatively, use the **servicetester.patch** file that described the changes to the servicetester.aspx file.)

13. Configure QPR Reporting Add-on settings listed in **Settings**. Especially check the **qprwebserviceaddress** carefully. Quick guide for usual configurations:

    a. When Windows authentication is not in use: **wcfsecuritymode**=**none** and **qprauthenticationmode**=**passedsession**

    b. When Windows authentication is in use: **wcfsecuritymode**=**message** and **qprauthenticationmode**=**windows**

14. Configure individual report part's settings, which are mentioned in the documentation of the report part. It's sufficient to do this only for those parts that need to be set operational.

15. Check that QPR Reporting Add-on is working by making the tests listed in chapter **Installation tests**. If you encounter any issues, check if any of the error situations described in

Troubleshooting were encountered.

16. Copy **DWV templates** folder as a DWV templates root folder (setting dwvtemplatesphysicalpath in QPR Reporting Add-on' web.config file). Also copy DWR templates folder as a DWR templates root folder (setting dwrtemplatesphysicalpath in QPR Reporting Add-on' web.config file) and **trend_down.png** and **trend_up.png** files from **DWV templates\Images folder** to **C:\inetpub\wwwroot\qpr201X-1\qprsoftware\portal\images**.

17. Deploy Reports Menu addon by replacing **mainview.tpl** and **headerview.tpl** in **C:\ProgramData\QPR Software\QPR 201X\201X.1\Servers\Templates\WAS\Portal** with the files from the **Reports Menu folder**. (Alternatively, use the **externalreportsmenu.patch** file.)

18. Copy **icon_external_reports.png** from **Reports Menu folder** to **C:\inetpub\wwwroot\qpr201X-1\qprsoftware\portal\images**.

19. Copy **jquery.filedownload.js** from **Reports Menu folder** to **C:\inetpub\wwwroot\qpr201X-1\qprsoftware\Common\scripts**.

20. Add the following CSS to **C:\inetpub\wwwroot\qpr201X-1\qprsoftware\stylesheets\custom.css**:

```
#ReportsToolbarMenu {
  cursor: pointer;
}

#ReportsToolbarMenu .activetarget {
  background-color: inherit;
}

.visiblereportmenulink, .reportmenumain > a {
  background: url(../portal/images/icon_reports.png) no-repeat left -1px;
}

.disabledreportmenulink {
  background: url(../portal/images/icon_reports.png) no-repeat left -1px;
  color: #BBBBBB !important;
  cursor: default;
}

#InformationViewFrame {
  background-color: white;
}
```

21. Restart Windows service for QPR Suite, or clear QPR Portal templates cache. In addition, clear web browser's cache.

2.1 Settings

QPR Reporting Add-on is configured using the file C:\inetpub\wwwroot\QPRWebServicesExtensions\web.config. The file has following settings in the **configuration** > **appSettings** section. The greyed settings are unusual, so for most installations they can be ignored.

Attribute	Description
qprwebserviceaddress	QPR Web Services' url address. This should point directly to QPR Web Services server's port. The default port is 9002, but the actual port in use can be seen in **QPR Configuration Manager** (**Common** > **Server locations**

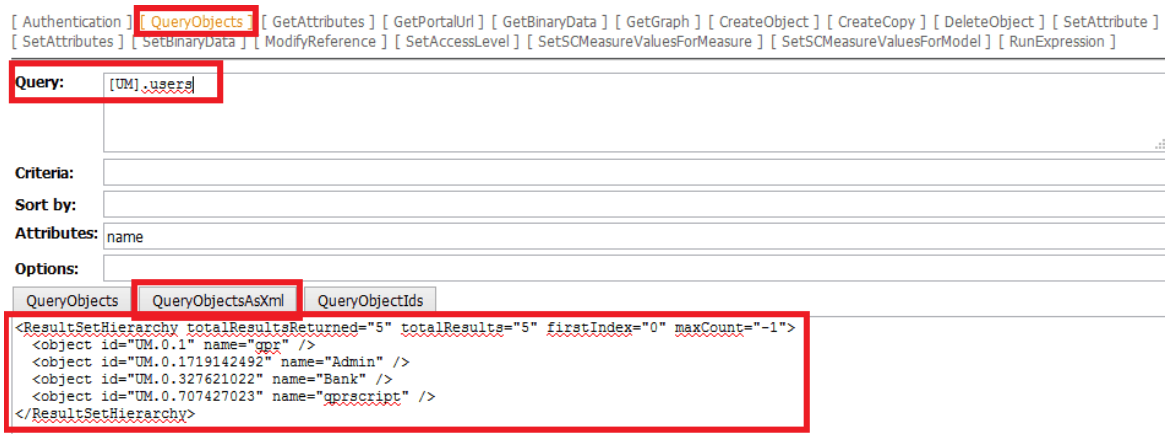
	> **Web services server**). Example: http://localhost:9002/QPR201X-1/Portal/QPR.Isapi.dll/wsforward/mainservice.svc/wshttp Notes: • It's recommended to use **localhost** as a hostname if the QPR Web Services Server is in the same computer. • The address starts with **http** even if QPR environment uses https. • Check the proper url path from QPR Portal address (**QPR201X-1** in the example). The path is QPR version specific by default. • Validity of the address can be checked by opening the address in the server computer using browser without the ending **/wshttp**, e.g. http://localhost:9002/QPR201X-1/Portal/QPR.Isapi.dll/wsforward/mainservice.svc. A page displaying **You have created a service** should open. Note that this address doesn't work in client computers (this is because "localhost" always references to that computer where the browser is running).
wcfsecuritymode	QPR Web Services security settings. Must correspond to QPR Web Services settings (in C:\Program Files\QPR Software Plc\QPR 201X.1 Servers\WebServices\web.config). (Refer to step 3 in the installation instructions.) Options: • **none**: Use this when Windows authentication is not in use. • **message**: Use this when Windows authentication is in use. • **transport**: Usually not used. • **transportwithmessagecredential**: Usually not used. • **notconfigured**: Usually not used. WCF client's security settings are not set programmatically. Manual configuration by the accelerator is used instead.
qprauthenticationmode	Determines how the add-on parts authenticate to QPR Web Services. Options: • **passedsession**: QPR Web Services' session id must be passed as a parameter. The parameter is passed differently in different accelerators. This options is usually used when *wcfsecuritymode* is *none*, but also *message* is possible. • **windows**: Windows user making the request is used to authenticate to QPR Web Services. When this option is used, *wcfsecuritymode* must be *message* (*none* is not possible). • **fixedcredentials**: A single user account is used for all requests. The account's username and password are defined in the configuration (see settings *username* and *password*). This options is not usually used. For more information, see **Authentication and Data Security**.
dwrtemplatesphysicalpath	Folder in the file system where QPR Word Reports template files are located.
dwvtemplatesphysical	Folder in the file system where QPR Web Views template files are located.

path	
username	Password for QPR Suite when *authenticationmode* is *fixedcredentials*.
password	Password for QPR Suite when *authenticationmode* is *fixedcredentials*.
executionTimeout (in the httpRuntime tag)	Timeout for request processing in seconds. Usually there is no need to change this setting. The value should be increased if there are heavy requests which take time to run. Note that the timeout limit works as a protection when the processing never ends as a results of an error (this may be possible in e.g. recursive reports).

2.2 Installation Tests

Do the following tests to confirm that QPR Reporting Add-on is working:

1. Open QPR Web Services Tester, which is usually in **http://SERVERNAME/QPR201X-1/Portal/QPR.Isapi.dll/wsforward/servicetester.aspx**. The SERVERNAME can be found in QPR Portal url address. The address starts with **https** instead of **http**, when also QPR Portal address starts with https.
2. Check that the page contains **RunExpression** tab.
3. Set valid credentials in the **Authentication** tab.
4. Go to RunExpression tab click **Run** using the expression it contains by default. It should return **Ok** (below the Run button). This confirms that the .Net 4.5 application is running in IIS and the QPR Web Services connection works.
   a. If this works, you may stop testing here.
   b. If this doesn't work, go to step 5.
5. Open **http://SERVERNAME/QPRWebServicesExtensions/ExpressionEngine.svc**. The address starts with **https** instead of **http**, when also QPR Portal address starts with https. There should open a page stating **You have created a service**. If an internal server error (error code 500) with no error details is returned, test the url in the server, because in the server the error message has more details.
   a. If this works, the .Net 4.5 application is running properly in IIS. Go to step 6.
   b. If this doesn't work, there is a problem with IIS settings or .Net 4.5 installation.
6. Confirm that QPR Web Services is working by making a query using QPR Web Services Tester in the QueryObjects tab. You can query for example **[UM].users** like in the image below. When clicking **QueryObjectsAsXml**, the test is successful, if a text starting with **<ResultsetHierarchy** … appears. Note that your query results may be different than the results in the image. If this test doesn't work, there is a problem with QPR Web Services.



2.3 Authentication and Data Security

This chapter contains information about authentication and data security of the add-on and QPR Suite to make the installation successful also from security point of view. The add-on needs to authenticate to QPR Web Services to get needed data. There are three methods to authenticate:

- Use Windows authentication. Windows user authenticates to the add-on (provided by IIS), and same user is used to authenticate to QPR Web Service (this is called impersonation). It's advisable to use this authentication method if available.
- Pass QPR Web Service's session id as a url parameter to the template. The accelerator uses the session id directly when accessing to QPR Web Services. This approach requires a functionality in QPR Portal to first authenticate to QPR Web Service and then pass the session id to the template. Use this authentication method, if Windows authentication is not possible.
- Preset username and password (either QPR or Windows user) in QPR Reporting Add-on configuration. These credentials are used by the add-on when logging in to QPR Web Services. This authentication method is only used for special purposes.

The authentication method is determined by QPR Reporting Add-on setting "qprauthenticationmode".

2.4 Running Multiple QPR Reporting Add-on Instances Simultaneously

It may be required to run multiple QPR Reporting Add-on instances simultaneously, e.g. when different versions or different QPR Reporting Add-on settings are needed. Running multiple QPR Reporting Add-on instances simultaneously is easy: Copy the QPRWebServicesExtensions folder with a different name to IIS root folder, and make all the settings made to the default folder to that folder. The other instance is referenced using the other folder name in URLs.

2.5 Uninstallation

Follow these steps to uninstall QPR Reporting Add-on**:**

1. In **IIS Management Console** click **Remove** for the **QPRWebServicesExtensions** web application (mouse secondary button).
2. Delete the web application's folder **C:\inetpub\wwwroot\QPRWebServicesExtensions** in the disk.
3. Revert the original **C:\Program Files\QPR Software Plc\QPR 201X.1 Servers \WebServices\servicetester.aspx** from release package.

2.6 Troubleshooting

Issue	Resolution
Web browser returns: Could not load type 'System.ServiceModel.Activation.HttpModule' from assembly 'System.ServiceModel, Version=3.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089'.	See the resolution here: https://support.microsoft.com/en-us/kb/2015129
Web browser returns: Memory gates checking failed because the free memory (nnn bytes) is less than x% of total memory. As a result, the service will not be available for incoming requests. To resolve this, either reduce the load on the machine or adjust the value of minFreeMemoryPercentageToActivateService on the serviceHostingEnvironment config element.	The reason for this error is that there is little free memory in the system, and thus the primary solution is to get more free memory in the system. It's still possible to set the free memory limit to a lower level, but this may cause instability. To do that, find the setting minFreeMemoryPercentageToActivateService from the QPR Web Services Extensions' web.config and set the limit lower (e.g. to 2). More information: https://msdn.microsoft.com/en-us/library/dn458357(v=vs.110).aspx
Following errors are returned by Expression Engine tester: The message with Action 'http://schemas.xmlsoap.org/ws/2005/02/trust/RST/Issue' cannot be processed at the receiver, due to a ContractFilter mismatch at the EndpointDispatcher. This may be because of either a contract mismatch (mismatched Actions between sender and receiver) or a binding\security mismatch between the sender and the receiver. Check that sender and receiver have the same contract and the same binding (including security requirements, e.g.	The reason is the QPR Web Services and QPR Web Services Extensions WCF settings don't match. Please check installation steps 3, 4 and 6.

<table>
<tr>
<td>Message, Transport, None).Secure channel cannot be opened because security negotiation with the remote endpoint has failed. This may be due to absent or incorrectly specified EndpointIdentity in the EndpointAddress used to create the channel. Please verify the EndpointIdentity specified or implied by the EndpointAddress correctly identifies the remote endpoint.

The message could not be processed. This is most likely because the action 'http://www.qpr.com/ns/IService/GetAttributeAsString' is incorrect or because the message contains an invalid or expired security context token or because there is a mismatch between bindings. The security context token would be invalid if the service aborted the channel due to inactivity. To prevent the service from aborting idle sessions prematurely increase the Receive timeout on the service endpoint's binding.</td>
<td></td>
</tr>
<tr>
<td>When opening QPR Web Services Tester, the following message appears …
Redirecting to mainservice page ....</td>
<td>There may be a problem with IIS handler mappings. Tests with following settings:

• Remove C:\inetpub\wwwroot\web.config if it exists. Make a backup before deleting.

• Contents of C:\inetpub\wwwroot\qpr201X-1\web.config should be

<?xml version="1.0" encoding="UTF-8"?>
<configuration>
<system.webServer>
<directoryBrowse enabled="false" />
<httpErrors>
<clear />
</httpErrors>
<handlers accessPolicy="Read, Execute, Script">
<remove name="AssemblyResourceLoader-Integrated-4.0" />
<remove name="AssemblyResourceLoader-Integrated" />
<remove name="AXD-ISAPI-4.0_64bit" />
<remove name="AXD-ISAPI-4.0_32bit" />
<remove name="AXD-ISAPI-2.0" />
<remove name="AXD-ISAPI-2.0-64" />
<remove name="PageHandlerFactory-ISAPI-4.0_64bit" />
<remove name="PageHandlerFactory-Integrated-4.0" />
<remove name="PageHandlerFactory-ISAPI-4.0_32bit" />
<remove name="PageHandlerFactory-Integrated" />
<remove name="PageHandlerFactory-ISAPI-2.0" />
<remove name="PageHandlerFactory-ISAPI-2.0-64" />
<remove name="svc-ISAPI-4.0_64bit" />
<remove name="svc-ISAPI-4.0_32bit" />
<remove name="svc-Integrated-4.0" />
<remove name="svc-ISAPI-2.0-64" />
<remove name="svc-ISAPI-2.0" />
<remove name="svc-Integrated" />
</handlers>
</system.webServer>
</configuration></td>
</tr>
</table>

	• Contents of **C:\inetpub\wwwroot\qpr201X-1\Portal\web.config** should be

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <system.webServer>
    <handlers accessPolicy="Read, Execute, Script" />
    <directoryBrowse enabled="false" />
    <defaultDocument>
      <files>
        <clear />
        <add value="Default.htm" />
        <add value="Default.asp" />
        <add value="index.htm" />
        <add value="index.html" />
        <add value="iisstart.htm" />
        <add value="default.aspx" />
        <add value="QPR.Isapi.Dll" />
      </files>
    </defaultDocument>
  </system.webServer>
</configuration>
```

3 QPR Expression Engine

Expression Engine for QPR Suite is contains two components:

- Expression Engine Web Service
- Expression Engine Service Tester

Expression Engine Web Service is a web service used to run NCalc expressions provided by QPR Suite Accelerator .Net Tools. This expression engine is embedded into many other accelerators, such as DWR, DWV and QPR Suite Internal Integration.

Expression Engine Service Tester is an extension to QPR Web Service Tester, and it can be used to test expressions when e.g. developing report templates or configuration files for other accelerators. It's possible to define multiple expression which are calculated consecutively in a single web service operation. In that case expression results are stored in variables, and variables can be used in subsequent expression. Example:

var1=**3 + 9 / 3**
var2=**2*[var1]**
var3=**'Value is ' + [var2]**

Expression Engine Web Services is a IIS hosted web application, and Expression Engine Service Tester is implemented with changes to QPR Web Service Tester web page html template.

QPR Expression Engine is compatible with QPR 2016.1.

3.1 Expression Engine Input Parameters

Type: **rootobject**		
Attribute	**Type**	**Description**
webServiceSessionId	string	QPR Web Service's session id. Used only if **authenticationmode**=**passedsession**.
expressionSet	string[]	List of expressions
variableNameSet	string[]	List of variable names. The variables get the calculated expression values, and the variables are available as arguments for the next expressions. The value of the last expression is returned by the operation.

3.2 Output Data Format

Following output data format is used by both services.

Type: **ResultDataTable**		
Attribute	**Type**	**Description**
headers	Header[]	Array of header rows. There can be one or two headers.
rows	ResultDataTableRow[]	Array of data rows

Type: **ResultDataTableRow**		
Attribute	**Type**	**Description**
cells	ResultDataTableCell[]	Array of data cells
leftIdentifier	string	
rightIdentifier	string	

Type: **ResultDataTableCell**		
Attribute	**Type**	**Description**
rawValue	string	Unformatted raw value
formattedValue	string	Value that is displayed to users
sortValue	string	Value that can be used to sort the values
datatype	string	Datatype of the value: string, numeric, int

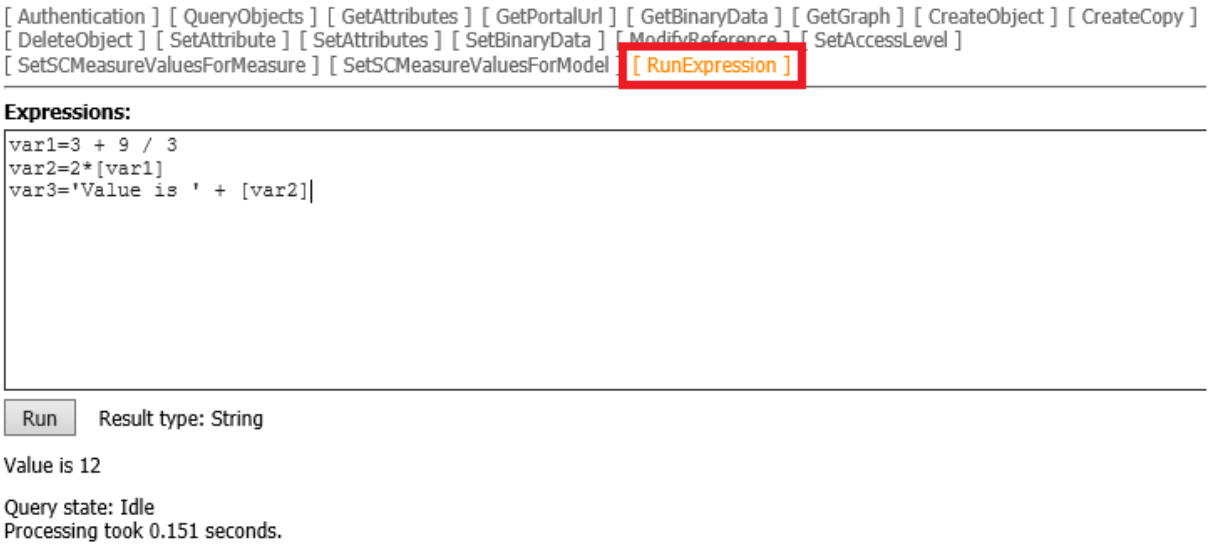
Type: **ResultDataTableHeader**		
Attribute	**Type**	**Description**
cells	ResultDataTableHeaderCell[]	

Type: **ResultDataTableHeaderCell**

Attribute	Type	Description
label	string	
columnSpan	int	Number of columns the header extends

3.3 Example

Following image shows QPR Web Service tester where Expression Engine Service Tester is installed.



4 QPR Word Reports

QPR Word Reports is a technology for QPR Suite to generate customized Word reports displaying content from the QPR system. QPR Word Reports works in IIS as a web application, and reports are accesses by a url address (using an http GET request). Data for reports is fetched from QPR system using QPR Web Service interface.

Microsoft Word is not needed on the server side, as Word file generation is implemented using Open XML SDK. Actually Word is not required in the client side either, because Word documents can be opened using OpenOffice (file format is Office Open XML).

Main Features

QPR Word Reports contains the following main features:

- The reports are run as they are requested, so the reports always contain the latest data.
- Reports can have parameters affecting report contents.
- QPR Word Reports contains an embedded formula calculation engine. All the tags may be defined using expressions which values are calculated as the report is run.
- The report templates are normal Word files so they can contain any Word file features (such as headings, tables, cross references, headers, footers and images).
- Reports may get theirs content from other reports, called subreports. Also recursive reports are supported (reports calling themselves). Subreports makes it possible to assemble the report from smaller report "parts".
- Adding Word hyperlinks is possible

Concepts

QPR Word Reports is based on the following concepts:

- Report templates are normal Word files which may have any contents. Reports are based on these template files. When a report is run, a template is taken as a basis, and during processing of the report, content from QPR system is added to the template, forming the final report.
- Templates contain tags, which instruct how the content is added to the report and how dynamic parts of the report are built.
- Tags contain attributes, which offer additional information for the tag.
- QPR Web Service is used through its operations. Operations used by QPR Word Reports are GetAttributeAsString, QueryObjects, GetGraph, GetBinaryData and GetPortalUrl. All QPR Web Service operations are documented under http://kb.qpr.com/qpr2016-1/index.html?functions.htm.
- Subreport is a report that is called from another report (main report). The subreport is processed like any report and its contents are embedded in the main report. Main report may pass parameters to the subreport.

- Loop is a list of QPR system objects (elements) returned by QPR Web Service's QueryObjects operation. In a loop, a subreport is called for every looped object.

To open QPR Word Reports reports in QPR Portal, QPR Reports Menu is needed. It's able to automatically pass parameters, such as model or selected object to the report.

QPR Word Reports cannot itself store previously run reports, or run reports based on a schedule. When a report is viewed, it's always generated at that point and data for the report is fetched from QPR system. Because of this, there may be a delay when getting a report, especially if the report contains dozens of pages.

Reports are accessed using a url address, which determines the report to be run and the url also passes all needed parameters to the report. The url depends on the server computer name and the path where QPR Word Reports is installed in IIS. Note that parameter names are case sensitive.

An example url: http(s)://SERVERNAME/QPRWebServicesExtensions/DynamicWordReports.ashx?report=reportname&parameter1=param1value&parameter2=param2value

QPR Word Reports is tested to be compatible with QPR 2016.1.

4.1 Configuration

Following table contains QPR Word Reports's parameters, which are configured in the **web.config** file of QPR Web Services Extensions.

Parameter name	Description
reporttemplateparameter (optional)	Defines the name of the parameter which passes the report template path. See chapter 2. Working with report templates for configuring the report template path.
qprtemplateidparameter (optional)	Parameter name which passes QPR's **Word report template** object id. This can be used when the report templates are stored in QPR Portal (i.e. QPR system objects) (see chapter 2. Working with report templates). Instead of report template path, the Word report template object id can be passed as a parameter. The Word report template object id gets a priority over report template path.
defaultimageformat (optional)	Default image format for QPR Web Service's GetGraph operation. This setting is used if image format is not explicitly defined.

4.2 Working with Report Templates

QPR Word Reports is based on Word report templates, which are normal Word files. When a report is run, the template is taken as a basis, and all the tags in the template are processed. Content is added to the report defined by the tags.

There are two options to store templates:

- **Reports tab in QPR Portal**: This option is suitable for running reports in production environments, because the report templates can be updated by QPR users (Portal administrator rights are required), and no access to QPR server is needed. Also, the report templates are stored in QPR system (i.e. in QPR database).
- **A disk drive** to where there is an access from the server running QPR Word Reports. This option is suitable for developing reports. It is faster for development because the developed template may be open while running the report. Also, option 1 requires the template to be saved manually to QPR Portal so that it can be run.

Both options can be used at the same time. If there are two templates with the same name in both places, the QPR Portal stored templates have priority.

Note that QPR Word Reports templates are entirely incompatible with QPR - Add-in for Microsoft Office report templates, so running them as QPR - Add-in for Microsoft Office reports is likely to cause an error.

Both Word file extensions (**docx** and **docm**) are supported. The output extension is the same as the extension of the main report template. File extension is not specified when referencing to template names in disk. If there are two files with the same name but different extension in the same folder, docx is selected as a template. Subreport extensions don't affect the result report.

When stored in a disk, reports may be organized in folders. In that case report templates are referenced using syntax **folderName/reportName**. Folders inside folders are also supported.

Report template root location is determined by configuration setting **reporttemplatephysicalpath**.

When a report processing fails, an html page showing an error message is displayed. The error message is likely to reveal in which report template and tag the error occurred. Also stack trace is shown, but that is helpful information only for developers.

For QPR Web Service related debugging, logging mode is available. The logging mode prints all QPR Web Service requests and responses in the end of the report. It can be used to get more information, whether an error is related to data fetched from QPR Web Service or the report itself. The logging mode is activated by adding **&loggingmode=true** to the report url.

4.3 Template Tags

Tags are used to add content to the report. Syntax for defining tags is following:

<#tagname attribute1="attribute1 value" attribute2="attribute2 value" attribute3="attribute3 value">

If an attribute value is calculated using an expression (formula), two equals signs are used instead of one, e.g.

<#tagname attribute1=="1+1" attribute2="1+1"> (attribute1 value is integer 2, attribute2 value is string "1+1").

In the following sub chapters optional tag attributes are marked with "(optional)". Other tags are mandatory – if any of them is missing, an error is caused. In addition, for some of the mandatory attributes the value of the attribute must not be empty (i.e. **attributeName=""** is not allowed). If the attribute value comes from an expression, note that the expression may evaluate as empty (thus causing an error in non-empty mandatory attribute).

When processing a tag, attribute expressions are calculated from left to right. All report parameters, report variables and already calculated attribute values are available as expression **arguments** when later attribute expressions are calculated. An argument is kind of an input variable that can be used in an expression with syntax **[argumentName]**. E.g.:

<#expression attribute1="3" value=="[attribute1] + 2"> (result is 5)

Tag specific arguments have a priority over report parameters and report variables, when there are same names used.

Attribute names should only contain characters a-z, 0-9 and _.

Values for all boolean valued tags (such as **visible**) may be defined as "true" or "false" or as a boolean valued expression, e.g. **visible=="[variable1]=1"** (the result of the equality comparison is of type boolean). Tag visibility is processed in the normal left to right order. When visibility tag is processed and its value is false, the tag processing stops at that point. This has an effect on report run performance. In addition, possible later expression errors don't emerge in that case.

Word has a tendency to use characters " and " instead of character " as quotation marks. Characters " and " are not suitable quotation marks for QPR Web Service, so QPR Word Reports replaces these characters with " character for strings inputted to QPR Web Service. Also characters ' and ' are replaced by '. Note that all of these quotation marks should be escaped, when there is escaping needed (more about escaping in Expression Language).

All existing tags can be used in following document parts: main document, header, footer, footnotes and endnotes. Only Attribute and Expression tags can be used in comments.

Following abbreviated tag names can be used: **attribute=att**, **expression=exp**, **parameter=par** and **variable=var**. Tags must be written in lower case.

4.3.1 Attribute Tag (att)

This tag is used to get properties of QPR objects by using QPR Web Service's **GetAttributeAsString** operation (more information: http://kb.qpr.com/qpr2016-1/index.html?getattributeasstring.htm).

This tag uses **QueryObjects** instead of GetAttributeAsString, if there are multiple id's in the **object** attribute or if attribute **followrelations** used (more information below). If QueryObjects is used, **separator** and **sortby** attributes are applied.

Attribute	Description
object (string)	Object id or list of object id's separated by comma (,). If this contains only one id, it is passed as a parameter **objectId** of **GetAttributeAsString** operation. If this attribute is not provided or is empty, the tag will be evaluated to empty. (optional)
attribute (string)	This is passed as parameter **attribute** of **GetAttributeAsString** or **QueryObjects** operation.
options (string)	This is passed as parameter **options** of **GetAttributeAsString** or **QueryObjects** operation. (optional)
expression (exp. string)	An expression where the result is put as an argument "value" (see the examples). (optional)

followrelations (string)	Name of the relation attributes to follow to get another object or a set of other objects. If there are multiple subsequent relations to follow, the relations are separated by comma (,) (see the examples). (optional)
separator (string)	Character(s) separating list of attributes of different objects. Empty values are not shown (i.e. no empties between separators). Default is a new line characters. (optional)
sortby (string)	Sorting if there are multiple objects returned. Default sorting is by attribute "name". (optional)
visible (boolean)	Determines content visibility shown by the tag. See Setting Content Visibility. (optional)
style (string)	Determines text style. See Defining Styles. (optional)

Examples:

<#attribute object=="[objectId]" attribute="name">

<#attribute object="[measureId]" attribute="measure.value(series=\"ACT\",period=\"1 / 2015\")" expression="[value] * 2.54">

<#attribute object=="[measureId]" followrelations="childobjects,childobjects" attribute="measure.value(series=\"ACT\",period=\"1 / 2015\")" separator=", " sortby="name">

4.3.2 Dataset Tag

This tag is for showing information from a *dataset* in the report. Dataset consists of multiple columns and rows. Dataset tags shows a **header subreport** (once) and **row subreports** for each row in the dataset. Dataset tag has following attributes

Attribute	**Description**
data (dataset)	Dataset shown by the tag.
headertemplate (string)	Subreport template for dataset header. The header subreport is shown once in the report above the row subreports. Header subreport can be used e.g. for showing a column names. Header subreport is optional, so if the parameter is left out, no header subreport is shown. Following type of parameters are passed to the header subreport: **columnheader_1**, **columnheader_2**, … **columnheader_n** containing names of the dataset columns. If the column names are known, the header can be set fixed in the mainreport and header subreport is not needed. (optional)
headertemplatedata (byte array)	Alternative to headertemplate. Contains the used header template as a byte array. (optional)

rowtemplate (string)	Subreport template for dataset rows. The subreport is looped for each row in the dataset in the order rows appear in the dataset. Following parameters are passed to the rows subreport: - **column_1**, **column_2**, … **column_n** containing dataset cell data - parameters which names are same as column names, containing dataset cell data - same parameters that are passed to the header subreport, i.e. **columnheader_1**, **columnheader_2**, … **columheader_n** (optional)
rowtemplated ata (byte array)	Alternative to row template. Contains row template as a byte array. (optional)
visible (boolean)	Determines content visibility shown by the tag. See Setting Content Visibility. (optional)

4.3.3 Embeddedfile Tag

This tags inserts an embedded file (using OLE technology) into the report. The file can be of any type. In addition to the file, an icon image needs to be defined. The size of the embedded object icon is set based on the size of the icon image.

Attribute	Description
filedata (byte array)	Embedded file data as byte array. See available functions in expression language documentation in Binary Data Functions.
icondata (byte array)	Icon image as byte array. See available functions in expression language documentation in Binary Data Functions.. Icondata is not needed when using replace mode and replaceable embedded file is showing its content in the Word instead of icon (DrawAspect="Content"). (optional)
contenttype (string)	Embedded file media type (also called MIME type). This information can be defined as fixed (when the content type is known in report design), or fetched using function available in expression language documentation in Binary Data Functions.. List of media types: http://www.iana.org/assignments/media-types/media-types.xhtml
replace (boolean)	Enables the **replace mode**. This makes it possible to use an existing embedded file (a.k.a. **placeholder content**) in the template which is replaced by the actual embedded file. The replace mode enables to use Word styles and formatting, which will end up in the actual content of the report. The placeholder embedded file must be in the same or next paragraph of the tag. It's advisable to use as small files as possible as a placeholder embedded objects so that the size of the template remains small. Replace mode is activated with "true" or deactivated with "false". Default is false. (optional)

visible (boolean)	Determines content visibility shown by the tag. See chapter "5 Setting content visibility". (optional)

Example:

<#embeddeddata filedata=="qprEmbeddedFileData([objectid], 'embeddedata', '')" icondata=="httpFileData('http://url…/' + GetAttribute([objectid], 'embeddedfilemimetype', '') + '.png')" contenttype=="GetAttribute([objectid], 'embeddedfilemimetype', '')">

4.3.4 Expression Tag (exp)

This tag is for adding a result of an expression (formula) in the report. The defined expression is evaluated and the result is set in place of this tag.

Note that the tag may contain any attributes which are serving as arguments to the **value** expression (see examples).

Attribute	Description
value (object)	Shown value itself. Note that the values shown in the report are always technically strings (sequence of characters), so the value must be of a type that can be converted into a string, such as string, integer, double, datetime, boolean, or an array of any of the previous types. If the value is not a string, an implicit conversion to string is made. In case of datetime type, dateformat defined in the Reportsettings tag is used.
visible (boolean)	Determines content visibility shown by the tag. See Setting Content Visibility. (optional)
style (string)	Determines text style. See Defining Styles. (optional)

Examples:

<#expression value=="[variable1]">

<#expression value1=="5" formula=="2 * [value1] + 3"> (result is 13).

4.3.5 Image Tag

This tag is used to show an image in the report. Images can be got using QPR Web Service's **GetGraph** operation, which can be used to get graphical content from QPR system, such as EA/PD diagrams, Metrics dashboards or measure graphs. For more information about GetGraph, see http://kb.qpr.com/qpr2016-1/index.html?ws_getgraph.htm.

When getting images using http, many kind of errors may be encountered, such as the image is not

found, it's not available, or user rights restrict. In case an image cannot be got, the report run still succeeds. If the reason for the error is not known, it's possible to see the http error message by using the **loggingmode**. Note also the **noimagemessage** attribute which is used in case of these errors.

Image size in Word depends on its resolution and the properties listed below. Image size is calculated as follows:

- **dpi** defines image base size (dpi can be defined explicitly or used image's dpi value)
- **scaling** is applied after that (expanding or shrinking)
- **maxwidth** and **maxheight** restrictions are applied last (always shrinking)

In attributes dpi, scaling, maxwidth and maxheight expression it's possible to use following arguments representing information from the image (see an example below):

- **width** (int)
- **height** (int)
- **horizontaldpi** (decimal)
- **verticaldpi** (decimal)
- **format** (string) (e.g. png, jpeg, gif)

Note that image resolution (width and height in pixels) is determined by QPR Web Service, so that can be affected only using web service's parameters (parameter **options**).

If you need an image to appear always as a certain size regardless of its resolution, the **scaling** can be defined as a large enough number, and set **maxwidth** (or **maxheight**) to be the desired image size. This way the maximum width restriction is always applied and the size of the image is the desired.

Attribute	Description
imagedata (byte array)	Image as a byte array. See available functions in expression language documentation in Binary Data Functions.. Note that the type of the file needs to be bitmap. (optional)
object (string)	This is passed as a parameter **objectid** of **GetGraph** operation. (optional)
options (string)	This is passed as a parameter **options** of **GetGraph** operation. (optional)
dpi (decimal)	Determines the image size as dots per inches. If not defined, dpi information of the image itself is used. (optional)
scaling (decimal)	Image scaling as percentage value ("100" means no change). The scaling is used to change the size of the image in the report. E.g. value "200" doubles the size of the image. Scaling is applied before maximum width and height restrictions. (optional)
maxwidth (decimal)	Maximum image width in centimeters. If the image is wider than this maximum width, the image is scaled smaller to so that its width is this maximum width. (optional)

maxheight (decimal)	Maximum image height in centimeters. If the image is higher than this maximum height, the image is scaled smaller to so that its height is this maximum height. (optional)
crop (boolean)	Determines whether unnecessary borders are removed from the image. Needed if the image contains too much white (or other background color) space. If the whole image is white color it is removed. Either "true" or "false". Default is false. (optional)
backgroundcolor (string)	Defines the background color of the image for cropping unnecessary borders. The background color is defined as an RGB hexadecimals, e.g. "00FFb7". Default is white ("FFFFFF"). (optional)
replace (boolean)	Enables the **replace mode**. There is a possibility to use an existing picture (a.k.a. **placeholder picture**) in the template which is replaced by the actual picture (this is the replace mode). The replace mode makes it possible to use Word styles and formatting, which will end up in the actual picture in the report. The placeholder picture must be in the next paragraph of the Graph tag. The placeholder picture may be any picture, and its size don't matter. Replace mode is activated with "true" or deactivated with "false". Default is false. (optional)
noimagemessage (string)	A text appearing instead of the image if no image could be added to the report for some reason. The reason may be that QPR Web Service didn't return any image or there was nothing left of the image after the cropping. This message is the only case when **style** attribute is needed for this tag. For instance the message could be "No image is available". (optional)
visible (boolean)	Determines content visibility shown by the tag. See Setting Content Visibility. (optional)
style (string)	Determines text style. See Defining Styles. (optional)

Examples:

<#image object=="[diagramid]" options="graphtype=processlevel,witdh=1500" dpi="100" maxwidth="17.5">

<#image source="embedded" object=="[attachmentId]" options="embeddeddata" maxwidth="17.5" dpi="1.5*[horizontaldpi]">

<#image imagedata=="httpFileData('http://someurl…')">

4.3.6 Link Tag

This tag is used to add a hyperlink in the report (both QPR Portal and external links). If the link address is empty or not valid for any reason, no link is displayed by the tag.

Attribute	Description

address (string)	Hyperlink's url address or location. Note that the address must not be added as a Word's hyperlink format but as a text. QPR Portal links can be got using an expression and **GetPortalUrl** function.
text (string)	Link's displayed text.
screentip (string)	Link's screen tip (see **Insert hyperlink** window). (optional)
target (string)	Link's target (see **Insert hyperlink** window). (optional)
visible (boolean)	Determines content visibility shown by the tag. See Setting Content Visibility. (optional)
style (string)	Determines text style. See Defining Styles. (optional)

Example:

<#link address=="getportalurl([diagramid], '', '')" text="Link to QPR Portal" target="_BLANK" screentip="Click here is go to QPR Portal">

4.3.7 Loop Tag

This tag is for looping through content in following alternative ways:

- Set of QPR elements defined by a QPR Web Service query, or For more information of QueryObjects, see http://kb.qpr.com/qpr2016-1/index.html?queryobjects.htm.
- List of strings (using attribute **list**). The list itself is a string where items are separated using defined character.

The defined subreport is called for every looped object.

Word *sections* are not transferred from subreports to a mainreport.

Attribute	Description
template (string)	Template name for the looped content (without the file type extension in file system templates). For more information see Working with Report Templates. (optional)
templatedata (byte array)	Looped template file as a byte array. See available function from Binary Data Functions. (optional)
query (string)	This is passed as a parameter **query** of **QueryObjects** operation. "List" attribute is alternative. (optional)
criteria (string)	This is passed as a parameter **criteria** of **QueryObjects** operation. This filters the output set of objects. (optional)

sortby (string)	This is passed as a parameter **sortby** of **QueryObjects** operation. This determines the order of the looped objects. Note that without sorting the report may look different between runs. (optional)
attributes (string)	This is passed as a parameter **attributes** of **QueryObjects** operation. All the queried attribute values are added as parameters to the subreport. (optional)
options (string)	This is passed as a parameter **options** of **QueryObjects** operation. (optional)
loopparameter (string)	Defines the name of the parameter that is passed to the subreport containing the looped object id. Like other parameters, also this parameter needs to be defined in the subreport.
filter (exp. string)	Additional filtering for returned objects. This filtering is applied after QPR Web Service query results have been returned to QPR Word Reports. The filter is defined as an expression (it's not a QPR Web Service criteria), which must evaluate to boolean. The filter expression is evaluated for each returned object separately. If it is evaluated to false, the object is filtered out. All looped subreport parameters, and report parameters and expressions are available as arguments for the expression. This additional filtering may be used if the desired condition cannot be written using QPR Web Service's criteria. Using QPR Web Service's criteria is recommended because it offers better performance. (optional)
sortvalue (exp. string)	An expression which results are used to sort the returned objects. This sorting is applied after the QPR Web Service's sorting (which is determined by attribute **sortby**). All looped subreport parameters, (main)report parameters, (main)report variables are available as expression arguments (subreport parameters have a priority in name conflicts). Note that because sortvalue is an expression itself, only a single equation sign is usually used.
list (array)	List of items as an array that is looped through (instead of a QPR Web Service query). When this attribute is used, no QPR Web Service query is made, and thus the following attributes are ignored: query, criteria, sortby, attributes and options. (optional)
visible (boolean)	Determines content visibility shown by the tag. See Setting Content Visibility. (optional)
style (string)	Determines text style. See Defining Styles. (optional)
(all other attributes)	All other attributes are passed to the looped subreports as parameters. (optional)

Example:

```
<#loop template="Process report" loopparameter="diagramid" query=="'[' + [diagramid] +
'].ChildObjects(hierarchy=\"processlevels\")'" sortby="name" attributes="description,owner.name(as=
\"processowner\")" recursionlevel=="[recursionlevel] + 1">
```

In the example, "recursionlevel" is an additional parameter that is passed to the subreport. Also parameters "description" and "processowner" are passed to the subreport.

4.3.8 Parameter Tag (par)

This tag defines parameters for the report. Only parameters that are defined using these tags are used by the report. The report request url may contain other parameters which are omitted.

Attribute	Description
inputname (string)	Name of the parameter (url parameter name or a parameter name passed to subreport). Any tag attribute names mentioned in this documentation cannot be used as parameter or variable names (*reserved names*).
outputname (string)	Parameter name used in the report. If output value is not defined, output value is same as input value. Any tag attribute names mentioned in this documentation cannot be used as parameter or variable names (*reserved names*). (optional)
defaultvalue (object)	The default value is used when a parameter value is not passed to the report. If no default value is defined and no parameter value is passed, an error occurs. (optional)
validation (exp. string)	Validation expression for the parameter. There is an argument "value" available for the expression containing the parameter value. If the expression returns false, a validation error is thrown by the report.
validationmessage (string)	Error message to show, if the validation fails (see parameter "validation").

Example:

<parameter inputname="nameOfUrlParameter" outputvalue="nameInTheReport" defaultvalue="value1">

4.3.9 Reportsettings Tag

This tag is for defining report level settings. In subreports this tag is not used (it's ignored). The tag is optional. Only one tag of this type can exist in a template.

Attributes **validate** and **contentdispositiontype** are for advanced purposes, so usually they can be ignored.

Attribute	Description
name (string)	Name of the downloaded file. If this is not defined, file name is same as name of the template. (optional)

updatefields (boolean)	Defines that Word updates all the fields in the document (table of contents, references, etc.) when the document is opened. Alternatives are "true" or "false". Default is false. Note that Word shows an message when a document is opened that fields are to be updated, and gives also an alternative of not to do it. (optional)
dateformat (string)	Format for showed dates. More information about formatting http://msdn.microsoft.com/en-us/library/8kb3ddd4(v=vs.100).aspx. Default value is **dd-MM-yyyy**. (optional)
validate (boolean)	Defines whether a Word document validation is performed. Validation reveals if a Word document contains structural errors (the xml document is inconsistent with the schema). It seems that Word sometimes produces files that are not valid (!). If that kind of file is used as a template, it is probable that the output file produced by QPR Word Reports is not valid either and the validation fails. If the validation fails, an error message is shown and the report cannot be downloaded. By default validation is not performed. Validation is useful, if QPR Word Reports produces a file that Word is not able to open. The validation error message may provide further information of the problem. (optional)
contentdispositiontype (string)	Alternative values are "attachment" and "inline". This is the http response's content disposition's disposition type. Default value is attachment. For more information http://www.ietf.org/rfc/rfc2183.txt. (optional)

Example:

```
<#reportsettings name="Process report" updatefields="true" contentdispositiontype="inline"
validate="true">
```

4.3.10 Subreport Tag

This tag is for adding content of another report (subreport) to a report (main report). This tag is useful e.g. when there are parts in a report that are structurally similar (making it possible to reuse a subreport where a part is defined once).

Word *sections* are not transferred from subreports to a mainreport.

Attribute	Description
template (string)	Report template name (without file type extension in file system templates). For more information see Working with Report Templates. (optional)
templatedata (byte array)	Looped template file as a byte array. See available function to get data in Binary Data Functions. (optional)
visible	Determines content visibility shown by the tag. See Setting Content Visibility.

(boolean)	(optional)
style (string)	Determines text style. See Defining Styles. (optional)
(all other attributes)	All other attributes are are passed to the subreport as parameters. (optional)

Example:

<#subreport template="Report 2" param1="value1" param2="value2" param3="value3">

4.3.11 Variable Tag (var)

This tag is used to add variables to the report. One tag defines one variable. Variables can be referenced in other tags using expression arguments. Variables are processed in the order they appear in the report. It's a good practice to place the variables in a same place in the beginning of a report. Variable values cannot be changed during the report run – their value is calculated when the report run begins.

Variables are useful, when there is an expression which value is used several places in the report. Variables may also be used to simplify formulas, when part of a formula is defined as a variable.

There must not be a variable having a same name as a parameter in the same report. The scope of variables is the report template. If the variable must available in a subreport, the variable must be passed as a parameter to the subreport.

Attribute	Description
name (boolean)	Variable name. Any tag attribute names mentioned in this documentation cannot be used as parameter or variable names (*reserved names*).
value (object)	Variable value.

Example:

Report has tag **<#variable name="variable1" value="2">**. This variable can be used in other expressions, e.g. **<#expression value="3 + [variable1]">** (result is 5).

4.3.12 Direct Tags

It's possible to reference to report parameters and variables using tags which name equal parameter or variable name. These are called **direct tags**. For example if a report has a variable **var1**, a tag **<#var1>** can be used. Note that, the same result can be achieved using variable tag: **<#variable**

value=="[var1]">. Purpose of direct tags is to shorten tag syntax.

Note the following when using direct tags:

- Direct tags cannot contain any attributes, unlike all other tags always contain at least one attribute.
- In the direct tags, variable value is implicitly converted into string. E.g. date variables are converted using date format defined in the Reportsettings tag.
- In direct tags it's not possible to define styles using styles attribute.
- It's not possible to use visible attribute.
- When writing direct tags, note the letter case in parameter and variable names.

4.4 Expression Language

QPR Word Reports uses an expression language enabling all tag parameters can be written using expressions (formulas). Expressions makes it possible to define values dynamically (values that are calculated when the report is run). Expression parameters are defined using two equal signs instead of one. As an example, a parameter value may be static, such as **visible="false"**, or the parameter value may come from an expression, such as **visible=="[typename]='Activity'"**.

When writing tags, characters escaping is needed. There are three levels where escaping need to be done, explained by the table below.

Level	Purpose	Needed escaping	Example
1	QPR Web Services input	Described in the QPR Web Service documentation	QPR Web Service query: [PG].models(criteria="name=\"model1\"")
2	Expressions	Characters **** and **'** need to be written **** and **\'**	Previous query escaped for an expression: [PG].models(criteria="name=\\"model1\\"")
3	Template tags	Character **"** need to be written **\"**	Previous expression escaped for template tag: [PG].models(criteria=\"name=\\\"model1\\\"\")

Note that the escaping for expressions (level 2) is not needed, if the query is not in an expression.

Data from mainreport to subreport may be passed in any datatype, so it doesn't have to be converted to string. Parameters passed to a mainreport from a url are always strings.

If an expression returns a datetime value, it is implicitly converted to string if it's shown in Word report (reportsettings tag's dateformat parameter is used here).

See the QPR Web Services Extensions Expression Language topic for available functions and more information of the expression language.

4.5 Setting Content Visibility

All applicable tags support the parameter **visible**, which is used to remove the whole content that the tag is about to render. Available values are **true** (shown) or **false** (hidden). The value may be a static or a result of an expression.

When a tag is set to hidden, its expressions are not evaluated nor is related QPR data fetched. Thus the tag may contain errors, which are not emerged when the tag is hidden.

The tag is optional, and by default its value is true (the tag is shown).

4.6 Defining Styles

There are two ways to defined styles in the reports:

- **Apply styles in the template (static way):** All the styles applied in the template are in the final reports. Styles can also be defined for tags, meaning that the style is applied for the content produced by the tag. If a style of a text block is always same, it is easiest to use this way.
- **Use the "styles" parameter (dynamic way):** All applicable tags support the parameter **style**, which is used to dynamically determine the style of the paragraph or characters that is rendered by the tag. Value of the style tag is simply the name of the style. Both paragraph and character styles may be used.

Note that the style must exist in the Word file, so that it can be used dynamically. All the available styles in Word are not necessary stored in the Word file, although they are visible in the ribbon in the Word application. To make sure the style definition is in the Word file, the style needs to be used somewhere in the template. If you need to add text to the report that should not to be in the final report, it is possible to use Variable tags for this purpose because they disappear in the final report.

As an example, in recursive reports the heading level is determined by how deep the processing is in the recursive tree. For that you can add **style==`Heading ' + [recursionlevel]"**, where recursionlevel is a parameter or a variable increased by one on every subreport call when going deeper in the hierarchy.

4.7 Authentication and Data Security

When running reports there are three levels of security to consider:

- **QPR Web Service**: Data to the report is fetched using QPR Web Service, and thus available data is restricted by the rights of the user that is used to authenticate to the QPR Web Service. Authentication options to the QPR Web Service is explained above.
- **Report template files**: If the report templates are in the file system, there are no security restrictions to the report template files itself (the files cannot be accessed, but the result report may reveal the content of the template). If the report template files are in QPR Portal, normal QPR Portal access control is applied (**Publish to** section in the **Modify Word report**

template window, in the tab **Reports** > **Manage Reports**), and this way access to report templates can be restricted. This is useful when report templates itself contain classified information.

- **Links to reports**: Reports are run by referencing them using a url, and these links may be in the portal (e.g. in the reports dropdown list). It's advisable that users which don't have rights to a report, don't see the report link at all (otherwise they get a "not found" error message). This is actually a usability issue, not security issue.

4.8 Best Practices for Developing Reports

Use the QPR Web Services tester

Most of the errors are related to QPR Web Services query or the expression language syntax. That's why, before placing queries to QPR Word Reports templates, use the QPR Web Services tester to check that the query syntax is right.

Use descriptive names for templates and variables

Prefer names that describe content of report or variables.

Use subreports to reuse report parts

If there are parts in a report that are structurally same, they can be put in a subreport and reused by calling the subreport with different parameters.

Use variables to split complex tags

If there are complex formulas, parts of the formulas can be put in variables. This makes the report templates easier to read. Also, if there is same static formula in multiple tags, the formula may be put in a variable, so that it is calculated only once for the report offering performance advantages.

Prefer to run the logic in QPR Web Service

When possible, write logic to run in QPR Web Service instead of formulas, because it offers performance advantages when round trips to Web Service are reduced.

4.9 Migration from DWR Accelerators

QPR Word Reports has previously been known as the DWR accelerator. This chapter describes the changes needed when converting templates made for the DWR accelerator to the QPR Word Reports.

Changes between versions are described below. The version numbers prior to 2016.1.0 refer to the DWR Accelerator.

1. Migration from 1.5.0 to 2016.1.0

- {} syntaxt removed

2. Migration from 1.4.0 to 1.5.0

- New functions for the expression language.

3. Migration from 1.3.0 to 1.4.0

- QPR Word Reports is part of QPR Web Services Extensions. See changed web.config parameter names from QPR Web Services Extensions documentation.

4. Migration from 1.2.0 to 1.3.0

- Tag parameter names cannot be used as report parameter and variable names (they are called *reserved names*)
- Following changes have been made to web.config:
  - removed **wssessionparameter** and **wsaddressparameter**
  - changed **authenticationmode** to **qprauthenticationmode**
  - added **wcfsecuritymode** (see documentation)
- names and syntax of some functions have changed

5. Migration from 1.1.0 to 1.2.0

- replace all the files in IIS folder (transfer the needed settings in web.config)

6. Migration from 1.0.8 to 1.1.0

- change **idparameter** attribute of loop tag to **loopparameter**.
- replace expression signs $ with two equals signs.
- **{}** syntax for attributes and variables is deprecated. Use expression arguments instead ([] syntax).
- **formula** attribute of expression tag removed. Use **value** attribute to define an expression.
- **portalurl** tag removed. Use **getportalurl** function in the **expression** tag instead.
- **source** and **url** attributes of **image** tag removed. For getting images using a url or QPR embedded image files, use new **imagedata** attribute instead.
- remove application level parameter **wcfsecuritymode**.
- configure WCF binding security settings according to chapter 8. Installation and configuration.

5 QPR Web Views

QPR Web Views is a technology for QPR Suite to generate web pages (html pages) displaying content from a QPR system. QPR Web Views works as an IIS web application, and data is fetched from QPR using QPR Web Service interface. The web views are based on file templates containing html, css and javascript. QPR Web Views works like QPR Web Application Server (WAS), but QPR Web Views tags are QPR Web Service based and more generic purpose.

QPR Web Views contains following main features:

- Web views are generated as they are requested, so the web views always contain the latest data.
- Web views can have parameters affecting web view's contents.
- QPR Web Views contains an embedded formula calculation engine. All tags may be defined using expressions which values are calculated as the views are generated.
- Web views may get their content from other templates, and templates can also be called recursively (templates calling themselves). Templates makes it possible to assemble views from smaller parts.

QPR Web Views is based on the following concepts:

- Web views are based on **templates**, which are normal text files which contain html, css and javascript. When a view is generated, a template is taken as a basis, and in processing, content from QPR system is added to the template, forming the final web page.
- Templates contain **tags**, which instruct how the content is added to the page and how dynamic parts of the view are built.
- Tags contain **attributes**, which offer additional information of the tag.
- QPR Web Service is used through its **operations**. Operations called by QPR Web Views directly are GetAttributeAsString and QueryObjects. Also GetPortalUrl, GetGraphAsStream and GetBinaryDataAsStream can be utilized but they are called by the result web page from user browser. All QPR Web Service operations are documented under http://kb.qpr.com/qpr2016-1/index.html?functions.htm.
- **Loop** is a list of QPR system objects (elements) returned by QPR Web Service's QueryObjects operation. In a loop a template is called for every looped object.

QPR Web Views views can be embedded as part of QPR Portal or called from **External Reports Menu** (accelerator) dropdown list. It's able to automatically pass parameters, such as model or selected object to the report.

Web views are accessed using a url, which determines the template to be run (parameter **tpl**), and the url also passes all needed parameters to the template. The url depends on the server computer name and the path where QPR Web Views is installed in IIS. Note that parameter names are case sensitive.

An example url: http://SERVERNAME/QPRWebServicesExtensions/DynamicWebViews.ashx?tpl=viewname&parameter1=param1value&parameter2=param2value

QPR Web Views is tested to be compatible with QPR 2016.1.0.

5.1 Parameters

Following table contains QPR Web Views's parameters, which are configured in the **web.config** file of QPR Web Services Extensions.

Parameter name	Description
dwvtemplatesphysicalpath	Folder in the file system where template files are located. Also this folder needs to be created.
templatecaching	Determines whether template files caching is enabled (**true**) or disabled (**false**). Template caching means that templates are read from file system to server memory when the IIS web application starts. When template caching is enabled, pages are processed faster and disk load as reduces. Template caching should be enabled for production environments. Template caching is usually be disabled for development work, so that changes in templates can be seen immediately in result pages. When template caching is disabled, all templates are read into memory every time, when a page is requested.
qprwebapplicationname	Name of the QPR web application in IIS. In the default installation for QPR 2016, it is **QPR2016-1**. This parameter is not mandatory, but it should be defined, as it can be used by html content to reference QPR resources published in IIS (such as images and css files).

5.2 Working with Templates

QPR Web Views is based on text templates files. When a web view is called, the template is taken as a basis, and all tags in the template are processed.

Templates are stored in disk drive where there the IIS is running. Template files have extension **tpl**. The result content fetched using http have in http response header **Content-Type**: **text/html; charset=utf-8**.

Templates may be organized in folders. Folders inside folders are also supported. Template **root** location is determined by configuration setting **templatephysicalpath**.

There are two ways to reference templates:

- **absolute**: Absolute paths start with **/** and they contain the full path to the template starting from the templates root. Example: /**folderName1/folderName2/templateName**.
- **relative**: Relative paths don't start with /, and they reference to a template in relation to the template where the reference is made. Example: to reference a template in the same folder use syntaxt **templateName**.

When a web view processing fails, an html page showing an error message is displayed. The error message a likely to reveal, in which template and tag the error occurred. Also stack trace is shown, but that is helpful information only for developers.

5.3 Template Tags

Tags are used to add content to the web view. Syntax for defining tags is following:

<#tagname attribute1="attribute1 value" attribute2="attribute2 value" attribute3="attribute3 value">

If an attribute value is calculated using an expression (formula), two equals signs are used instead of one, e.g.

<#tagname attribute1=="1+1" attribute2="1+1"> (attribute1 value is integer 2, attribute2 value is string "1+1").

In the following sub chapters optional tag attributes are marked with "(optional)". Other tags are mandatory – if any of them is missing, an error is caused. In addition, for some of the mandatory attributes the value of the attribute must not be empty (i.e. **attributeName=""** is not allowed). If the attribute value comes from an expression, note that the expression may evaluate as empty (thus causing an error in non-empty mandatory attribute).

When processing a tag, attribute expressions are calculated from left to right. All template parameters, template variables and already calculated attribute values are available as expression **arguments** when later attribute expressions are calculated. An argument is kind of an input variable that can be used in an expression with syntax **[argumentName]**. E.g.

<#expression attribute1="3" value=="[attribute1] + 2"> (result is 5)

Tag specific arguments have a priority over template parameters and template variables, when there are same names used.

Attribute names should only contain characters **a**-**z**, **0**-**9** and **_**.

Values for all boolean valued tags (such as **visible**) may be defined as "true" or "false" or as a boolean valued expression, e.g. **visible=="[variable1]=1"** (the result of the equality comparison is of type boolean). Tag visibility is processed in the normal left to right order. When visibility tag is processed and its value is false, the tag processing stops at that point. This has an effect on processing performance. In addition, possible later expression errors don't emerge in that case.

Following abbreviated tag names can be used: **attribute**=**att**, **expression**=**exp**, **parameter**=**par** and **variable**=**var**. Tags must be written in lower case.

5.3.1 Attribute Tag (att)

This tag is used to get properties of QPR objects by using QPR Web Service's **GetAttributeAsString** operation (more information: http://kb.qpr.com/qpr2016-1/index.html?getattributeasstring.htm).

This tag uses **QueryObjects** instead of GetAttributeAsString, when there are multiple id's in the **object** attribute or attribute **followrelations** used (more information below). If QueryObjects is used, **separator** and **sortby** attributes are applied.

Attribute	Description
object	Object id or list of object id's separated by comma (,). If this contains only one id, it is passed as a parameter **objectId** of **GetAttributeAsString** operation. If this

	attribute is not provided or is empty, the tag will be evaluated to empty. (optional)
attribute	This is passed as parameter **attribute** of **GetAttributeAsString** or **QueryObjects** operation.
options	This is passed as parameter **options** of **GetAttributeAsString** or **QueryObjects** operation. (optional)
expression	An expression where the result is placed as an argument "value" (see the examples). (optional)
followrelations	Name of the relation attributes to follow to get another object or a set of other objects. If there are multiple subsequent relations to follow, the relations are separated by comma (,) (see the examples). (optional)
separator	Character(s) separating list of attributes of different objects. Empty values are not shown (i.e. no empties between separators). Default is a new line ( ). (optional)
sortby	Sorting if there are multiple objects returned. Default sorting is by name. (optional)
visible	Determines content visibility shown by the tag. See Setting Content Visibility. (optional)

Examples:

<#attribute object=="[objectId]" attribute="name">

<#attribute object="[measureId]" attribute="measure.value(series=\"ACT\",period=\"1 / 2015\")" expression="[value] * 2.54">

5.3.2 Dataset Tag

This tag is for showing information from a *dataset* in the web view. Dataset consist of multiple columns and rows. Dataset tags shows a **header template** (once) and **row templates** for each row in the dataset. Dataset tag has following attributes

Attribute	**Description**
data (dataset)	Dataset shown by the tag.
headertemplate (string)	Template for dataset header. The header template is shown once before the row templates. Header template can be used e.g. for showing a column names. Header template is optional, so if the parameter is left out, no header template is shown. Following type of parameters are passed to the header template: - **columnheader_1**, **columnheader_2**, … **columheader_n** containing names of the dataset columns.

	- **headerarray** containing column names as an object array Note that if the column names are known, the header can be set fixed in the web view and header template is not needed. (optional)
headertemplatedata (byte array)	Alternative to headertemplate. Contains the used header template as a byte array. (optional)
rowtemplate (string)	Template for dataset rows. The template is looped for each row in the dataset in the order rows are in the dataset. Following parameters are passed to the rows template: - **column_1**, **column_2**, … **column_n** containing dataset cell data - parameters which names are same as column names, containing dataset cell data - same parameters that are passed to the header template, i.e. **columnheader_1**, **columnheader_2**, … **columheader_n** - **dataarray** containing row's data as an object array - **headerarray** containing column names as an object array (optional)
rowtemplatedata (byte array)	Alternative to row template or rowtemplateexpression. Contains row template as a byte array. (optional)
rowtemplateexpression (string)	Expression defining a row template. Alternative to row template or rowtemplatedata. (optional)
footertemplate (string)	Template for dataset footer. The footer template contents is shown once after the row templates. Footer template can be used e.g. for adding table end tags. The footer template is optional, so if the parameter is left out, no footer template is shown. The footer template has the same parameters are the header template. Following type of parameters are passed to the footer template: **columnheader_1**, **columnheader_2**, … **columheader_n** containing names of the dataset columns. (optional)
footertemplatedata (byte array)	Alternative to footer template. Contains the used footer template as a byte array. (optional)
visible (boolean)	Determines content visibility shown by the tag. See Setting Content Visibility. (optional)

5.3.3 Expression Tag (exp)

This tag is for adding a result of an expression (formula) in the web view. The defined expression is evaluated and the result is set in place of the tag. Also static values can be used.

Note that the tag may contain any attributes which are serving as arguments to the **value** expression.

Attribute	Description
value	Value of the tag.
visible	Determines content visibility shown by the tag. See Setting Content Visibility. (optional)

Examples:

<#expression value=="[variable1]">

<#expression value1=="5" formula=="2 * [value1] + 3"> (result is 13).

5.3.4 Loop Tag

This tag is for looping through content in following alternative ways:

- Set of QPR elements defined by a QPR Web Service query (more information of QueryObjects, see http://kb.qpr.com/qpr2016-1/index.html?queryobjects.htm
- List of strings (using attribute **list**). The list itself is a string where items are separated using defined character.

The defined template is called for every looped object.

Following additional parameters are passed to the template:

- **loopindex** is an integer starting from zero
- **attributesasarray** contains all attribute values as an array

Attribute	Description
tpl	Template name for the looped content (without the file type extension in file system templates). For more information see Working with Templates. (optional)
tplexpression	Expression defining the template. Alternative to tpl or templatedata. (optional)
templatedata	Looped template file as a byte array. See available function from Binary Data Functions. (optional)
query	This is passed as a parameter **query** of **QueryObjects** operation.
criteria	This is passed as a parameter **criteria** of **QueryObjects** operation. This filters the output set of objects. (optional)
sortby	This is passed as a parameter **sortby** of **QueryObjects** operation. This determines the order of the looped objects. Note that without sorting the web views may look different between runs. (optional)

attributes	This is passed as a parameter **attributes** of **QueryObjects** operation. All the queried attribute values are added as parameters to the template. (optional)
options	This is passed as a parameter **options** of **QueryObjects** operation. (optional)
loopparameter	Defines the name of the parameter that is passed to the template containing the looped object id. Like other parameters, also this parameter needs to be defined in the template.
filter	Additional filtering for returned objects. The filtering is applied after QPR Web Service query has been run. The filter is defined as an expression (it's not a QPR Web Service criteria), and the expression must be evaluated to boolean. All looped template parameters are available as arguments for the expression. The filter expression is evaluated for each returned object separately. If it is evaluated to false, the object is filtered out. This additional filtering may be used if the filtering condition cannot be written using QPR Web Service's criteria. Using QPR Web Service's criteria is recommended because it offers better performance. (optional)
sortvalue	This is an expression which results are used to sort the returned objects. This sorting is applied after the QPR Web Service's sorting (that is determined by attribute **sortby**). All looped parameters, template parameters, template variables are available as expression arguments (template parameters have a priority in name conflicts). Note that because sortvalue is an expression itself, only a single equation sign is usually used.
list	List of items as an array that is looped through instead of a QPR Web Service query. When this attribute is used, no QPR Web Service query is made, and thus the following attributes are ignored: query, criteria, sortby attributes, and options. (optional)
visible	Determines content visibility shown by the tag. See Setting Content Visibility. (optional)
(all other attributes)	All other attributes are passed to the looped template as parameters. (optional)

Example:

<#loop template="ProcessReport" loopparameter="diagramid" query=="'[' + [diagramid] + '].ChildObjects(hierarchy=\"processlevels\")'" sortby="name" attributes="description,owner.name(as=\"processowner\")" recursionlevel=="[recursionlevel] + 1">

In the example, "recursionlevel" is an additional parameter that is passed to the template. Also parameters "description" and "processowner" are passed to the template.

5.3.5 Parameter Tag (par)

This tag defines parameters of the template. Only parameters that are defined using these tags are used by the template, i.e. the url may also contain other parameters that are omitted.

Attribute	Description
inputname	Name of the parameter (url parameter name or a parameter name passed to template)
outputname	Parameter name used in the template. Makes it possible to use a parameter with different name than the input name. If output value is not defined, output value is same as input value. (optional)
defaultvalue	The default value is used when a parameter value is not passed to the template. If no default value is defined and no parameter value is passed, an error occurs. (optional)

Example:

<parameter inputname="nameOfUrlParameter" outputvalue="nameInTheTemplate" defaultvalue="value1">

5.3.6 Templatesettings Tag

This tag defines template level settings. The tag is optional. Only one tag of this type can exist in a template.

Attribute	Description
dateformat	Format for shown dates. More information about formatting http://msdn.microsoft.com/en-us/library/8kb3ddd4(v=vs.100).aspx. (optional)
contenttype	Http response's Content-Type header when QPR Web Views data is requested. This setting only have an effect in the top level template. Default value is **text/html**. E.g. **application/xml** can be used for xml data. Available values: http://www.freeformatter.com/mime-types-list.html (optional)

5.3.7 Template Tag

This tag is for adding content from another template. This tag is useful e.g. when there are parts that are structurally similar (making it possible to reuse a template where a part is defined once).

Attribute	Description

tpl	Template name (without file type extension in file system templates). For more information see Working with Templates. (optional)
templatedata	Looped template file as a byte array. See available functions in expression language documentation in QPR Web Services Extensions Expression Language. (optional)
visible	Determines content visibility shown by the tag. See Setting Content Visibility. (optional)
(all other attributes)	All other attributes are are passed to the called template as parameters. (optional)

Example:

<#template template="Template2" param1="value1" param2="value2" param3="value3">

5.3.8 Variable Tag (var)

This tag is used to define variables in the template. Variables can be referenced in other tags using expression arguments. Variable tags are processed in the order they appear in the template. It's a good practice to place the variables in a same place in the beginning of a template. Variable values cannot be changed during the generation – their value is calculated when the run begins.

Variables are useful, when there is an expression which value is used several places in the template. Variables may also be used to simplify formulas, when part of a formula is defined as a variable.

There must not be a variable having a same name as a parameter in the same template. The scope of variables is the template. If the variable must available in other called templates, the variable must be passed as a parameter to the template.

Attribute	Description
name	Variable name
value	Variable value. May be a static value or calculated from an expression.

Example:

Template has tag **<#variable name="variable1" value="2">**. This variable can be used in other expressions, e.g. **<#expression value="3 + [variable1]">** (result is 5).

5.4 Expression Language

QPR Web Views uses an expression language enabling all tag parameters can be written using expressions (formulas). Expressions make it possible to define values that are calculated when the web view is generated. Expression valued parameters are defined using two equal signs instead of one. As an example, a parameter value may be

- static (i.e. not an expression), such as **visible="false"**, or
- calculated using an expression, such as **visible=="[typename]='Activity'"**.

See QPR Web Services Extensions Expression Language for available functions and more information of the expression language.

When writing tags, character escaping is required. There are three levels of escaping, explained by the following table.

Level	Purpose	Needed escaping	Example
1	QPR Web Services input	Described in the QPR Web Services documentation	QPR Web Services query: [PG].models(criteria="name=\"model1\"")
2	Expressions	Characters **** and **'** need to be written as **** and **\'**	Above query escaped for an expression: [PG].models(criteria="name=\\"model1\\"")
3	Template tags	Character **"** need to be written as **\"**	Previous expression escaped for template tag: [PG].models(criteria=\"name=\\\"model1\\\"\")

Note that the escaping for expressions (level 2) is not used, when the query is not in an expression.

Data from templates to other templates may be passed with any datatype, so data don't have to be converted to string. Parameters passed to a template from a url are always strings.

If an expression tag returns a datetime value, it is implicitly converted to string if it's shown in the web view. Templatesettings tag's dateformat parameter is used for this.

5.5 Setting Content Visibility

All tags support the parameter **visible**, which is used to hide the whole content the tag is about to render. Available values are **true** (shown) or **false** (hidden). Also visible parameters can be defined using expressions.

When a tag is set to hidden, its expressions are not evaluated nor is related QPR data fetched. The tag may thus contains errors, which don't emerge if the tag is hidden.

Visible tag is optional, and by default its value is true (the tag is shown).

5.6 Calling QPR Web Services from Javascript

Some QPR Web Views based solutions need to call QPR Web Services from javascript. The following wrappers are available for usually needed QPR Web Services operations. It's recommended to use the wrappers instead of calling QPR Web Services directly.

Web Services operation	Description
GetBinaryData.ashx	Wrapper for QPR Web Services' GetBinaryData operation (http://kb.qpr.com/qpr2016-1/index.html?getbinarydata.htm). Parameters: • wssession • objectid • attribute • options
GetGraph.ashx	Wrapper for QPR Web Services' GetGraph operation (http://kb.qpr.com/qpr2016-1/index.html?ws_getgraph.htm). Parameters: • wssession • objectid • options
GetPortalUrl.ashx	Wrapper for QPR Web Services' GetBinaryData operation (http://kb.qpr.com/qpr2015-1/index.html?getportalurl.htm). Parameters: • wssession • objectid • viewname • options

Example:

http://localhost/QPRWebServicesExtensions/GetBinaryData.ashx?
objectid=PG.12345.6789&attribute=embeddeddata&options=

6 QPR Reports Menu

Reports Menu is a dropdown menu for QPR Portal to open reports and web pages that can be referenced using urls. The menu is able to add context related information automatically to the url as parameters, such as model, diagram and object id.

Report items in the menu are configured in the file C:\DWV templates\ReportsMenu \ReportsMenuConfiguration.xml (the location may vary).

6.1 Reports Configuration

Report items in the menu are configured in the file **C:\DWV templates\ReportsMenu \ReportsMenuConfiguration.xml** (note that the location may be different). The configuration is an xml file having a root tag **reportsmenu** and subtags **menuitem** for individual report items. The menuitem tag has menu item settings as subtags which are described in the following table.

Settings can be defined using static string values or expressions. If the value is defined using an expression following attribute is added to the settings tag: **expression="true"**.

Note that the setting tags must be in the order that they are defined in the following table. Settings that are not mandatory, can be left out.

Parameter	Description
reportname (string)	Visible name of the report in the dropdown menu.
reporturl (string)	Url to the report. Characters **?** and **&** are automatically added to the end of the url if needed for appending url parameters. To build SSRS report urls, see http://msdn.microsoft.com/en-us/library/ms153586(v=sql.110).aspx.
availabletabs (string)	List of QPR Portal tabs where the report is available. Available tab names can be found in http://kb.qpr.com/qpr2016-1/index.html?getportalurl.htm (see "Also Portal internal view names..."). The tabs need to be defined in lower case. If this parameter is empty, the report is visible in all tabs. Following default tabs name are available: pgplugin_processmaps, pgplugin_navigator, pgplugin_actions, pgplugin_organizations, scplugin_scorecards, scplugin_strategymaps, scplugin_analysis, scplugin_navigator, scplugin_reports, scplugin_actions, discussion, byuser, bytime, actionanalysis
passparameters (string)	List of parameters that are passed to the external report (see available parameters in chapter 4. Portal context parameters). Passed parameters may be restricted, because e.g. SSRS gives an error when parameters are passed that are not defined in the report.
order (integer)	An integer which determines order of reports in the menu. The reports are shown in the ascending order. If there are multiple reports with a same order number, the alphabetical order of the reportName parameter determines the ordering (secondary sorting).

waitanimation (boolean)	Determines whether to use *report loading wait animation*, which is able to improve user experience. The report source needs to support this feature (DWR and DER does). Alternatives **true** and **false**. The animation may only be used if the report access url has same DNS name and port than QPR Portal. This is because the disappearance of the animation is based on a cookie that is set by report providing server when the report is ready. When the cookie is received from the response of the report providing server, the loading animation is removed. If the loading animation doesn't work correctly for any compatibility issue, disable it (set to false).
visibility (string)	Determines the visibility status of the report item in the menu. Available statuses are: • **visible**: The report is visible and can be opened. • **disabled**: The report is visible in the menu in a grey color, and it's not openable. This status can be used to imply, that there is some condition preventing from running the report, such as the requqired type of element is not selected. • **hidden**: The report item is not visible in the menu at all.
visiblemessage (string)	Tooltip text shows for visible report items when cursor is moved over the report item. The text may contains html code.
disabledmessage (string)	Tooltip text shown for disabled report items when cursor is moved over the report item. The text may contains html code.
accessrights (boolean)	Determines if user has access to see the report item as boolean value ("true" if there are rights to see the report item).
[OpenWindow parameters]	QPR Portal's **OpenWindow** function parameters. These parameters determine e.g. size and position of the window. Available parameters: • target (string) • width (integer) • height (integer) • scroll (boolean) • preventcaching (boolean) • x (integer) • y (integer)

6.2 Portal Context Parameters

The following table lists information that can be passed to reports as url parameters.

Parameter	Description

modelid	PD/EA or Metrics model id
diagramid	Diagram id (i.e. process level). Note that for the top diagram level, the id is "PG.<modelid>.0" which represents the model object.
objectid	Object in details pane.
tab	Name of the tab where the report is run.
zoom	Zoom percentage. Only available in PD/EA tabs.
viewid	View id
wssession	QPR Web Service session id. External report can use this to login to QPR Web Service.
portalsessionid (SES)	QPR Portal session id. Note that this is different than QPR Web Service session id. Portal session id is needed, when the url refers to the QPR Portal itself and no Windows authentication is used. QPR Portal session id is added to the url as a parameter named SES.
currentuserid	Current user id.
scorecardid	Scorecard id. Only available in Metrics.
seriesid	Series id. Only available in Metrics.
periodid	Period id. Only available in Metrics.

7 QPR Web Services Extensions Expression Language

This chapter describes an expression language used in several QPR Suite extensions. Expressions (formulas) make it possible to define values dynamically, i.e. values are calculated at runtime.

The evaluation process is case sensitive meaning parameter and function names must be written with right case letter. (Actually all function names are not case sensitive, but it's easiest to consider they are).

Expressions can be combined using **operators**. Expression's priority order starting from largest priority is: primary, unary, power, multiplicative, additive, relational, logical **and**, logical **or**. Following operators can be used:

- Logical or: **or**, **||** (these are equivalent)
- Logical and: **and**, **&&** (these are equivalent)
- Relational: **=**, **!=**, **<>**, **<**, **<=**, **>**, **>=** (!= and <> are equivalent)
- Basic calculations: **+**, **-**, *****, **/**, **%**
- Bitwise: **&** (bitwise and), | (bitwise or), **^** (bitwise xor), **<<** (left shift), **>>** (right shift)
- Unary: **!**, **not**, **-**, **~** (bitwise not)
- functions: **Abs(1)**, **doSomething(1, 'dummy')**

In expressions, characters **** and **'** need to be written as **** and **\'** (i.e. escaped).

The expression language supports any .Net 4.5 datatype. Following table contains examples, how to write literals of basic datatypes.

Data type	Example
integer (int)	123456
double	123.456, 0.123
boolean	true, false
string	'Hello world!'

When operating with dates, all dates returned by QPR Web Services are strings formatted in XML date format (yyyy-MM-ddTHH:mm:ss.fffzzz). They can be converted to datetimes using function **StringToDate**.

In the expressions any unicode characters can be defined using syntax **\uxxxx**. For example, new line is **\u000D\u000A** (carriage return and line feed characters used in Windows systems).

7.1 Arithmetic Functions

Function	Description	Example expression	Result
Abs	Returns the absolute value of a specified number.	Abs(-1)	1
Acos	Returns the angle whose cosine is the specified number.	Acos(1)	0 (double)
Asin	Returns the angle whose sine is the specified number.	Asin(0)	0 (double)
Atan	Returns the angle whose tangent is the specified number.	Atan(0)	0 (double)
Ceiling	Returns the smallest integer greater than or equal to the specified number.	Ceiling(1.5)	2 (double)
Cos	Returns the cosine of the specified angle.	Cos(0)	1 (double)
Exp	Returns e raised to the specified power.	Exp(0)	1 (double)
Floor	Returns the largest integer less than or equal to the specified number.	Floor(1.5)	1 (double)
IEEERemainder	Returns the remainder resulting from the division of a specified number by another specified number.	IEEERemainder(3, 2)	-1 (double)
in	Returns whether an element is in a set of values.	in(1 + 1, 1, 2, 3)	true
Log	Returns the logarithm of a specified number.	Log(1, 10)	0 (double)
Log10	Returns the base 10 logarithm of a specified number.	Log10(1)	0 (double)
Max	Returns the larger of two specified numbers.	Max(1, 2)	2
Min	Returns the smaller of two numbers.	Min(1, 2)	1
Pow	Returns a specified number raised to the specified power.	Pow(3, 2)	9 (double)
Random	Returns a random value between 0 and 1 (the value can be 0 but cannot be 1).	Random()	0,358024762
Round	Rounds a value to the nearest integer or specified number of decimal places.	Round(3.222, 2)	3.22 (double)
Sign	Returns a value indicating the sign of a number.	Sign(-10)	-1
Sin	Returns the sine of the specified angle.	Sin(0)	0 (double)

Sqrt	Returns the square root of a specified number.	Sqrt(4)	2 (double)
Tan	Returns the tangent of the specified angle.	Tan(0)	0 (double)
Truncate	Calculates the integral part of a number.	Truncate(1.7)	1

7.2 String Functions

Function	Parameters	Description
CharAt (string)	• input string • index number (int)	Return character of the **index number** position in the **input string**. The indexing starts from zero.
Contains (boolean)	• first string • second string	Return true if the **first string** contains the **second string**; otherwise false.
EndsWith (boolean)	• first string • second string	Return true if the **first string** ends with the **second string**; otherwise false.
IndexOf (int)	• first string • second string • start index (int)	Return the index number of the first occurrence of the **second string** in the **first string**. Indexing starts from zero. If the **start index** is provided, the search is started from that index.
LastIndexOf (int)	• first string • second string • index (int)	Return the index number of the last occurrence of the **second string** in the **first string**. Indexing starts from zero. If the **start index** is provided, the search is started from that index (calculated from the start of the string) towards the beginning of the string.
Length (int)	• input string	Return the number of characters in the **input string**.
Replace (string)	• first string • second string • third string	Replaces all occurrencies of the **second string** with the **third string** in the **first string**. Example**:** **Replace('abcd', 'b', 'e')** gives **aecd**.
StartsWith (boolean)	• first string • second string	Return true if the **first string** starts with the **second string**; otherwise false.
Substring (string)	• input string • start index (int) • length (int)	Returns a substring of the **input string** starting from the **start index**. If the **length** is provided, the returned string contains maximum of that number of characters.
ToLower (string)	• input string	Return a string where all the characters of the **input string** have been converted to lower case characters.

ToUpper (string)	• input string	Return a string where all the characters in the input string have been converted to upper case characters.
Trim (string)	• input string • string array	Removes spaces from the start and end of the **input string** (if **string array** is not provided). If the string array is provided, the string in the array are removed from the input array.

7.3 Datetime Functions

Custom function	Parameters	Description
AddMilliseconds (datetime)	• datetime • milliseconds (double)	Adds the specified number of milliseconds to a datetime.
AddMonths (datetime)	• datetime • number of months (int)	Adds the specified number of months to a datetime. Negative number of months means number of months to subtract. This function takes into account years, so e.g. when adding one month to a date in December, the result is January in the next year.
AddTimespan	• datetime • timespan	Adds the specified timespan to a datetime.
CompareDates (int)	• datetime 1 • datetime 2	Compares two datetimes and returns zero when the datetimes equals, greated than zero if the former is later, and less than zero if the former is earlier.
CurrentDateTime (datetime)	[none]	Returns a datetime representing the current date and time as UTC. As there are no parameters, use syntax **CurrentDateTime()**
FromMetricsDateFormat (datetime)	• numerical value (string, double, int)	Returns a date that this the provided number of days from 1.1.1900 00:00:00 UTC. QPR Metrics date values uses this date format.
GetDate (int)	• datetime	See (1) below (property: Date)
GetDay (int)	• datetime	See (1) below (property: Day)
GetMonth (int)	• datetime	See (1) below (property: Month)
GetTicks (int)	• datetime	See (1) below (property: Ticks)
GetYear (int)	- datetime	See (1) below (property: Year)
NewDatetime	• year (int)	Returns a new datetime with the following parameter values. The date is speficied as UTC.

	• month (int) • day (int) • hour (int) • minute (int) • second (int) • millisecond (int)	- year: 1 - 9999 - month: 1 - 12 - day: The day 1 through the number of days in month - hour: 0 - 23 - minute: 0 - 59 - second: 0 - 59 - millisecond: 0 - 999
SubtractTimespan	• datetime • timespan	Substracts the specified timespan from the datetime.
Timespan (datetime)	• days (int) • hours (int) • minutes (int) • secords (int) • milliseconds (int)	Creates a new timespan. A timespan represent an interval between two datetimes.
ToMetricsDateFormat (double)	• datetime	Returns number of days between 1.1.1900 00:00 UTC and the provided datetime. QPR Metrics handles date values using this format. Also the function is needed for criteria in Web Service's QueryObjects, e.g. **'createddate>' + ToMetricsDateFormat(CurrentDateTime())**

(1) DateTime properties in .Net Framework 4.5: https://msdn.microsoft.com/en-us/library/system.datetime_properties(v=vs.110).aspx

7.4 Array Functions

An array is a list of objects of any type, such as strings or integers. Arrays may contains same valued items multiple times.

Function	Parameters	Description
ArrayExcept (array)	• array • array	Returns an array which contains items that are in the first array but not in the second array.
ArrayIntersect (array)	• array • array	Returns an array which contains items that are in both provided arrays.
ArrayReverse	• array	Reverses an array, i.e. first element becomes last and so

(array)		on.
ArraySize (int)	• array	Calculates number of items in an array.
ArraySort (array)	• array	Sorts an array
ArrayUnion (array)	• array • array	Joins two arrays. Duplicates are not removed. If only one parameter is provided, that parameter is expected to contain an array, item of which are arrays. In that case all the items of all arrays are joined.
ArrayUnique (array)	• array	Returns an array where duplicate values are removed. Order of items is preserved.
ArrayWhere (array)	• array • expression (string)	Filters out all items in an array, where the expression is evaluated as false. The expression must return boolean value. Available arguments in the expression: - **item** (item in the array) - **index** (iteration order number starting from 0)
Average (double)	• array	Calculates an average of array items. Array must contains numerical data.
Concatenate (string)	• array • separator (string)	Concatenates items of an array into a string separated by a provided separator. Array must contain items that can be converted into string.
IndexOfAnyInArray (int)	• array • array	Returns an index number of the any of objects of the second array in the first array. The second array objects are searched from left to right. First object array starting from index 0. If the first array doesn't contain any of objects in the second array, -1 is returned.
IndexOfInArray (int)	• array • item to search (object)	Returns an index number of an object in the array. First object array starting from index 0. If object is not found, -1 is returned.
ItemAt (object)	• array • int	Returns an item in the provided index number.
Median (double)	• array	Returns the median of the provided array.
Minimum (double)	• array	Returns the smallest number in the provided array.
Maximum (double)	• array	Returns the largest number in the provided array.
OnlyValue (object)	• array	Returns the only value of the array. Throws an exception if the array contains more than one value. Return null if the array contains no values.
Sum (double)	• array	Calculates a sum of array items. Array must contains

		numerical data.
Transform (array)	• array • expression (string)	Transforms every item of an array to another array using the provided expression. The expression has following arguments: - **value** which gets the value of the item. - **index:** iteration order number starting from 0 E.g. **transform([inputarray], 'substring([value], 5)')** returns first five characters from every string of the array.

7.5 Dictionary Functions

Function	Parameters	Description
Dictionary (dictionary)	• keys (keys array) • values (object array)	Creates a new dictionary based on two arrays. The two arrays must not be null, and must be the same length.
GetDictionaryValue (object)	• dictionary • key (string) • default value (object)	Returns a value from a dictionary based on key. If the key is not found and a default value is defined, the default value is returned. If the key is not found and a default value is not defined, an exception is thrown.
ContainsValue (boolean)	• dictionary • key (string)	Returns true if the provided dictionary contains the provided value; otherwise false.

7.6 Conversion Functions

Function	Parameters	Description
ArrayToDataset (dataset)	• array • column name (string)	Converts an array to a dataset. The dataset has one column which name is given as a second parameter.
ConvertToDouble (double)	• input value (string) • value in error (double)	Converts a string value to a double value. The input value can also be in other formats that can be converted to double. The second parameter is optional, and it's returned if the conversion fails. If it's not defined and the conversion fails, an error is thrown.
ConvertToInt (int)	• input value	Converts a string value to an integer value. The input

	(string) • value in error (int)	value can also be in other formats that can be converted to integer. The second parameter is voluntary, and it's returned if the conversion fails. If it's not defined and the conversion fails, an error is thrown.
ConvertToString	• input value (object) • value in error (string)	Converts an object to string. The second parameter is optional, and it's returned if the conversion fails. If it's not defined and the conversion fails, an error is thrown.
DatasetColumnToArray (array)	• dataset • column name (string)	Converts a column of a dataset to an array. The column is referenced using the column name.
DatasetRowToArray (array)	• dataset • row number (int)	Converts a row of a dataset to an array. The row is referenced using the row number (the first is 1).
DateToString (string)	• date to convert (datetime) • local time (boolean) • dateformat (string)	Converts a datetime to a string. The local time parameter is optional; if omitted, the string is assumed to represent local time. If the third parameter is omitted, the string is interpreted as XML schema date format. More information of how to construct the dateformat: https://msdn.microsoft.com/en-us/library/8kb3ddd4(v=vs.110).aspx
FileDataToString (string)	• file data (byte array)	Converts a file provided as a byte array into string using UTF8 encoding. If null is provided, null is returned. If zero valued array is provided, an empty string is returned.
HtmlEncode (string)	• string	Performs an HTML encoding. More information: https://msdn.microsoft.com/en-us/library/system.net.webutility.urlencode(v=vs.110).aspx
InstanceIdFromFullId (string)	• element full id (string)	Extracts an instance id from a full id. The function returns an empty string, if the instance id cannot be extracted. Example: **InstanceIdFromFullId('SC.6346904.5290187.4353690')** returns 4353690
ItemsToArray (array)	• object 1 • object 2 • …	Converts provided parameters as an array. There must be at least one parameter. Number of parameters is unrestricted.
ModelIdFromFullId (string)	• element full id (string)	Extracts a model id from a full id. The function returns an empty string, if the model id cannot be extracted. Example: **ModelIdFromFullId('SC.6346904.5290187')** returns 6346904

NumberToString	• value (int, double) • format (string) • value in error (string)	Converts a numerical value to string. This function can be used when there are special requirements related to e.g. thousand separators. Behaviour of this function depends on the server's locale settings. If the convertion fails and the value in error is defined, that value is returned. Otherwise an error is thrown. More information about formattings: • https://msdn.microsoft.com/en-us/library/dwhawy9k(v=vs.110).aspx#NFormatString • https://msdn.microsoft.com/en-us/library/0c899ak8(v=vs.110).aspx
ObjectIdFromFullId (string)	• element full id (string)	Extracts an object id from a full id. The function returns an empty string, if the object id cannot be extracted. Example: **ObjectIdFromFullId('SC.6346904.5290187')** returns 5290187
StringToArray (array)	• string to convert • separator (string) • remove empties (bool)	Converts a string to an array using provided separator string. If remove empties is true, there are no empty strings in the output array.
StringToDate (datetime)	• string (string to convert) • local time (boolean) • string (dateformat)	Converts a string to a datetime. The local time parameter is optional; if omitted, the string is assumed to represent local time. If the third parameter is omitted, the string is interpreted as XML schema date format. More information of how to construct the dateformat: https://msdn.microsoft.com/en-us/library/8kb3ddd4(v=vs.110).aspx
UrlEncode (string)	• string	Performs a URL encoding. More information: https://msdn.microsoft.com/en-us/library/4fkewx0t(v=vs.110).aspx

7.7 QPR Web Services Functions

Function	Parameters	Description
BelongsToGroup (boolean)	• group name (string)	Returns true if the current user belongs to the provided group; otherwise false. QPR Web Services query is made to get the information. The current user is the user account which is logged to the QPR Web Services when the query is made. False is returned if the provided group doesn't exist. Upper and lower case characters need to be written correctly.

		Example: **BelongsToGroup("Administrators")**
CurrentUser (string)	• attribute name (string)	Returns any attribute of the current user object. If no attribute is provided, user id is returned (attribute "id"). Example: **CurrentUser()**
GetAttribute (string)	• objectid (string) • attribute (string) • options (string)	QPR Web Service's GetAttributeAsString operation, see http://kb.qpr.com/qpr2016-1/index.html?getattributeasstring.htm.
GetAttributes (string array)	• objectid (string) • attributes (string) • options (string)	Returns multiple attributes of a single object as an array. Attributes are defined as comma separated. Based on QPR Web Service's QueryObjects operation.
GetPortalUrl (string)	• objectid (string) • view (string) • options (string)	QPR Web Service's GetPortalUrl operation, see http://kb.qpr.com/qpr2015-1/index.html?getportalurl.htm.
LatestValuePeriod (string)	• objectid (string) • series symbol or series id (string)	Returns the id of the period that contains the latest (newest) measure value. Return an empty string, if no period contains a value. This information can be used e.g. in web service attribute **'prettyvalue(period=' + [periodid] + ')'**
QueryObjects (string array)	• query (string) • filter (string) • sortby (string) • attributes (string) • options (string)	Executes QPR Web Service's QueryObjects and returns results as a string array. Only one attribute should be defined for "attributes" parameter.
QueryObjectsAverage (double)	• query (string) • filter (string) • attributes (string) • options (string)	Average of queried objects numerical attributes. Only one attribute is defined in "attributes". Error is thrown if attribute values are not numerical.
QueryObjectsConcatenate (string)	• query (string) • filter (string) • sortby (string) • attributes (string) • options (string) • separator (string)	Concatenates attribute values of all queried objects using defined separator. Only one attribute is defined in "attributes".

QueryObjectsCount (int)	• query (string) • filter (string) • options (string)	Returns number of objects returned by QueryObjects. "sortby" and "attributes" cannot be defined as they don't affect the result.
QueryObjectsFirstAttribute (string)	• query (string) • filter (string) • sortby (string) • attributes (string) • options (string)	Attribute value of the first object of queried objects. Only one attribute is defined in "attributes".
QueryObjectsSum (double)	• query (string) • filter (string) • attributes (string) • options (string)	Sum of queried objects of numerical attributes. Only one attribute is defined in "attributes". Error is thrown if attribute values are not numerical.
QueryObjectsUnique (string)	• query (string) • filter (string) • sortby (string) • attributes (string) • options (string) • separator (string)	Concatenates unique attribute values of all queried objects using defined separator. Only one attribute is defined in "attributes".
SubAttributesAsArray (array)	• objectid (string) • attribute name (string) • subattribute name (string) • filter expression (string)	Returns attribute values (as an array) which are presented in more complex structures. This type of attributes are e.g. *graphicalproperties*, *customattributetypes* and *properties*. The filter expression is for selecting desired rows. In the filter expression there are following arguments: - all sub tag attributes (with their tag names) - argument **ordernumber** having values 0, 1, … for xml tag order number. Example, following graphicalproperties attribute: <graphicalproperties> <symbol id="920256947" x="790" y="190" width="400" height="30" instanceid="2029031131" /> <symbol id="2009602777" x="230" y="640" **width="200"** height="40" **instanceid="0"** /> <symbol id="368985118" x="730" y="40" width="100" height="60" instanceid="369716419" /> </graphicalproperties> Calling: **SubAttributesAsArray([instanceid], 'graphicalproperties', 'width', '[instanceid]=0')**

		returns 200.
SubAttributesAsDataset (dataset)	• objectid (string) • attribute name (string) • columns (array)	Returns attributes which are presented in more complex structures as a dataset. This type of attributes are e.g. graphicalproperties, customattributetypes, properties and hotspots. Dataset columns (names of the sub attributes) are listed as an array (in lower case).

7.8 Dataset Functions

Dataset is a table like object containing multiple named columns and multiple rows. Dataset can contains zero rows, but at least one column must exist. Dataset may contain any type of data. Dataset is like a table in relational database or a worksheet in Excel.

Usually dataset functions don't change the input dataset, but create new dataset (exceptions are mentioned). Thus the input datasets may also be used in other functions.

Dataset functions get their idea from SQL language.

Function	Parameters	Description
AddColumn (dataset)	• dataset • columnName (string) • expression (string) • compare mode (boolean)	Adds a new column to the dataset. All other column values are available as arguments in the expression. Also there is an argument **rowordernumber**, which is the row order number starting from 0. In the compare mode (true), there are also values of the previous and next rows available as arguments with suffixes **_previous**, **_current** and **_next**. For example if there are columns **column1** and **column2**, and adding a new column **column3**, there are arguments column1_previous, column2_previous, column3_previous , column1_ current, column2_current, column1_next and column2_next.
AddDatasetRow (dataset)	• column 1 value (object) • column 2 value (object) • …	Adds a new row to a dataset with the provided values. A row is added to the provided dataset, i.e. no new dataset is created.
BuildHierarchy (dataset)	• dataset • instance id column name (string) • object id column name (string) • parent object id column name (string)	Builds an element hierarchy based on data containing element instances. The data is provided in a dataset, where there are following columns: instance id, object id and parent object id. The object id can be derived from instance id. The function constructs a new dataset where lines are in the hierarchy order. The function can handle situations where also other than the bottom level objects have been

	• Hierarchy id column name (string) • Hierarchy parent id column name (string) • level column name (string) • filtering formula (string) • sorting formula (string)	instantiated; it means that the result hierarchy will contain same instances multiple times. Following new columns are added: - hierarchy id (explained below) - hierarchy parent id (referring to the hierarchy id) - level (top level is 0, the level directly below top is 1, …) The **hierarchy id** means an id that is unique for the whole hierarchy. To enable this following technique is used: If there is an instance id PG.x.y.z which appears multiple times, the first hierarchy id is PG.x.y.z, the second is PG.x.y.z.1, the third is PG.x.y.z.2, and so on.
CreateDataset (dataset)	• column names (string array)	Creates a new dataset with provided column names. The created dataset contains no rows.
DatasetCell (object)	• dataset • column name (string) • row number (int)	Gets an item from a dataset from the named column and from the named index (first row is number 1). Returned type is the type of the cell.
DatasetSize (int)	• dataset	Number of rows in a dataset. If dataset is null, -1 is returned.
Distinct (dataset)	• dataset	Removes duplicate rows, i.e. rows that contains identical data.
Except (dataset)	• dataset1 • dataset2	Result dataset contains rows that are in the first dataset but not in the second dataset. The datasets must have same columns.
From (dataset)	• query (string) • filter (string) • sortby (string) • attributes (string) • options (string)	Returns QPR Web Service's QueryObjects result as a dataset. Columns get their names from the result data. Column names can be changed with QueryObjects syntax **attribute(as="ColumnName")**.
FullOuterJoin (dataset)	• left dataset • right dataset • matching expression (string)	Full outer join of two datasets. The expression have all column names as arguments. The datasets cannot have columns with same names.
GroupBy (dataset)	• dataset • list of grouped columns (string array) • list of combined column names (string	Groups the dataset. Parameters: - 1. parameter is the dataset to group - 2. parameter is an array of columns to group - 3. parameter is an array of combined column names

	array) • list of combine expressions (string array)	- 4. parameter is combine expression (e.g. count, sum, average or number of rows). All expressions have as arguments an array of objects to combine. Parameter names equal to column names. Example: GroupBy([dataset1], ItemsToArray('modelname', 'typename'), ItemsToArray('count', 'objectnames'), ItemsToArray('ArraySize([name])', 'Concatenate([name])'))
InnerJoin (dataset)	• left dataset • right dataset • matching expression (string)	Inner join of two datasets. The expression have columns names as available arguments. The datasets cannot have columns with same names.
Intersect (dataset)	• dataset1 • dataset2	Intersect of two datasets. Result dataset contains only those rows that are in both datasets. The datasets must have same columns.
LeftJoin (dataset)	• left dataset • right dataset • matching expression (string)	Left join of two datasets. Expression have columns names as available arguments. The dataset cannot have columns with same names.
Matrix (dataset)	• dataset • row expression (string) • column expression (string) • value expression (string) • grouping expression (string) • row column name (string)	Builds a matrix based on the provided dataset. Row and column expressions are executed for each row of the dataset, and the data is positioned to rows and columns in the matrix based on the row and column expression values. The value for the input dataset row is determined by the value expression. The grouping expression has an input argument "value", containing an array of all calculated value expressions for the matrix cell. *Row column name* defines the column containing matrix row names. Columns are sorted as an alphabetical order (except the matrix row name is the first).
MetricsMeasureValues (dataset)	• element id's (string array) • series symbols (string array) • value column names (string array) • value attribute name (string)	Returns Metrics measure values as a dataset from multiple periods (dataset rows) and multiple series or measures (dataset columns). Each period id will have a separate row, i.e. data from different measures are combines in a same row if the measures have a same period level. Element id's, series symbols and value column names in the result dataset are provided as a separate arrays. All arrays size must be the same The result dataset will have **columns**, **periodid**, **startdate**, **enddate** and **periodname**. Valid value attribute names are **value** and **prettyvalue**

		(these are from QPR Web Service's *values* attribute). Data in result dataset are strings, and in case "value" attribute is used, they can be converted to numeric.
RemoveColumns	• dataset • columnNames (string array)	Removes the specified columns from a dataset. Example: RemoveColumns([dataset1], ItemsToArray('name'))
RightJoin (dataset)	• left dataset • right dataset • matching expression (string)	Right join of two datasets. Expression have columns names as available arguments. The dataset cannot have columns with same names.
SortBy (dataset)	• dataset • sorting definition (string)	Sorts the dataset. Sorting is defined as comma separating the sorted columns. Optionally **asc** (default) or **desc** can be added after the column name. E.g. **attribute1 asc,attribute2 desc**
Split (dataset)	• dataset • split column name (string) • split expression (string)	Splits every row of a dataset into multiple rows based on the split expression. The split expression must return an array containing splitted items for a single row. Thus, the number of items in the array determines, to how many rows a single row is splitted. The splitted items are stored in a new column. The split expression has all dataset column values as arguments. Note that the splitting may also result to a single row or even zero rows, if the split expression returns an array of one or zero items. Example: Split([dataset], 'splitted', 'StringToArray([column_1], \', \')')
Transpose(dataset)	• dataset	Transposes a dataset, i.e. changes its rows to columns and columns to rows. In the transposed dataset, first column name is **headers** and it contains the header names of the original dataset. Rest of the header names are **column_1**, **columns_2**, …
Union (dataset)	• dataset1 • dataset2	Union of two datasets. Results dataset contains all rows of the input datasets. The datasets must have same columns.
Where (dataset)	• dataset • expression (string)	Filters out all rows in the dataset where expression is evaluated as false. Expression has all columns as arguments.

7.9 XML Functions

Function	Parameters	Description
SelectXMLAttribute (string)	• XDocument or XElement • XPath expression (string) • namespace prefix (string) • namespace URI (string) • attribute name (string)	Gets an XML attribute value from an XML element that is selected from an XML document using an XPath expression. An empty string is returned if no element matches or if the attribute is not found.
SelectXMLAttributes (string array)	• XDocument or XElement • XPath expression (string) • namespace prefix (string) • namespace URI (string) • attribute name (string)	Gets a list of XML attribute values from a list of XML elements that are selected from an XML document using an XPath expression. An empty set is returned if no element matches.
SelectXMLElement (XElement)	• XDocument or XElement • XPath expression (string) • namespace prefix (string) • namespace URI (string)	Selects an element from an XML document using XPath expression. The namespace definition is for the XPath expression. Null value is returned if no element matches.
SelectXMLElements (XElement array)	• XDocument or XElement • XPath expression (string) • namespace prefix (string) • namespace URI (string)	Selects an array of elements from an XML document using XPath expression. The namespace definition is for the XPath expression. An empty set is returned if no element matches.
XmlAttribute (string)	• XElement • attribute name (string) • default value (string)	Gets an XML attribute value from an XML element. Empty value (or optional default value) is returned if the attribute is not found.
XmlDocument (XDocument)	• XML data (string) • XML schema (string) • custom error message (string)	Constructs an XML document from a string and validates it. Type of the returned object is **XDocument**. The XML data is provided as the first parameter and an XML schema as a second parameter. If the XML document is not compatible with the schema (i.e. valid), an error is thrown.
XPathEncode (string)	• string	Encodes a string to be suitable for an XPath expression. The function adds quotes ("") if needed.

7.10 Binary Data Functions

File data functions are for getting file contents as a byte array from different sources. There are also functions for getting media types of files.

Function	Parameters	Description
HttpFileData (byte array)	• url (string) • alternateUrl (string)	Fetches a file as byte array through an http(s) using the provided url address. The address must point directly to the file resource. If no resource is found from url, an alternateUrl is tried.
HttpFileMediaType (string)	• url (string) • alternateUrl (string)	Media type of the file fetched using HttpFileData function with a same parameter. If no resource is found from url, an alternateUrl is tried.
LoadFileFromDisk (byte array)	• location (string)	Loads a file from the local disk drive as byte array. To take into account data security, avoid implementation where users are able to select the file to load. Users need to have read access to the file.
QprEmbeddedFileData (byte array)	• object (string) • attribute (string) • options (string)	Gets an embedded file from QPR system as byte array using QPR Web Service's **GetBinaryData** operation with provided parameters. More information http://kb.qpr.com/qpr2016-1/index.html?getbinarydata.htm.
QprGraphData (byte array)	• object (string) • options (string)	Gets an image file as byte array using QPR Web Service's **GetGraph** operation with provided parameters.
QprGraphInfo (string array)	• object (string) • options (string)	Get image width and heigth as an integer array.
QprGraphMediaType (string)	• object (string) • options(string)	Media type of the file fetched using qprEmbeddedFileData function with same parameters.

7.11 Other Functions

Function	Parameters	Description
ApplicationSetting (string)	• setting name (string)	Returns application setting value by its name. Settings depend on the application where the expression language is used. E.g. application settings for applications, that are part of QPR Web Services Extensions, are defined in web.config of QPR Web Services Extensions.

ApplicationVersion (string)	[none]	Returns a version number of the application using the expression language, e.g. DWR or DWV.
Coalesce (object)	• object • object, …	Returns first of the objects (counting from left to right) which value is - for strings, first string that is not null or empty string, or - for other than strings, first value that is not null. Minimum of two parameters are needed. If all parameters are nulls or empties, the first parameter is returned.
DataType (string)	• object	Return the type of provided parameter, e.g. int, string, datetime.
DiagramPath (string array)	• object id (string) • attribute name (string) • parent relation (string) • separator (string) • options (string)	Returns an array of EA/PD diagram paths for an object or an object instance. An array is returned as there can be several diagram paths if the object has several instances or if diagrams have been instantiated. Value of the provided attribute is used to identify an object in the diagram path (usually object "name" is used). Parent relation is the relation attribute name for getting parent objects (usually “parentobjects” is used). Also a "separator" character must be provided (usually "/"). "Options" is for QueryObjects operation for getting parent objects.
DynamicWebViews (byte array)	• template name (string) • parameter names (string array) • parameter values (object array)	Runs a QPR Web Views template.
DynamicWordReport (byte array)	• report name (string) • parameter names (string array) • parameter values (object array)	Runs a QPR Word Reports report.
Eval (object)	• expression (string)	Evaluates an expression.
ExecuteRecursion	• return expression	Executes a recursion based on provided expressions. There are two expressions:

	(string) • recursion expression (string) • recursion initial value (object) • exclude traversed (boolean)	- **return expression**: Determines the value a recursion step returns. The expression contains an argument **recursionresult** which is an array containing the return expressions of all the one level below recursion steps. - **recursion expression**: Determines the next level recursions item. Must return an array. The value of this array item will be given to the next step of the recursion. The expression contains an argument **currentrecursionstep** which is the recursion value of the current step. The **recursion initial value** is the value where the recursion starts. Exclude traversed means that items that have already been encountered during the recursion are excluded from next level recursion items. Example: **ExecuteRecursion('1 + Sum([value])', 'QueryObjects(\'[\' + [value] + \'].childobjects\', \'\', \'\', \'id\', \'\')', 'PG.123.456')**
ExecuteSearch (string array)	• search configuration (XDocument)	Executes a search for QPR objects. The search is based on a search configuration provided as an xml document. The schema of the XML document is described in Appendix 1.
ExpressionLanguageVersion (string)	[none]	Returns expression language version number, i.e. version of QPR Suite Accelerator .Net Tools. QPR Suite Accelerator .Net Tools can be upgraded by replacing the QPRSuiteAcceleratorTools.dll in the QPR Web Services Extensions folder under IIS with a new one.
GenerateNumberArray	• start (int) • increment (int) • count (int)	Generates an integer array starting from **start** integer, using defined **increment** till the **count** is generated.
If	• condition (boolean) • true value (object) • false value (object)	Usual programming conditional statement. The **condition** is evaluated, and if the condition is true, the **true value** is returned; otwerwise it returns the **false value**. If the condition is evaluated to a null value, the false value is returned. The false value is optional, and in that case null is returned if condition is false. If the condition is true the false value (expression) is not evaluated and vice versa. Example: **if([variable1] = 2, 'value is two', 'value is something else')**

IsNull (boolean)	• object	Return true if the value is null, otherwise false.
IsNumeric (boolean)	• string	Returns true if the input string can be converted to a numerical value, otherwise false.
Loop (object)	• array • expression (string)	Loops through an array and calculates an expression for every iteration. The function gives a value of the last iteration's expression as a result. If array length is zero, a null value is returned. Following arguments are available in the expression: - **value**: Item in the array. - **previousresult:** Result of previous iteration's calculated expression. For the first iteration, this value is null. - **index**: Iteration order number starting from 0. Example: **Loop(StringToArray('4,2,3', ','), 'coalesce([previousresult], 0) + [value]')** (gives 4+2+3=9)
NullValue	[none]	Return null value.
RaiseError	• error message (string)	Raises (throws) an error and shows an error message that is passed as a parameter. This can be used if there is a more complex logic for parameter validation.
ReportsMenu	• menu configuration(s) (string or string array) • template name (string)	Return html for report items. Template must have parameter reportitems, which is a dataset containing report definitions. visibility, reporturl, reportname, visiblemessage, disabledmessage, target, width, height, scroll, preventcaching, x, y
SwitchCase (object)	• control expression (object) • condition1 (object) • value1 (object) • condition2 (object) • value2 (object) • default value (object)	Conventional programming "switch" statement. If **control expression** equals **condition1**, **value1** is returned and so on. Control expression may be string, integer, double or date. If no condition matches, the **default value** is returned. Note that the number of parameters must be an equal number (4, 6, 8, …). Example: **SwitchCase([variable1], 1, 'value is one', 2, 'value is two', 'value is something else')**
UsageLog (string)	• Log line (string)	Writes a line of text to a local file in disk. The function can be used e.g. for usage logging.

	• Log file (string) • Log mode (string)	Parameters: - **Log line** is a line of text to append to the log file. - **Log file** is the full path and name for the log file. - **Log mode** is optional can be one of the following: ○ **1**: Errors are skipped. ○ **2**: Error message is returned by function as a string. ○ **3**: An exception is thrown by the function. This means that the expression calculation fails. If no errors occur, the function returns an empty string. When using Windows authentication, all users need to have write access to the log file. The function may be used in an Expression or Variable tag in Word reports. Note the possible security and performance issues when writing files. Example: **UsageLog(CurrentDateTime () + ';' + CurrentUser() + ';' + ReportName(), 'C:/Logs/QPRReportsLog.txt')**
WsSessionId (string)	[none]	Returns QPR Web Service session id, if there is a valid session. If not, returns an empty string.

7.12 QPR Word Reports Functions

These functions are only available when used in QPR Word Reports.

Function	Parameters	Description
ReportTemplateData (byte array)	templatepath	Gets a QPR Word Reports template Word file as byte array based on provided template path.
ReportName (string)	[note]	Returns the name of the current report.

7.13 QPR Web Views Functions

These functions are only available when used in QPR Web Views templates.

Function	Parameters	Description

ExistsTemplate (boolean)	- template name (string)	Returns true if the template exists; otherwise false.
ReadTemplate (string)	- template name (string)	Returns template contents as a string. Template must be referenced using absolute path (see DWV documentation). It's possible to read any type of files by adding the file suffix to the file name (for **tpl** files, no suffix is added) Example: to read file schema.xsd in the root folder, use path **/schema.xsd**.
TemplateName	[none]	The current template, i.e. the name of the template where the expression is run.
TemplateParameters	[none]	Returns all template parameters as a dictionary (string, object).
TemplatePath	[none]	The current template's path, i.e. the folder path of the template where the expression is run. The folder path starts from the templates root folder. No starting or ending slashes are part of the path.

7.14 ExecuteSearch Function Configuration

The search is based in a defined **scope**, which is a set of objects where the search is targeted. A scope can be e.g. all published models, all certain types of elements. The scope also includes element attributes where the search is targeted. A scope consist of multiple **scope parts**, which are QPR Web Services queries.

The objects defined by the scope are filtered using a **criteria** (the search results are the matching objects). Criteria can be any expression containing and, or, not and parenthesis.

XML element	Description	Attributes
executesearch (1)	parent: none	**scopecombining**: and, or
scopepart (1…n)	Defines a scope part, which contains a web service query, transformation and attributes. Possible parent element: executesearch	**query**: QPR Web Services query **options**: QPR Web Services query options **transformation**: transformation relation name
criteria (1…n)	Defines criteria for search. There can be criteria elements inside other criteria elements to form a hierarchical structure representing expression calculation order. Criteria can be defined under executesearch for a criteria for all criteria parts, or the criteria can be	**type:** one of the following: • and (one to many sub criteria) • or (one to many sub criteria) • not (one sub criterion) • text (no sub criteria) **searchtext:** searched string. Used when

	defined under a certain criteria part. Possible parent elements: executesearch, scopepart, criteria	type=text. **attribute:** searched QPR Web Service attribute. Used when type=text. **comparisontype:** one of the following: contain, begin, is. Used when type=text **matchcase:** true or false
attribute (0..n)	Defines searched attributes related to a scope part. Parent element is scopepart	**name:** name of the attribute

Example:

```
<?xml version=”1.0” encoding=”utf-8”?>
<executesearch xmlns=”http://www.qpr.com/QPRSuite/ExecuteSearch” scopecombining=”and”>
   <scopepart query=”[PG.1221241820].Ominaisuus” transformation=”” options=”QueryModelingLanguage=EN”>
      <criteria type=”and”>
         <criteria type=”text” searchtext=”haku12” attribute=”name” comparisontype=”begin” matchcase=”true”/>
         <criteria type=”text” searchtext=”haku3” attribute=”sanastontermi.value” comparisontype=”is” matchca
      </criteria>
   </scopepart>
   <scopepart query=”[PG.1221241820].Käsite” transformation=”käsitteenominaisuus”>
      <criteria type=”and”>
         <criteria type=”text” searchtext=”haku1” attribute=”name” comparisontype=”contain” matchcase=”false”
      </criteria>
   </scopepart>
</executesearch>

Schema:

<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.qpr.com/QPRSuite/ExecuteSe

  <xs:complexType name="criteria">
    <xs:sequence>
      <xs:element name="criteria" type="search:criteria" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="text"/>
          <xs:enumeration value="and"/>
          <xs:enumeration value="or"/>
          <xs:enumeration value="not"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="searchtext" type="xs:string"/>
    <xs:attribute name="attribute" type="xs:string"/>
    <xs:attribute name="comparisontype">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="contain"/>
          <xs:enumeration value="is"/>
          <xs:enumeration value="begin"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="matchcase" type="xs:boolean"/>
  </xs:complexType>

  <xs:element name="executesearch">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="scopepart" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="criteria" type="search:criteria" minOccurs="0" maxOccurs="1"/>
```

```
              <xs:element name="attribute" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:attribute name="name" type="xs:string"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="query" type="xs:string" use="required"/>
            <xs:attribute name="options" type="xs:string"/>
            <xs:attribute name="transformation" type="xs:string"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="criteria" type="search:criteria" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

8 Acknowledgements

Open XML Power Tools. Copyright (c) Microsoft Corporation

This license governs use of the accompanying software. If you use the software, you accept this license. If you do not accept the license, do not use the software.

1. Definitions
The terms "reproduce," "reproduction," "derivative works," and "distribution" have the same meaning here as under U.S. copyright law.
A "contribution" is the original software, or any additions or changes to the software.
A "contributor" is any person that distributes its contribution under this license.
"Licensed patents" are a contributor's patent claims that read directly on its contribution.

2. Grant of Rights
(A) Copyright Grant- Subject to the terms of this license, including the license conditions and limitations in section 3, each contributor grants you a non-exclusive, worldwide, royalty-free copyright license to reproduce its contribution, prepare derivative works of its contribution, and distribute its contribution or any derivative works that you create.
(B) Patent Grant- Subject to the terms of this license, including the license conditions and limitations in section 3, each contributor grants you a non-exclusive, worldwide, royalty-free license under its licensed patents to make, have made, use, sell, offer for sale, import, and/or otherwise dispose of its contribution in the software or derivative works of the contribution in the software.

3. Conditions and Limitations
(A) No Trademark License- This license does not grant you rights to use any contributors' name, logo, or trademarks.
(B) If you bring a patent claim against any contributor over patents that you claim are infringed by the software, your patent license from such contributor to the software ends automatically.
(C) If you distribute any portion of the software, you must retain all copyright, patent, trademark, and attribution notices that are present in the software.
(D) If you distribute any portion of the software in source code form, you may do so only under this license by including a complete copy of this license with your distribution.
If you distribute any portion of the software in compiled or object code form, you may only do so under a license that complies with this license.
(E) The software is licensed "as-is." You bear the risk of using it. The contributors give no express warranties, guarantees or conditions. You may have additional consumer rights under your local laws which this license cannot change. To the extent permitted under your local laws, the contributors exclude the implied warranties of merchantability, fitness for a particular purpose and non-infringement.

Open XML SDK. Copyright (c) Microsoft Open Technologies, Inc. http://msopentech.com

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of

this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "{}" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.
