QlikView

WorkBench

Version 11.20 for Microsoft Windows®

Lund, Sweden, 2017

Authored by QlikTech International AB

Contents

1 What's New in Version 11.20 SR5

As of the release of QlikView 11.20 SR5, QlikView WorkBench includes integration with SharePoint 2013. This means QlikView can be integrated into SharePoint 2013 via `div` tags or via iFrame, displaying QlikView objects and allowing interaction.

QlikView WorkBench now includes an extra proxy (`QProxy`) that can be installed separately.

2 Version Compatibility

The version of QlikView WorkBench must always match the version of QlikView Server.

3 Introduction

QlikView WorkBench is used to build solutions with the QlikView AJAX client. It is a development toolbox for use with Microsoft® Visual Studio® and contains controls and templates for development of web sites and extension objects.

3.1 Tools

The following tools are included in QlikView WorkBench:

QvControl	Simplifies the development by allowing drag-and-drop of QlikView objects into a web form, and by setting common tasks and properties without the need to write any code.
Templates for web site projects	Generates an ASP.NET web site project that includes a proxy page. The proxy page helps avoid cross-site scripting issues.
Template for extension objects	Creates the extension object files and includes a wizard for generation of extension object property pages and features for packaging and deployment of extension objects to QlikView Server.

3.2 Requirements

To successfully install and use QlikView WorkBench, the following requirements must be fulfilled by the target system:

- WorkBench tag present in the License Enabler File (LEF) for QlikView Server
- .NET 3.5 (or later)
- Microsoft Visual Studio 2010, 2012 or 2013

Note! The screen shots in this document are based on Microsoft Visual Studio 2010.

4 Installing QlikView WorkBench

Note! Only install QlikView WorkBench on a machine used for development with Microsoft Visual Studio.

Note! QlikView Server and AccessPoint must be up and running before starting the QlikView WorkBench installation.

Proceed as follows to install QlikView WorkBench:

1. Start the installation program, `QlikViewWorkBench_x64Setup.exe` or `QlikViewWorkBench_x86Setup.exe`.
2. The installation program unpacks the files and computes the space needed for the installation. A welcome screen is then displayed. Click **Next** to continue.
3. Select the language profile to use during the installation. Click **Next** to continue.
4. The software license agreement is displayed. Read it and select **I accept the terms in the license agreement**. Click **Next** to continue.
5. Specify the **User Name** and **Organization** for which the installation is to be personalized. Click **Next** to continue.
6. Select the setup type.

   Complete: installs all program features.

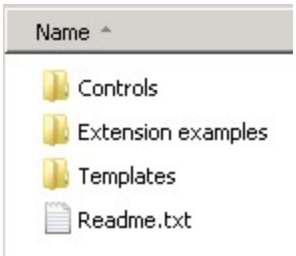
   Custom: lets you chose the program features to be installed.

   Click **Next** to continue.
7. **Note!** This step is only applicable if Custom setup was selected in previous step.

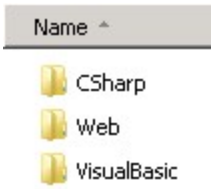
   If you want to omit a feature from being installed, open the context menu for that feature (`WorkBench` or `QProxy`) and **This feature will not be available**.

   Click **Next** to continue.
8. The default installation path is displayed. If another path is to be used, click **Change...** and select or enter a path. Click **Next** to continue.
   Default value: `C:\Program Files\QlikView\WorkBench`
9. Enter the path to the `QvAjaxZfc` virtual directory on the web server, that is, either QlikView Web Server or Microsoft Internet Information Services (IIS).
   Default value: `http://<mycomputer>/QvAjaxZfc/`
10. Click **Test URL** to confirm the path to the directory. If the test was successful, click **Next** to continue.
11. The software is now ready to be installed. Click **Install** to continue.
12. After the installation is done, click **Finish** to complete the installation process.

After the installation, the installation path contains the following files and folders:



The templates that come with QlikView WorkBench are installed in `C:\Program Files\QlikView\WorkBench\Templates`:



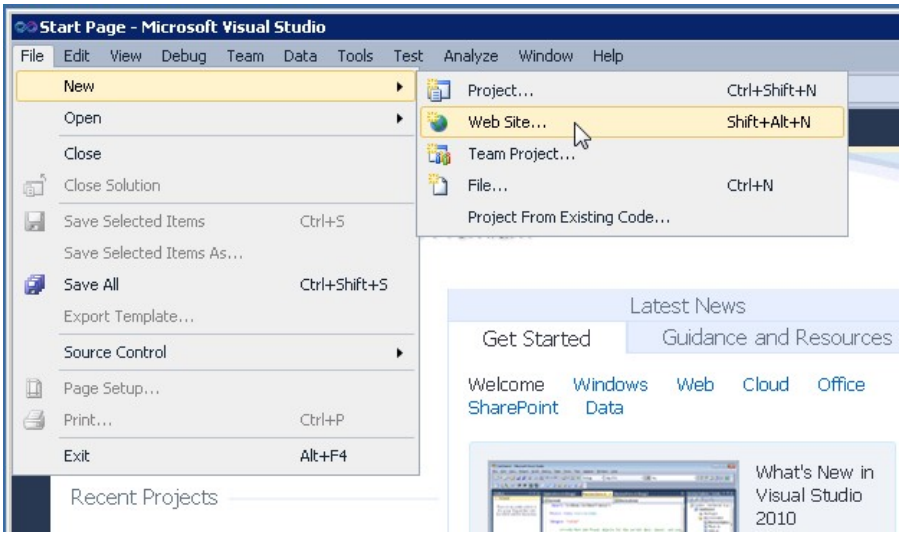
The templates are also installed in the Visual Studio template directory, so that they are available when creating a new web site.

5 Web Site

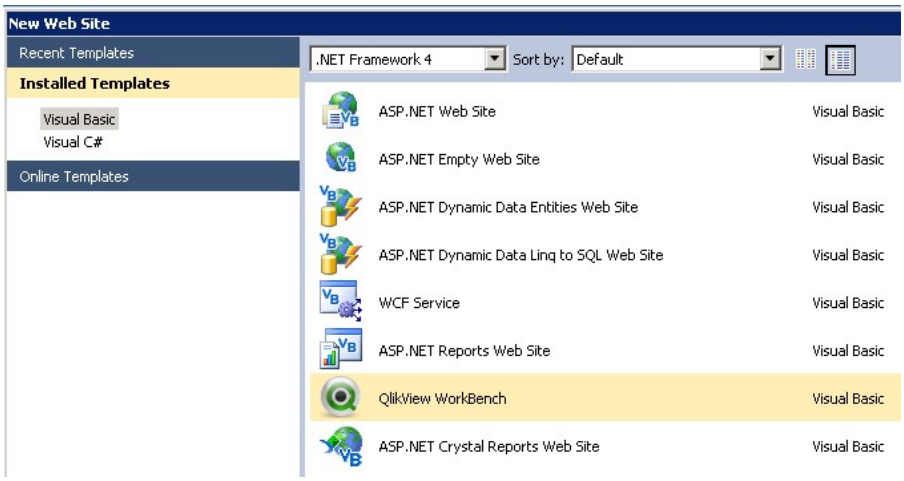
5.1 Creating a Web Site

To create a new, empty web site, proceed as follows:

1. As a Windows® system administrator, open Visual Studio.
2. Select **File>New>Web Site...**



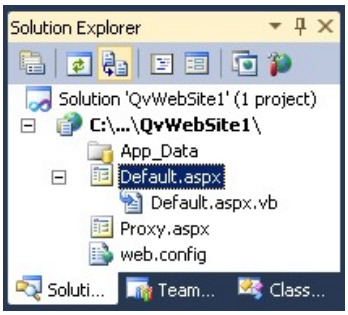
3. The New Web Site dialog is displayed. Select the **Language** (**Visual Basic** or **Visual C#**) and the **Location** (**QlikView WorkBench**) for the web site.



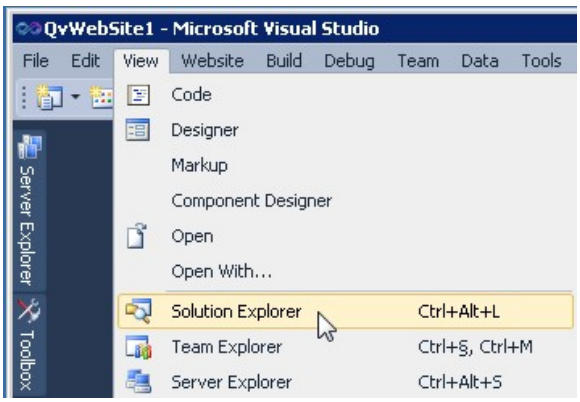
4. There are optional QlikView WorkBench templates included in the installation. These templates contain the `Proxy.aspx` page and settings in the `Web.config` file, allowing the web site to run on any machine, no matter where the `QvAjaxZfc` virtual directory is located. It is recommended to use these templates, even though an empty ASP web site can be used instead. Click **OK** to create the web site.

Note! When using a QlikView WorkBench template, File System can be used as Location in step 3.

The project is now created and can be viewed in the Solution Explorer, which shows the structure of the web site. The `QlikViewWorkBench.dll` and `QlikViewWorkBench.xml` items are *not* displayed until the QvObject has been added on a page in the solution.



If the Solution Explorer is not displayed, select **View>Solution Explorer** to open it.



5.2 Using Proxy

If the AccessPoint (that is, the `QvAjaxZfc` virtual directory) is *not* on the same machine as the QlikView WorkBench web site, a proxy must be used to avoid cross-site scripting issues. For `asp.net` sites, `Proxy.aspx` can be used. If using a QlikView WorkBench template to create the web site, `Proxy.aspx`, is automatically used.

As an alternative, a custom proxy can be created. See *Using Custom Proxy (page 42)* for more information.

5.3 Header Authentication

As an alternative to configuring Kerberos between the server that hosts the web site and the server that runs QlikView Web Server (QVWS), header authentication can be used. A header is always sent, but to specify which header to send, add the tag `Header` and a proper value in `Web.config`:

```
<add key="Header" value="QVUSER"/>
```

Example:

```
<configuration>
  <configSections>
    <sectionGroup name="QlikViewWorkBench">
      <section name="General" type="System.Configuration.
      NameValueSectionHandler" require-Permission="false"/>
    </sectionGroup>
  </configSections>
  <QlikViewWorkBench>
    <General>
      <add key="Proxy" value="/Proxy.aspx"/>
      <add key="LogFile" value="wb.log"/>
      <add key="Header" value="QVUSER"/>
      <add key="QvAjaxZfcPath" value="http://<SERVERNAME>/QvAjaxZfc/"/>
    </General>
  </QlikViewWorkBench>
</configuration>
```

In the example above, a header with the value `QVUSER\<user that is logged in>` is sent to QVWS.

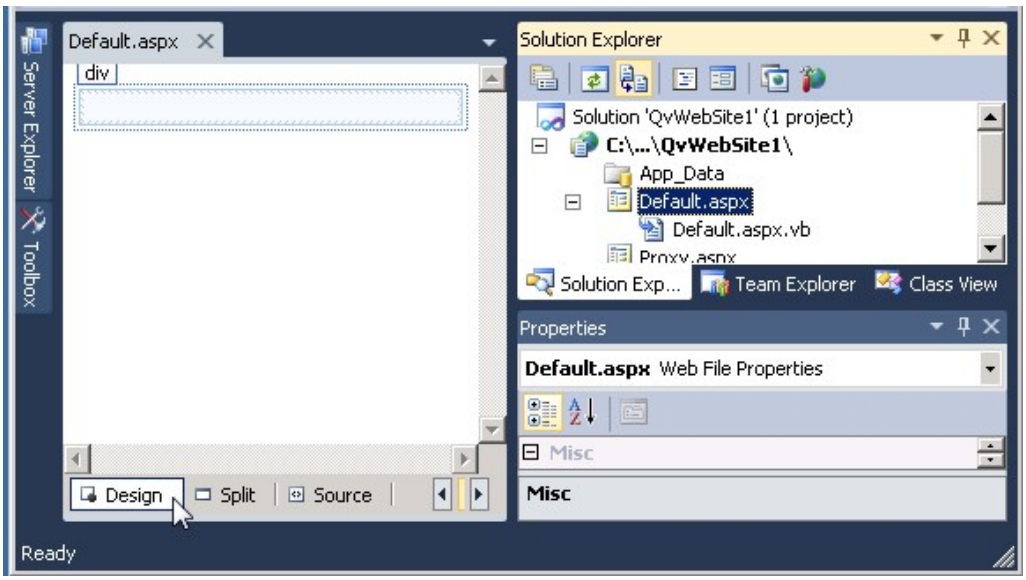
Note! Since headers can be manipulated, it is important to protect the communication between the web site and QVWS.

For information on how to configure QVWS to use header authentication, see the QlikView Server Reference Manual.

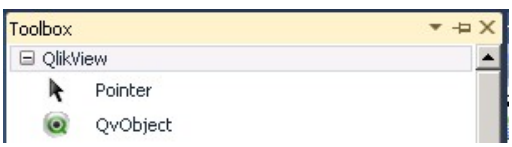
5.4 Building a Web Page

To build a web page, proceed as follows:

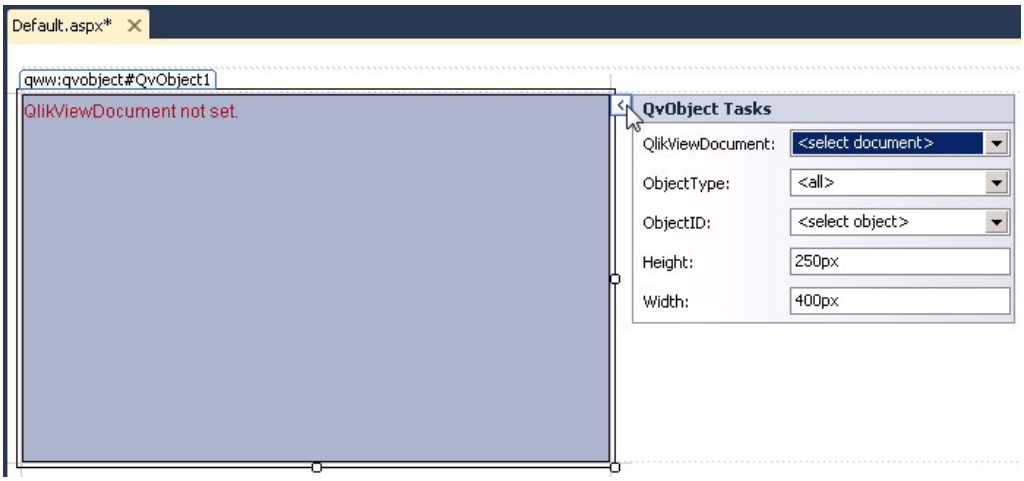
1. Double-click the `Default.aspx` web page to open it and enter Design mode.



2. Select **View>Toolbox** to open the QlikView area.



3. Select **QvObject** in the **Toolbox** and drag it to the web page. The control and a Smart Tag are opened. A Smart Tag is a Visual Studio feature that presents the most common properties of the object that have to be set. To view the Smart Tag at any time, click the arrow tab in the top right corner of the control.



Note! If QvObject is not available in the Toolbox, see *QlikView Control (page 32)* for information on how to add it manually.

4. Press **Ctrl+S** to save the `Default.aspx` web page.

Smart Tag Settings

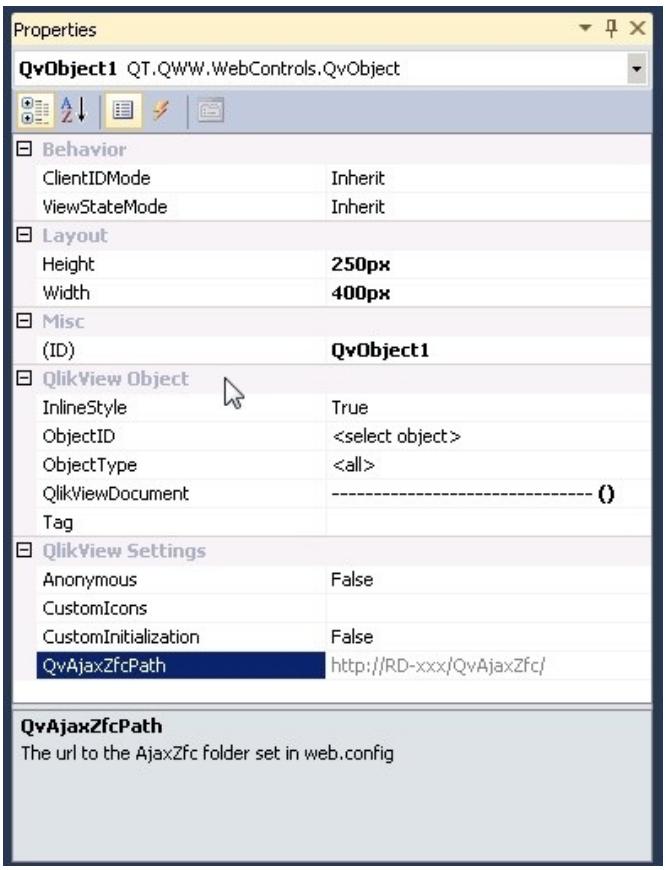
The following object properties are available in the Smart Tag:

QlikViewDocument	Select a document to connect to on QlikView Server.
ObjectType	Select an object type from the QlikView document. This filters the list for ObjectID (see below).
ObjectID	Select an object from the QlikView document.
Height	Set the height of the object on the web page.
Width	Set the width of the object on the web page.

Note! If there are no documents in the QlikViewDocument drop-down list, the QvAjaxZfcPath path setting might be incorrect; see *QlikView Settings (page 18)*.

Property Viewer

The object also has a properties view. Right-click the control to display the Properties pane, and make sure the control is highlighted. Only the properties under the QlikView Object and QlikView Settings headings are relevant to the object.



QlikView Object

The following object properties, in addition to the ones described in *Smart Tag Settings (page 16)*, are available:

InlineStyle See *Inline Styles (page 31)*.

Tag A custom tag can be defined for the object and used in, for example, JavaScript. This can be used for customizing a QvObject, marking a QvObject for special action, distinguishing between one QvObject and another one at runtime, and so on. The tag information can be used to better integrate a QvObject.

Note! The tag is added as an attribute of the `div` tag of the QvObject.

The following example shows a function that returns an array of page elements (representing QvObjects) that have the tag that was passed as input:

```
<script type="text/javascript">
  GetAllQvObjectsByTag = function(tag) {
    var tagObjs = [];
    var divs = document.getElementsByTagName("div");
    for (var i = 0; i < divs.length; i++) {
      if (divs[i].getAttribute("Tag") == tag) {
        tagObjs.push(divs[i]);
      }
    }
    return tagObjs;
  }
</script>
```

QlikView Settings

Note! The properties defined in the QlikView Settings group affect *all* QlikView objects on the web page.

Anonymous Set this property to **True** to allow the compiled web page to automatically assign the users opening the web page to *Anonymous*. It also affects the fetching of documents and objects during design.

CustomIcons Custom images can be used to substitute caption icons. The following caption icons can be replaced by custom icons:

- Lock (icon code: `LOC`)
- Unlock (icon code: `ULC`)
- Clear Other Fields (icon code: `CO`)
- Select Excluded (icon code: `SE`)
- Select Possible (icon code: `SP`)
- Select All (icon code: `SA`)
- Search (icon code: `SEARCH`)
- Send to Microsoft Excel® (icon code: `XL`)
- Clear (icon code: `CD`)
- Print (icon code: `PR`)

The syntax of custom image(s) is `icon code:icon url`. Custom icons must use relative paths. Separate each custom icon with a comma, if more than one is used. For additional information on custom icons, see the JavaScript API documentation.

CustomInitialization Set this property to **True** to enable custom functions (or to make other adaptations, see the JavaScript API documentation) to run.
See *Customizations (page 19)* for an example of how to use this property.

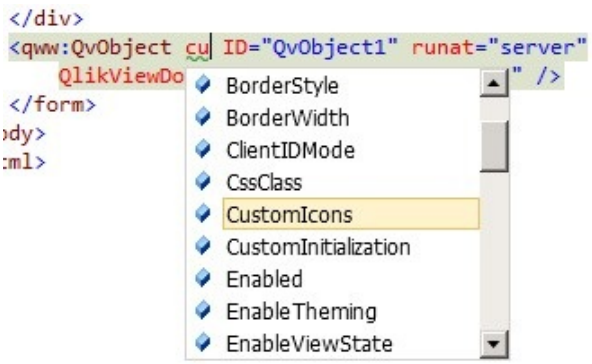
QvAjaxZfcPath	Presents the path to the AJAX directory on the QlikView Web Server (QVWS and IIS, if configured). For example, `/QvAjaxZfc/`, if using a local web server, or `http://QvWebServer/QvAjaxZfx/` (http address), if using a remote server. This property can only be edited in the `Web.config` file.

Source View

A web page can be viewed in Design mode or Source mode. The latter shows the actual web page code:

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb" Inherits="_Default" %>

<%@ Register assembly="QlikViewWorkBench" namespace="QT.QWW.WebControls" tagprefix="qww" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    </div>
    <qww:QvObject ID="QvObject1" runat="server"
        QlikViewDocument="Presidents (Local)" />
    </form>
</body>
</html>
```

The `<qww:QvObject...` tag is the code created by inserting the QvObject on the web page. Properties can be specified here. Thanks to full integration with Visual Studio, there is also the advantage of IntelliSense®, that is, Visual Studio presents the available properties and methods of the control.



Customizations

If custom JavaScript functions are to run after the QlikView object has been rendered, add these functions in the `BodyOnLoadFunctionNames` array. To call the function `MyInit`, add the following line in the JavaScript for the page:

```
Qv.InitWorkBench({ View: 'Films', Host: 'Local', BodyOnLoadFunctionNames:
'MyInit'});
```

To add a custom function that is executed when a specific QlikView object is changed, use the function `GetObject`. In the following example, the number of values that are `Enabled` when a change is made are presented:

```
<script type="text/javascript">
    var mydoc;
    var myobject;
    MyInit = function() {
        mydoc = Qv.GetDocument("Films");
        myobject = mydoc.GetObject("LB1458");
        myobject.SetOnUpdateComplete(MyListIsUpdated);
    }
    MyListIsUpdated = function() {
        alert("Number of enabled values: " +
        myobject.Data.GetEnabled().length);
    }
    Qv.InitWorkBench({ View: 'Films', Host: 'Local', BodyOnLoadFunctionNames: 'MyInit'});
```

If custom functions are to be executed (or other adaptations are to be made, see the JavaScript API documentation), proceed as follows:

1. Set the QvObject property **CustomInitialization** to **True**, see *QlikView Settings (page 18)*.

2. Add an adjusted `Qv.InitWorkBench` code row, containing the needed parameters. A function, `MyInit`, is executed at start-up, in the following example:
   `Qv.InitWorkBench({ View: 'Films', Host: 'Local', BodyOnLoadFunctionNames: 'MyInit'})`

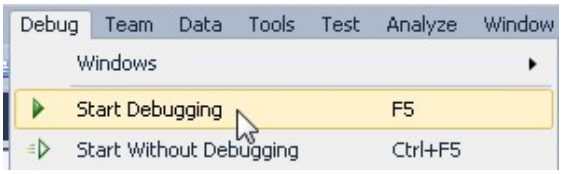
For more information on the arguments that can be passed using functions, see the JavaScript API documentation.

5.5 Running a Web Site in Visual Studio

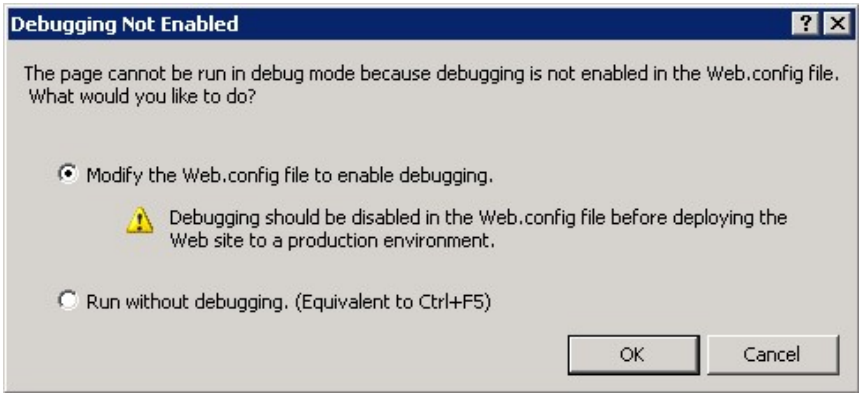
When there is an object on the web page, the web site can be run from Visual Studio. Proceed as follows:

1. Either press **F5** on the keyboard or click the **Run Debug** button in the toolbar:

Alternatively, select **Debug>Start Debugging**:



2. Visual Studio prompts a question, Debugging Not Enabled.
   Select one of the following options:
   Modify the Web.config file to enable debugging – select this option to have Visual Studio edit the `Web.config` file. By selecting this option, the question is *not* prompted again, but can be re-enabled by setting the `compilation debug` parameter to **false** in the `Web.config` file, as follows:
   `<compilation debug="false" strict="false" explicit="true">`
   Run without debugging – select this option to avoid editing the `Web.config` file.
   Click **OK** to verify the selection.



Note! Debugging should be *disabled* in the `Web.config` file before deploying the web site in a *production environment.*

The web site is now displayed in an Internet browser.

6 Extension Object Wizard

The purpose of the Extension Object wizard is to simplify:

- Creation of custom property pages for extension objects
- Creation of Graphical User Interface (GUI) for extension objects
- Deployment of extension objects

However, extensive knowledge about JavaScript and the QlikView JavaScript API is still needed. The wizard generates code and files that previously had to be created manually. The output of the wizard can be used as a template to develop the extension object further.

Note! To make extensions work in QlikView Desktop, the WebView mode must be used.

6.1 QlikView Extensions

In QlikView, there are two kinds of extensions:

- Extension objects
- Document extensions

Extension objects are custom-built QlikView objects, designed to display information from a QlikView document, which can be rendered using web technologies like AJAX, HTML, JavaScript, Java®, Flash®, Silverlight®, and so on.

Document extensions are designed to provide a mechanism to inject JavaScript code that accesses the QlikView JavaScript API in order to extend and modify QlikView documents accessed through an AJAX client.

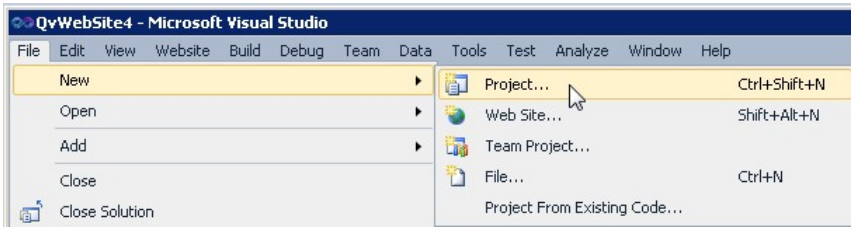
QlikView extensions are designed to work with AJAX clients or with QlikView Desktop in WebView mode. The QlikView JavaScript API library provides access to the information in the QlikView document.

Note! The wizard can be used to create extension objects, but *not* to create document extensions.

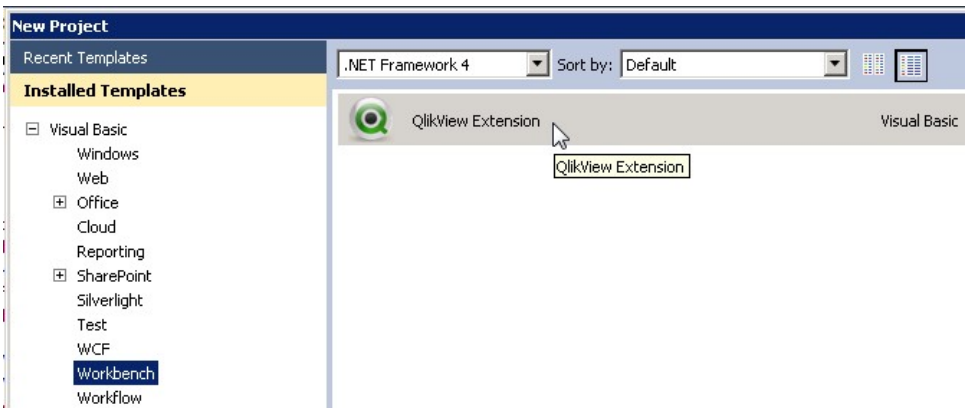
6.2 Creating an Extension Project Using a Template

To create a new project with the extension already placed on the `Default.aspx` page, using the QlikView WorkBench template, proceed as follows:

1. As a Windows system administrator, open Visual Studio.
2. Select **File>New>Project...**

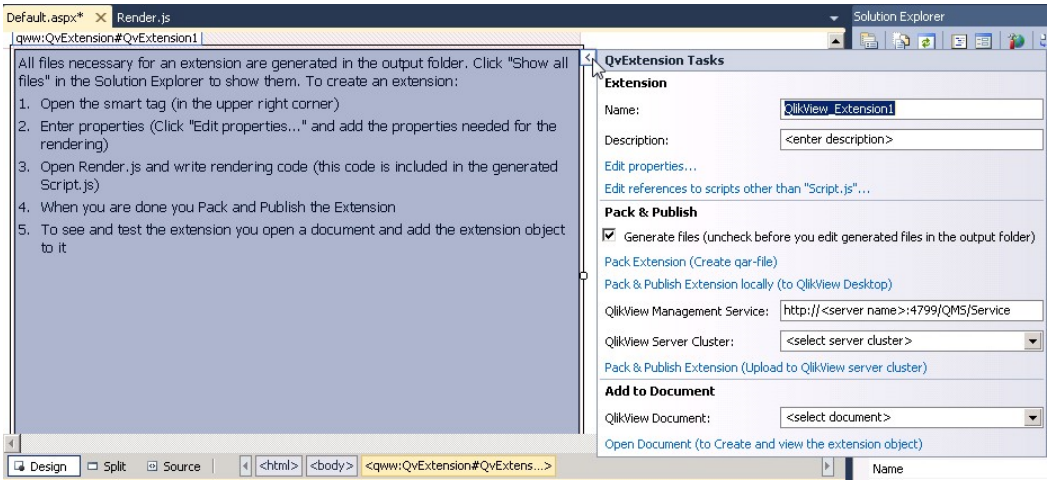


3. The New Project dialog is displayed. In the **Installed Templates** list, select the **Language (Visual Basic** or **Visual C#**) and the **Location (WorkBench**) for the project.
4. Select **QlikView Extension** and click **OK**.



The extension object has now been created. To start using the extension, proceed as follows:

1. Enter Design mode.
2. Follow the steps in the left pane.



Extension

The Extension frame in the QvExtension Tasks pane is used to provide the name of the extension and a description that appears next to the name when a user adds a new extension to a QlikView document.

The frame also contains links to the Edit properties and Edit references to scripts dialogs. The Edit references to scrips dialog is used to edit references to scripts other than `Script.js`, which is the principal script file for the extension. The dialogs provide a simple way to generate properties and script references without the need to write any code.

For information on manual generation of property pages, see **Qvpp Syntax Instructions** in the SDK.

Note! The source path for script files is either a relative path or an http path for external script files.

Pack & Publish

The Pack & Publish frame in the QvExtension Tasks pane does *not* contain actions to modify an extension, but is used for *testing* an extension, either in QlikView Desktop or in QlikView Server.

Generate files

Untick the **Generate files** box to stop changes in the extension from being reflected in the packed extension.

Pack Extension

Click the **Pack Extension** link to create a `.qar` file, which is recognized and used by QlikView.

Pack & Publish Extension locally

Click the **Pack & Publish Extension locally** link to run the `.qar` file (that is, QlikView Desktop is started and the extension is installed).

To run the extension on QlikView Server, make sure that the account running Visual Studio is member of the QlikView Administrator group *and* the QlikView Management API group on the server.

6.3 Deploying an Extension Object from Visual Studio

To simplify the testing of an extension, an extension object can be deployed directly from Visual Studio.

Deploying an Extension Object

An extension object can be deployed in either of the following ways:

- Directly to QlikView Desktop
- Directly to QlikView Server

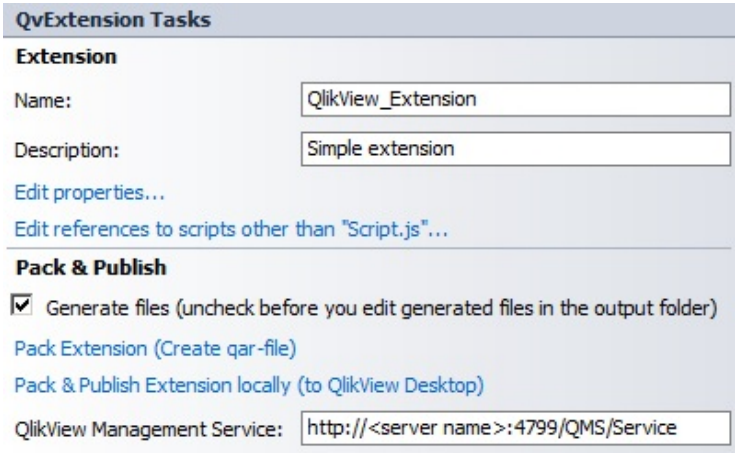
When deploying extension objects to QlikView Server, the objects are moved to the following locations:

- Windows Server® 2003 example: `C:\Documents and Settings\All Users\Application Data\QlikTech\QlikViewServer\Extensions\Objects`
- Windows Server 2008 example: `C:\ProgramData\QlikTech\QlikViewServer\Extensions\Objects`

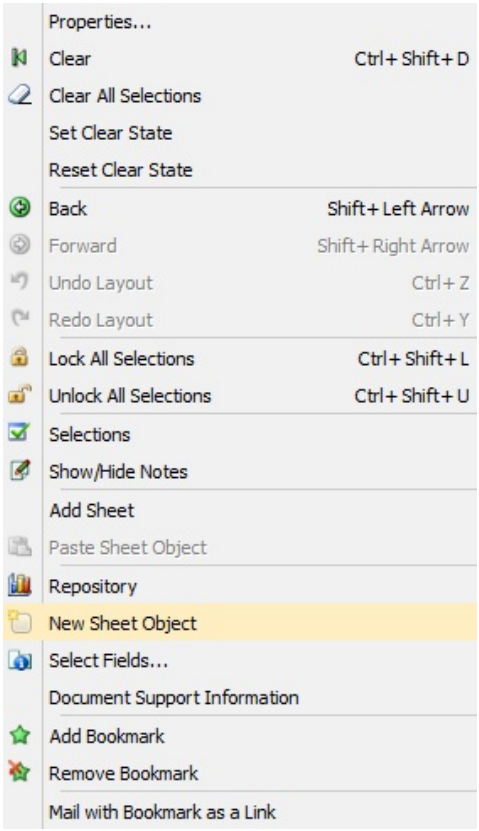
Deploying an Extension Object Directly to QlikView Desktop

To deploy an extension object directly to QlikView Desktop, proceed as follows:

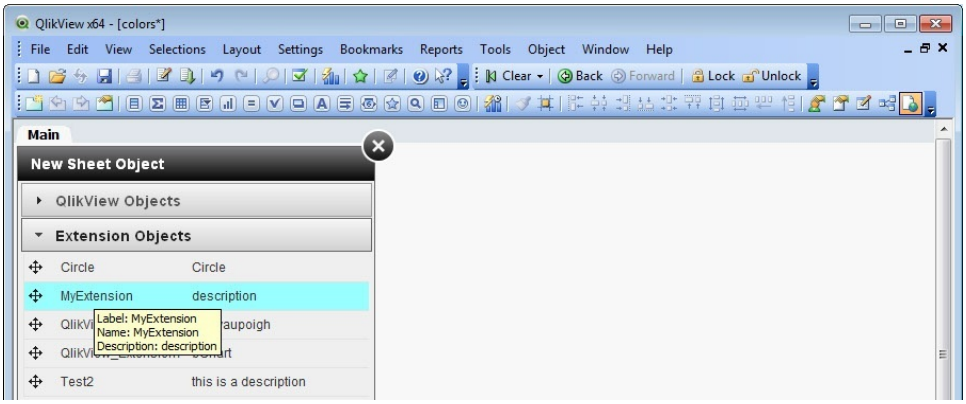
1. Click the **Pack & Publish Extension locally** link in the Pack & Publish frame in the QvExtension Tasks pane when an extension object has been created.
   The extension is packed into a `.qar` file and installed on QlikView Desktop. In addition, the `.qar` file is opened in the started QlikView Desktop.



2. Activate WebView in QlikView Desktop (if not already done).
3. Right-click and select **New Sheet Object** to add the new object.



4. Expand the **Extension Objects** list.
5. Select the extension in the list and drag it to the sheet.



Deploying an Extension Object Directly to QlikView Server

To deploy an extension object directly to QlikView Server, proceed as follows:

1. After an extension object is created, enter the address (of the QlikView Management Service that is used to manage the server to which the extension is to be uploaded) in the **QlikView Management Service** field in the Pack & Publish frame in the QvExtension Tasks pane.

2. Select the QlikView Server cluster to upload to in the **QlikView Server Cluster** drop-down list.

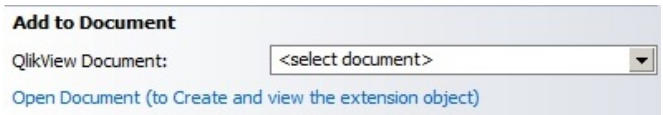
Note! A *single* QlikView Server is also considered a cluster.

3. Click the **Pack & Publish Extension** link to upload to the QlikView Server cluster.

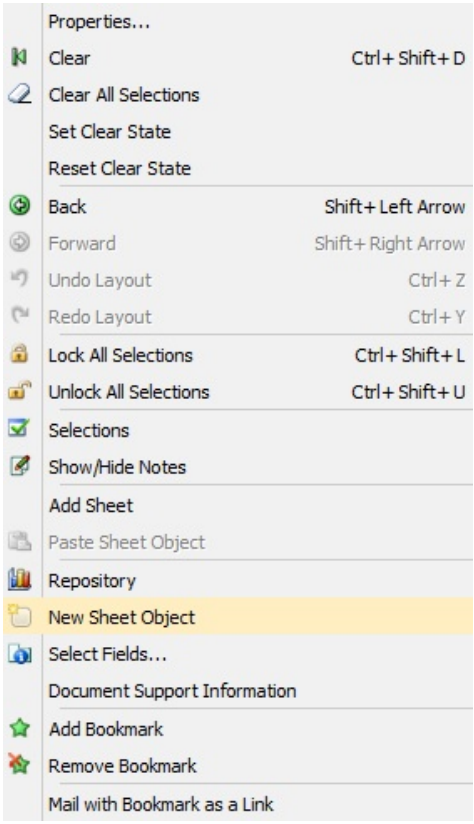
Adding an Extension Object

To add an extension to a QlikView document, proceed as follows:

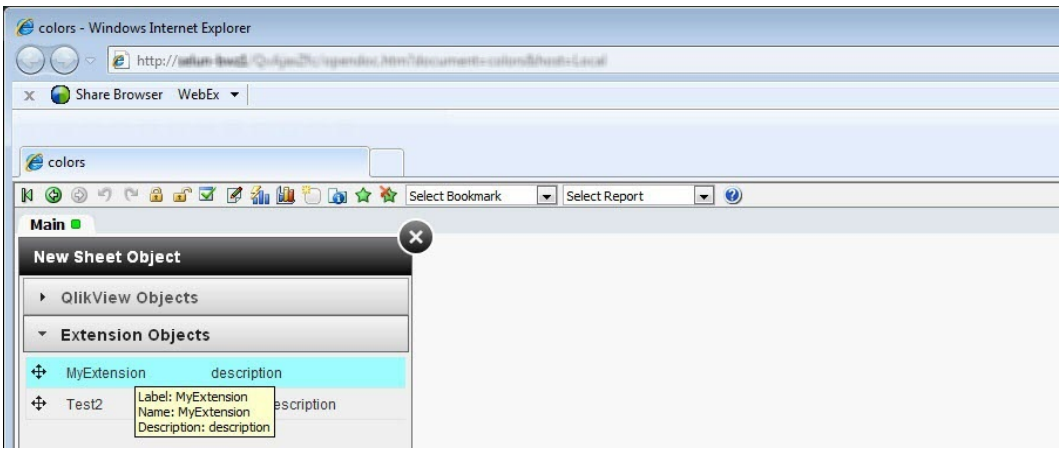
1. In the **Add to Document** frame in the QvExtension Tasks pane, select a document in the **QlikView Document** drop-down list.



2. Click the **Open Document** link to open the document in the AJAX client.
3. In the browser, right-click and select **New Sheet Object**.



4. Expand the **Extension Objects** list.



5. Select the new extension in the list and add it to the document.

7 Troubleshooting

Symptom(s)	Cause(s)	Solution(s)
The developer can open QlikView documents via AccessPoint, yet QvControl cannot access object types or object IDs	There is no WorkBench license in the License Enabler File (LEF) for QlikView Server.	Verify that the WorkBench tag is present in the LEF file for QlikView Server: `WORKBENCH;YES;;`
Some operations in Visual Studio cannot be performed or are not performed correctly	Visual Studio requires the user to have administrator privileges when, for example, accessing files.	Run Visual Studio as Windows system administrator.
Unable to test the connection to the `QvAjaxZfc` directory during the installation of QlikView WorkBench	The `QvAjaxZfc` directory is inaccessible.	Verify that documents can be accessed from QlikView AccessPoint on the machine where QlikView WorkBench is to be installed.
Templates are missing after installation or upgrade of Visual Studio	The templates that come with QlikView WorkBench are installed in `C:\Program Files\QlikView\WorkBench\Templates`. They are also installed in Visual Studio and available when creating a new web site. However, if Visual Studio is installed or updated *after* installing QlikView WorkBench, the templates may not be available in Visual Studio.	Re-install QlikView WorkBench.

8 Appendix

8.1 Logging

By default, all errors are logged in the event log. To extend the logging for debugging purposes, add the `LogFile` key attribute in the `General` section of the `QlikViewWorkBench` tag in the `Web.config` file.

Example:

```
<configuration>
  <configSections>
    <sectionGroup name="QlikViewWorkBench">
      <section name="General" type="System.Configuration.
      NameValueSectionHandler" require-Permission="false"/>
    </sectionGroup>
  </configSections>
  <QlikViewWorkBench>
    <General>
      <add key="Proxy" value="/Proxy.aspx"/>
      <add key="LogFile" value="wb.log"/>
      <add key="Header" value="QVUSER"/>
      <add key="QvAjaxZfcPath" value="http://<SERVERNAME>/QvAjaxZfc/"/>
    </General>
  </QlikViewWorkBench>
</configuration>
```

The operations in Visual Studio can also be logged. To initiate the logging, set the `LogFile` entry in the Windows registry. The default location may differ depending on the operating system:

- Default location in 64-bit (x64) Windows operating systems: `HKEY_LOCALMACHINE\SOFTWARE\Wow6432Node\QlikTech\QlikViewWorkBench`
- Default location in 32-bit (x86) Windows operating systems: `HKEY_LOCAL_MACHINE\SOFTWARE\QlikTech\QlikViewWorkBench`

8.2 Inline Styles

Some styles are set using a stylesheet created by the AJAX engine in QlikView Server. These styles can be overridden by a custom stylesheet. However, other styles are added to the inline HTML generated by the QlikView Server AJAX engine. Normally, these styles cannot be overridden by a custom stylesheet, but changing the default setting, True, of the QlikView object property `InlineStyle` to False allows the inline styles to be overridden by a custom stylesheet. The styles that are provided inline by the QlikView Server AJAX engine are used for fonts, borders, and colors:

- fontfamily
- fontsize
- fontstyle
- fontweight
- textalign
- verticalalign
- textdecoration
- paddingTop
- paddingLeft
- paddingRight
- paddingBottom

- color
- background-color
- text-align
- font-style
- font-weight
- text-decoration
- font-size
- border-bottom
- border-top
- border-left
- border-right
- MozBorderRadiusTopleft
- MozBorderRadiusTopright
- MozBorderRadiusBottomleft
- MozBorderRadiusBottomright
- WebkitBorderTopLeftRadius
- WebkitBorderTopRightRadius
- WebkitBorderBottomLeftRadius
- WebkitBorderBottomRightRadius

8.3 QlikView Control

The QvObject can be added to and removed from the Toolbox.

Adding QlikView Control to the Toolbox

This is usually performed automatically by the QlikView WorkBench installation, but if necessary, proceed as follows:

1. Start Visual Studio.
2. Create a new ASP.NET web application or open an existing project.
3. Right-click in the **Toolbox** (normally in a tab to the left) and select **Add Tab**.
4. Name the tab "QlikView".
5. Right-click in the space below the added QlikView tab and select **Choose Items...**
6. After a while, the Choose Toolbox Items dialog opens. Click the **Browse...** button.
7. Navigate to the file `QlikViewWorkBench.dll` (normally installed in `C:\Program Files\QlikView\WorkBench\Controls\QlikView`) and click **Open**.
8. Click **OK** in the Choose Toolbox Items dialog.
9. The QlikView tab should now contain a QvObject control. If not, make sure that an `.aspx` page is selected in the solution.

Removing QlikView Control from the Toolbox

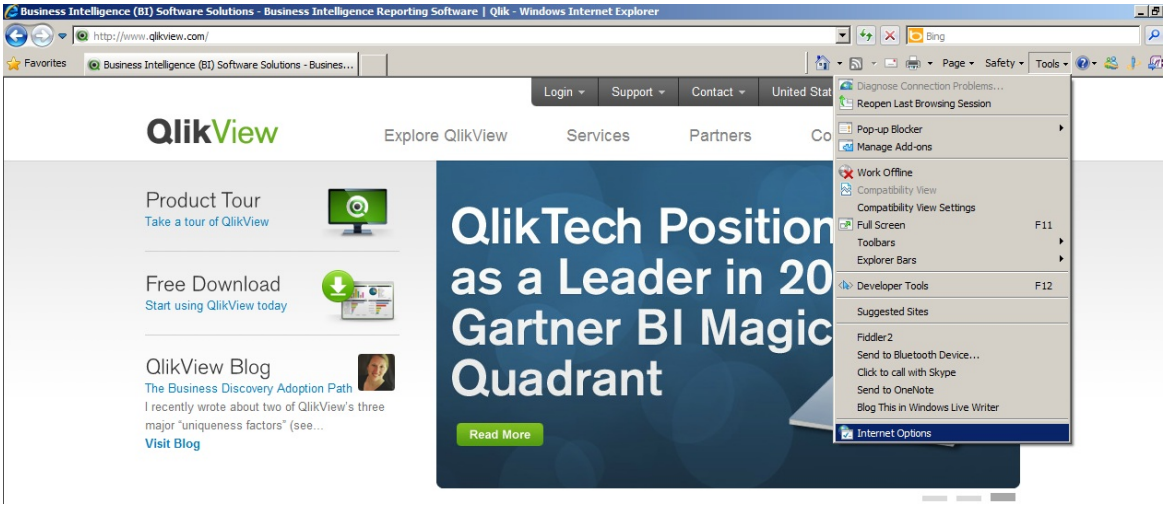
This is usually performed automatically by the QlikView WorkBench un-installer, but if necessary, proceed as follows:

1. Start Visual Studio.
2. Create a new ASP.NET web application or open an existing project.
3. Make sure that an `.aspx` page is selected.
4. Right-click the **QlikView** tab in the **Toolbox** and select **Delete Tab**.
5. Click **OK** in the confirmation dialog to remove the QlikView tab along with its controls.

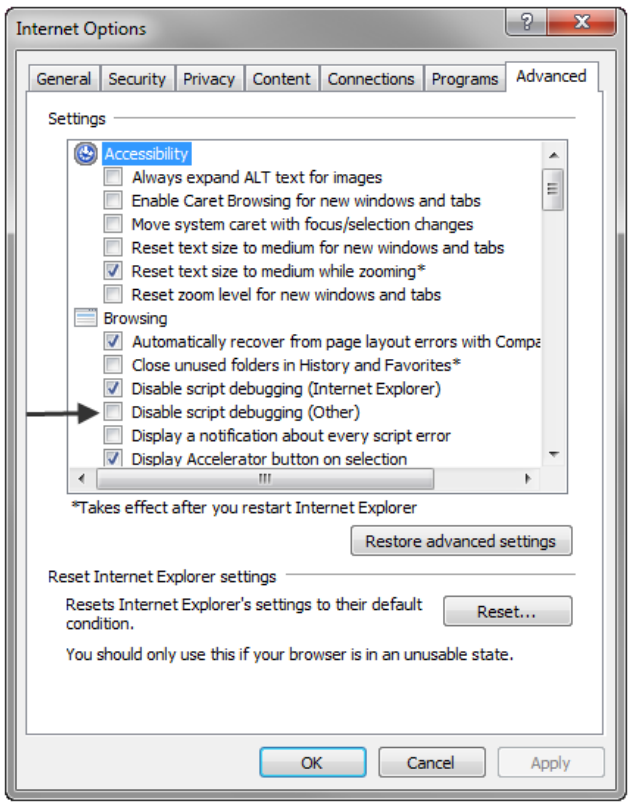
8.4 Debugging Extensions in Visual Studio

To debug an extension in Visual Studio, script debugging has to be enabled in Internet Explorer®. Proceed as follows:

1. Open Internet Explorer.
2. Select **Tools>Internet Options**.



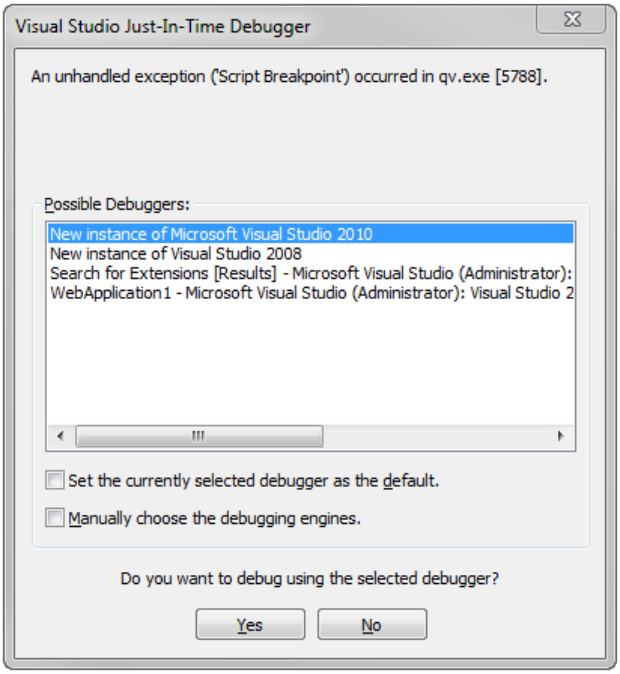
3. In the Internet Options dialog, select the **Advanced** tab.



4. Make sure that the **Disable script debugging (Other)** box is unticked.
5. To make the execution of the extension break, add the statement `debugger;` anywhere in the JavaScript code. For example, the following causes Visual Studio to prompt for how to debug:

```
Qv.AddExtension('QlikView/Examples/HtmlText', function()
{
debugger;
<normal_extension_code>
});
```

6. If Visual Studio is used for developing an extension, select to debug in that instance of Visual Studio, otherwise select a new instance.



8.5 Connecting QlikView WorkBench to QlikView Web Server via JavaScript div Tag

QlikView WorkBench can be used within an `asp.net` web site project, and it can connect to a QlikView Web Server via the `div` tag in a JavaScript without the need for a `.NET .dll` component.

Note! If the `.dll` component is not used, the proxy cannot be used. See *Using Proxy (page 14)* for information on why the proxy is needed.

Below follows an example of the JavaScript code using the `div` tag (as a part of the `.html` code instead of the `.aspx` code):

```
<script type="text/javascript"
src="http://localhost/QvAjaxZfc/htc/QvAjax.js"></script>
<script type="text/javascript">
    var mydoc;
    var MyQvObject;
    MySheetObjects = function(data) {
        alert(data.length);
    }
    MyDocuments = function(data) {
        alert(data.length);
    }
    mySearch = function() {
        MyQvObject.Data.SelectTexts("3","4");
    }
    MyInit = function() {
        mydoc=Qv.GetDocument("Films");
        MyQvObject = mydoc.GetObject("LB1458");
    }
    GetDocumentlist = function() {
        Qv.GetAllDocuments(function(data) {
            alert("data:" + data.length);
        });
    }
    GetObjectList = function() {
        mydoc.GetAllObjects(function(data) {
            alert(data.length);
        });
    }
    Qv.InitWorkBench({ View: 'Films', BodyOnLoadFunctionNames: ["MyInit"],
CustomIcons: {SEARCH:"API_test/Images/NewSEARCH.bmp", CD:"API_test/Images/NewCD.bmp"}
});
</script>
```

Below follows an example of the `div` tag:

```
<div style="width:400px;height:250px;">
    <div class="QvFrame" avqview="Films" avq="object:.Document\LB1458"
    id="Document\LB1458" style="width:400px;height:250px;">
</div>
```

8.6 QlikView Integration with SharePoint 2013

As of the release of QlikView 11.20 SR5, QlikView WorkBench includes integration with SharePoint 2013. This means QlikView can be integrated with SharePoint 2013 via `div` tags or via iFrame, displaying QlikView objects and allowing interaction.

There are two ways of including QlikView content on your SharePoint 2013 site pages:

- As a Script Editor web part which require `QProxy`, or
- As a Page Viewer web part (iFrame), which does not require `QProxy`.

QProxy

As of the release of QlikView 11.20 SR5, QlikView WorkBench includes an updated proxy, `QProxy.aspx`.

It is possible to install only the `QProxy`. In such case, follow the installation procedure in *Installing QlikView WorkBench (page 11)* and select Custom setup and then select not to install `WorkBench` but only the `QProxy`.

Editing web.config

Make sure that section `configSections` of the `web.config` file declares the proxy.

```
<configSections>
  <sectionGroup name="QProxy">
    <section name="General" type="System.Configuration.NameValueSectionHandler"
             requirePermission="false"/>
  </sectionGroup>
</configSections>
```

Then add the `QProxy` section and make sure it specifies the location of the QlikView Server (`QvAjaxZfcPath`).

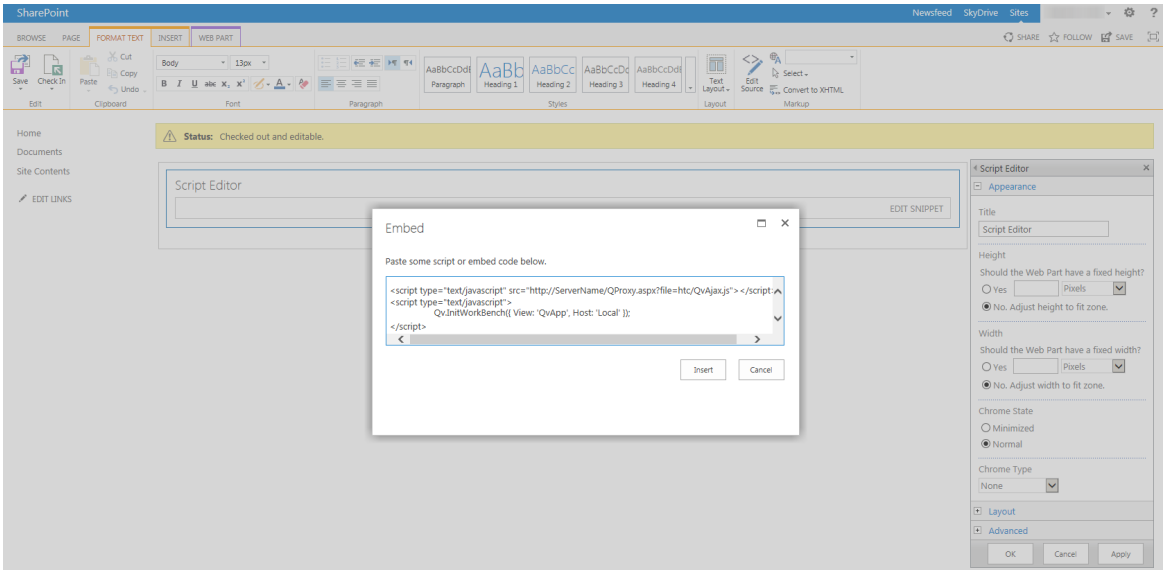
```
<QProxy>
  <General>
    <add key="Proxy" value="/QProxy.aspx"/>
    <add key="LogFile" value="/QProxy_%date%.log"/>
    <add key="LogFilter" value="all, user, url, post, header, trace, response"/>
    <add key="Header" value="QVUSER"/>
    <add key="QvAjaxZfcPath" value="http://QvServerName/QvAjaxZfc/"/>
  </General>
</QProxy>
```

Configurable Proxy

If the user identity in SharePoint is not in the exact same format as required by QlikView Server, the `GetUserName` function can be modified. In a configuration using header authentication, this function can help ensure that the user identity parsed to QVS is in the appropriate format.

Create Site Page - Script Editor Web Part

You can add QlikView content to your SharePoint 2013 web site via `div` tags. You insert a Script Editor web part and then you select to edit snippet. You then insert your `div` tags and also the Ajax Client location.



For more information on adding `div` tags, see *Connecting QlikView WorkBench to QlikView Web Server via JavaScript div Tag (page 34)*.

Examples

Connection Example:

Note! In the example below, `'Test'` is the name of the QlikView document to `view` the objects from, and `host` defines the name of the server cluster.

```
<script type="text/javascript" src="http://SharepointServerName/QProxy.aspx?file=htc/QvAjax.js"></script>
<script type="text/javascript">
        Qv.InitWorkBench({ View: 'Test', Host: 'QVS@[QlikViewServerName]' });
</script>
```

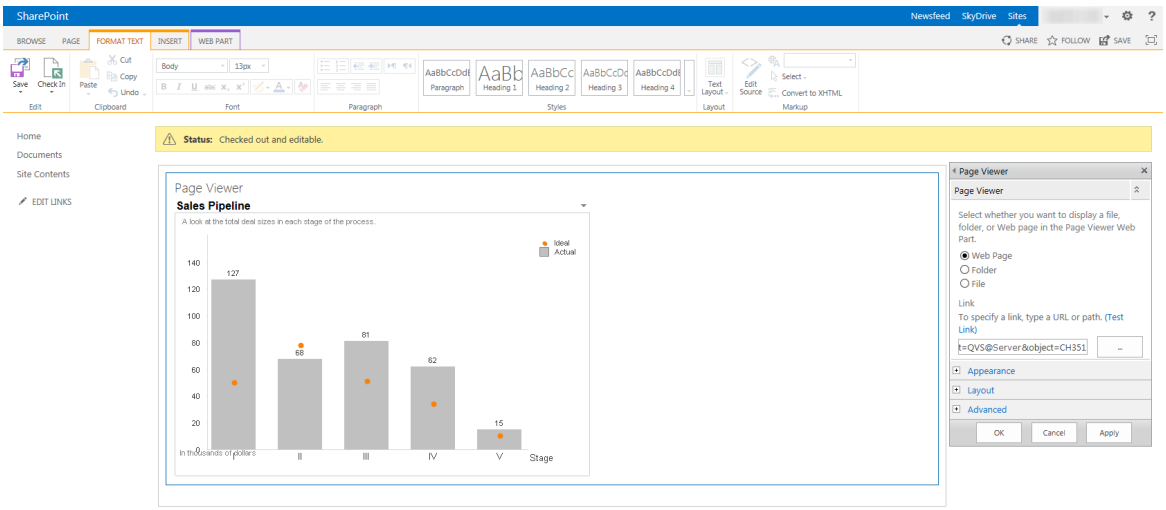
Div Tags Example:

```
<div style="width:100px;height:150px;border-collapse:collapse;">
    <div class="QvFrame" qvqview="Test" qvq="object:.Document\LB01"
         id="Document\LB01" style="display:none;width:100px;height:150px;">
    </div>
 </div>
<div style="width:408px;height:352px;border-collapse:collapse;">
    <div class="QvFrame" qvqview="Test" qvq="object:.Document\CH01"
         id="Document\CH01" style="display:none;width:408px;height:352px;">
    </div>
 </div>
```

Create Site Page - Page Viewer Web Part

Note! This scenario does not require `QProxy` to be installed.

You can add QlikView content to your SharePoint 2013 web site and display it via iFrame. For this you need to specify the URL and Object ID. You insert a Page Viewer web part and when you edit you define the URL and the SingleObject.htm and the ObjectID of the QlikView object you want to display.



SingleObject Example

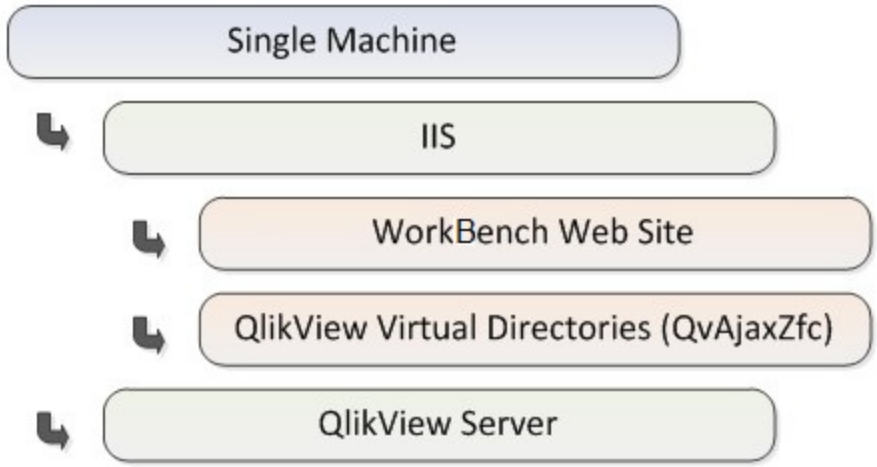
```
http://ServerName/QvAJAXZfc/singleobject.htm?
document=Sales%20Compass.qvw&host=QVS@ServerName&object=CH351
```

8.7 QlikView WorkBench Deployment Scenarios

This chapter describes the most common deployments setups.

1-Tier Scenarios

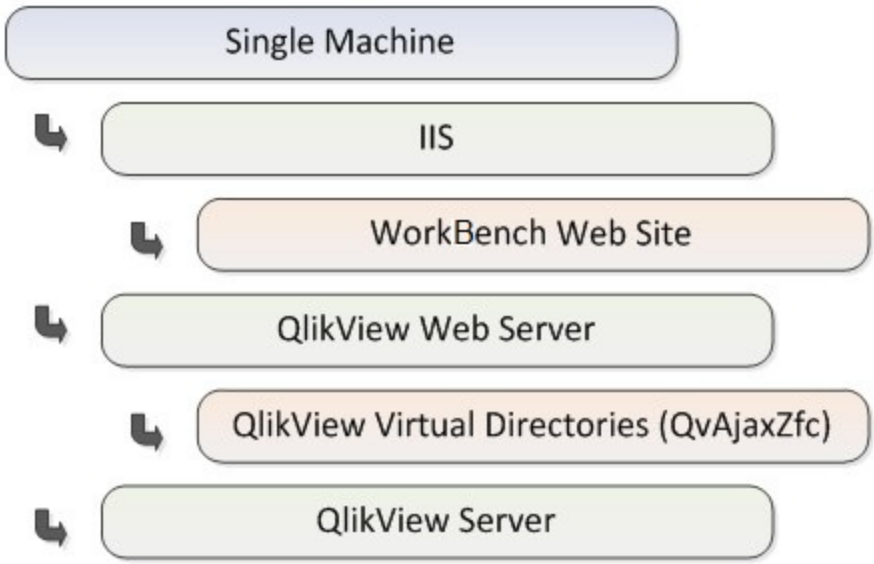
Using Only IIS



This setup is ideal for small or centralized deployments or where only a single machine is available.

It is generally *not* recommended to run a QlikView Server and a web server on the same machine, if the web server is tasked with serving non-QlikView content; in such a case, the load on the web server could negatively impact the performance of QlikView Server. If there is more than a marginal load on the web server, a 2- or 3-tier configuration is recommended.

Using Both IIS and QVWS



Typically, it is easier to use only Microsoft IIS (see *Using Only IIS (page 38)*), which only requires the installation and configuration of a single web server, rather than going with this setup.

However, this setup may prove useful if already having QlikView deployed using QlikView Web Server (QVWS), and later wanting to add a WorkBench web site on the same machine, without disrupting the existing QlikView deployment.

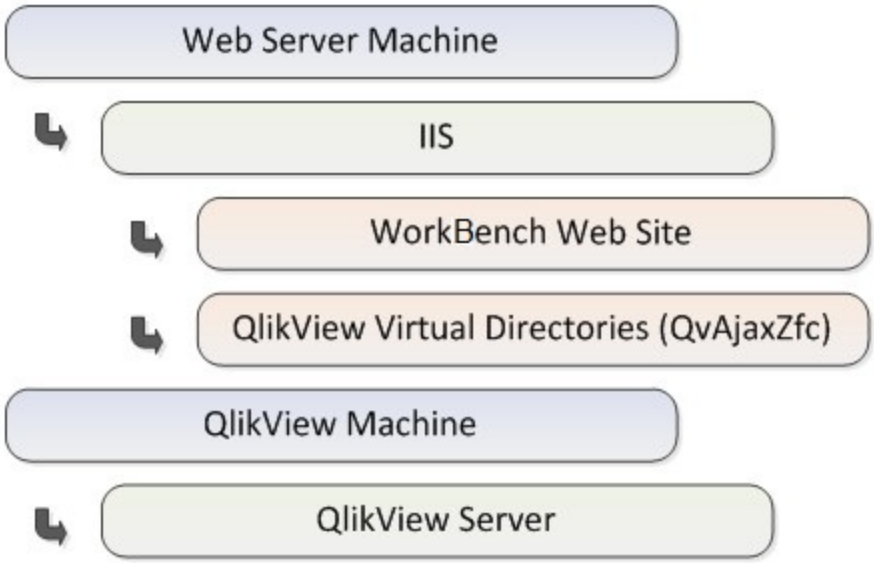
This setup requires port sharing to be turned on in the operating system, or running IIS and QVWS on separate ports.

Running IIS and QVWS on separate ports causes cross-site scripting issues, unless using the proxy page provided by the QlikView WorkBench web site template or another reverse proxy.

It is generally *not* recommended to run a QlikView Server and a web server on the same machine, if the web server is tasked with serving non-QlikView content; in such a case, the load on the web server could negatively impact the performance of QlikView Server. If there is more than a marginal load on the web server, a 2- or 3-tier configuration is recommended.

2-Tier Scenarios

QlikView Server on Own Machine – Using Only IIS



This setup is one of the recommended deployment architectures, as it offers a good combination of performance and ease of setup. Of course, environmental factors, for example, existing QlikView deployments and existing web servers, may make other deployment scenarios more desirable at a given site.

QlikView Server on Own Machine – Using Both IIS and QVWS

Typically, it is easier to use only Microsoft IIS (see *QlikView Server on Own Machine – Using Only IIS (page 39)*), which only requires the installation and configuration of a single web server, rather than going with this setup.

However, this setup may prove useful if already having QlikView deployed using QlikView Web Server (QVWS), and later wanting to add a WorkBench web site on the same machine, without disrupting the existing QlikView deployment.

This setup requires port sharing to be turned on in the operating system, or running IIS and QVWS on separate ports.

Running IIS and QVWS on separate ports causes cross-site scripting issues, unless using the proxy page provided by the QlikView WorkBench web site template or another reverse proxy.

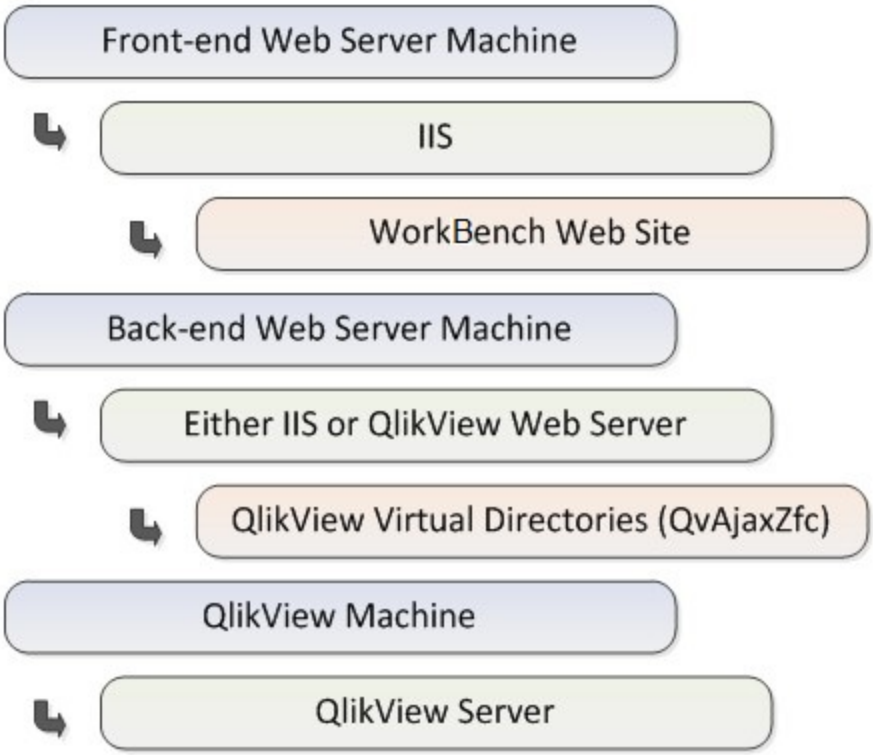
WorkBench Web Site on Own Machine

This setup requires the use of the proxy page provided by the QlikView WorkBench web site template or another reverse proxy, as there are two separate web server domains.

This setup might pose some challenges when using NTLM to execute Windows Integrated Authentication (WIA), as it is subject to the multi-hop issue inherent in NTLM. To execute WIA, Kerberos may have to be configured to allow delegation in this setup. This is not an issue, if NTLM is *not* used as authentication mechanism.

3-Tier Scenario

Every Component on a Separate Machine



This setup requires the use of the proxy page provided by the QlikView WorkBench web site template or another reverse proxy, as there are two separate web server domains.

This setup might pose some challenges when using NTLM to execute Windows Integrated Authentication (WIA), as it is subject to the multi-hop issue inherent in NTLM. To execute WIA, Kerberos may have to be configured to allow delegation in this setup. This is not an issue, if NTLM is *not* used as authentication mechanism.

8.8 Using Custom Proxy

When creating a custom proxy, the following conditions must be fulfilled:

- Cookies are copied from the client request to the server request.
- Headers are copied from the server response to the client response.
- The server response is copied in binary form to the client response.
- The `QvAjaxZfcPath` property is retrieved from the `Web.config` file.
- The requested file is sent using `querystring` using `'file='`.
  Example:
  ```
  url=/QvWebSite2/Proxy.aspx?file=QvsViewClient.aspx&mark=&host=
   Local&view=Films&platform=WORKBENCH&dpi=96
  The QvAjaxZfcPath, http://<SERVERNAME>/QvAjaxZfc/, is retrieved from
  the Web.config file.
  The proxy performs the following request:
  http://<SERVERNAME>/QvAjaxZfc/QvsViewClient.aspx?mark=&host=
   Local&view=Films&platform=WORKBENCH&dpi=96
  ```