



Custom Code Library

**Creating your own libraries of script code to use in
Confirmit surveys**

Copyright © 2009 by Confirmit. All Rights Reserved.

This document is intended only for registered Confirmit clients. No part of the contents of this document may be reproduced or transmitted in any form or by any means without the prior written permission of Confirmit.

Confirmit makes no representations or warranties, express or implied, regarding the contents of this document, and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. The information in this document is subject to change without notice.

Any companies, names and data used or described in the examples herein are fictitious.

The information herein describes the Confirmit Custom Code functionality and its features as of March 2009. New features may be introduced into the Confirmit Custom Code functionality after this date. Go to www.confirmit.com or check "News" on the Confirmit Extranet for the latest updates.

TABLE OF CONTENTS

- 1 Introduction - 4 -**
 - 1.1 ABOUT THIS MANUAL..... - 4 -
- 2 Creating a Custom Code Library DLL - 5 -**
 - 2.1 CREATING THE DLL USING JSCRIPT .NET..... - 5 -
 - 2.2 CREATING THE DLL USING C#..... - 7 -
- 3 Deploying the Custom Library Code DLL - 11 -**
 - 3.1 SERVER-WIDE ACCESSIBLE CUSTOM CODE LIBRARIES..... - 11 -
 - 3.2 CUSTOM CODE LIBRARIES AVAILABLE ONLY FOR SPECIFIC PROJECTS - 11 -
 - 3.3 CUSTOM CODE LIBRARIES AVAILABLE ONLY FOR SPECIFIC COMPANIES - 11 -
- 4 Using Custom Code Library from Confirmit Script Code..... - 13 -**

1 Introduction

The **Custom Code Library** functionality enables you to program your own code libraries and use them in Confirmit surveys.

This is useful for script functionality that is used across surveys. For example, if you have some customized validation code that you use in all your surveys, instead of the survey programmer having to copy the code into each survey, you can include the code in a Custom Code Library. The Confirmit script programmer can then use this code from a script in any survey.

The Custom Code Library assemblies can be scoped to be project-specific, company-specific or available in all surveys across all companies.

The Custom Code Library functionality is also useful if you need to program a bridge between Confirmit surveys and other systems, or for direct database lookup from external databases.

1.1 *About This Document*

The document is intended for Confirmit script programmers, and for administrators of Confirmit server installations.

To program a Custom Code Library, see Chapter 2 Creating a Custom Code Library DLL. You can choose between all the .NET programming languages. This manual describes how to create a Custom Code library for Jscript .NET and C#.

To be able to use the Custom Code Library functionality in surveys, you must deploy the custom code DLL to the Confirmit installation. This is described in Chapter 3 Deploying the Custom Library Code DLL.

2 Creating a Custom Code Library DLL

A Custom Code library DLL is a standard Microsoft .NET assembly that can be created in any of the .NET programming languages. This manual presents examples with Jscript .NET and C# syntax.

2.1 Creating the DLL using Jscript .NET

To create a Custom Library DLL with Jscript .NET, proceed as follows:

1. Declare a class inside a package. This class must extend the class CustomCode which is defined in Firmglobal.Confirmit.SurveyEngine.Common assembly.
2. The functions you wish to expose through your Custom Code Library must be declared static.
3. The standard Confirmit script functions can be accessed with the Survey prefix and using the dot notation. For example, the f(..) function can be accessed from the Custom Code Library with Survey.f(..). See Figure 1 for an example of a Custom Code Library file. Note that the package name, class name and file name can be chosen arbitrarily.
4. You must compile your assembly with an AssemblyInfo.js file. This file should have a fixed version number (for example 1.0.0.0). Do not use the star (*) notation as this will cause the assembly to be incompatible when upgrading Confirmit. A complete AssemblyInfo.js file is provided in Figure 4. The version information is located in the AssemblyVersion tag in this file.
5. You must have your assembly signed with a key distributed by the Confirmit installation. This key is located at [confirmprogram]\bin\GeneratorCode.snk. Note the AssemblyKeyFile tag in Figure 4.
6. Use the line below to compile the code files (the [CONFIRMITPROG] needs to be substituted with the correct path the Confirmit program folder):

```
jsc /t:library /r:[CONFIRMITPROG]web\wix\bin\Firmglobal.Confirmit.SurveyEngine.Common.dll  
CustomCodeLibrary.js AssemblyInfo.js
```

```
// JScript CustomCodeLibrary source code  
//  
import Firmglobal.Confirmit.SurveyEngine.Common;  
  
package MyPackage {  
  
    public class CustomCodeLibrary extends CustomCode  
    {  
  
        // Example function  
        static function GetCurrentPID()  
        {  
  
            return Survey.CurrentPID()  
  
        }  
  
    }  
  
}
```

Figure 1: The Jscript .NET syntax for the custom code class (CustomCodeLibrary.js)

If your custom code library needs access to a web service then we need to Assert that permission before the call to the web service, as shown in Figure 2. Note that Asserting permissions is a required procedure as from Confirmit 12.

```

// JScript CustomCodeLibrary source code
//
import Firmglobal.Confirmit.SurveyEngine.Common;
import System.Net;
import System.Security.Permissions;

package MyPackage {

    public class CustomCodeLibrary extends CustomCode
    {
        // Example function using Confirmit Web Services
        static function MyFunctionThatCallsAWebService()
        {
            new WebPermission(PermissionState.Unrestricted).Assert();
        }
    }
}

```

Figure 2: The Jscript .NET syntax for the custom code class (CustomCodeLibrary.js) accessing Confirmit Web Services

If your custom code needs access to other resources that are protected by a permission, then you will need to Assert them also, for example if you want to write or read a file. If you want your custom code to have unrestricted access to all resources then you can use the permission shown in Figure 3.

```

// JScript CustomCodeLibrary source code
//
import Firmglobal.Confirmit.SurveyEngine.Common;
import System.Net;
import System.Security.Permissions;

package MyPackage {

    public class CustomCodeLibrary extends CustomCode
    {
        // Example function using Confirmit Web Services
        static function MyFunctionThatNeedsFullTrust()
        {
            new PermissionSet(PermissionState.Unrestricted).Assert();
        }
    }
}

```

Figure 3: The Jscript .NET syntax for the custom code class (CustomCodeLibrary.js) with unrestricted access

```
import System.Reflection;
import System.Runtime.CompilerServices;
import System.Security;

//
// General Information about an assembly is controlled through the
// following set of attributes. Change these attribute values to
// modify the information associated with an assembly.
//
[assembly: AssemblyTitle("")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("")]
[assembly: AssemblyCopyright("")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]
[assembly: AllowPartiallyTrustedCallers()] // Needed from Confirmit 11

//
// Version information for an assembly consists of the following
// four values:
//
//     Major Version
//     Minor Version
```

Figure 4: The Jscript .NET syntax of the AssemblyInfo.js file

2.2 Creating the DLL using C#

The easiest way to create a Custom Code Library in C# is to use Visual Studio .NET. From Visual Studio .NET do the following:

1. Go to the **File > New > Project** menu command.
2. Select project type **Visual C# Projects**, and template **Class Library**.
3. Give the new project a logical name, such as **CustomCodeLibrary**.
Visual Studio will now create a project with two source files, **AssemblyInfo.cs** and **Class1.cs**.
4. Rename **Class1.cs** to an appropriate name (in this example it is called **CustomCodeLibrary.cs**).
5. Select the project from Solution explorer, and right click **Add reference....**
6. Browse to the file **Firmglobal.Confirmit.SurveyEngine.Common.dll** located at `[CONFIRMITPROG]\web\wix\bin\`
7. Replace the content of **CustomCodeLibrary.cs** (Class1.cs) and **AssemblyInfo.cs** with the contents of Figure 5 and Figure 8 respectively. Note that the namespace name, class name and file name for the CustomCodeLibrary.cs file can be chosen arbitrarily.
8. Build the solution (press **Ctrl+Shift+B**).

Note the following:

- The methods you wish to expose through your Custom Code Library must be declared static and public.
- The standard Confirmit script functions can be accessed with the Survey prefix and using the dot notation. For example, the f(.) function can be accessed from the Custom Code Library with Survey.f(..).
- You must compile your assembly with an AssemblyInfo.cs file. This file should have a fixed version number (for example 1.0.0.0). Do not use the star (*) notation as this will cause the assembly to be incompatible when upgrading Confirmit. The version information is located in the AssemblyVersion tag in the **AssemblyInfo.cs** file (See Figure 8).
- You must have your assembly signed with a key distributed by the Confirmit installation. This key is located at [confirmprogram]\bin\GeneratorCode.snk. The location of the key file is specified in the AssemblyKeyFile tag in the **AssemblyInfo.cs** file (See Figure 8).

```
using System;
using Firmglobal.Confirmit.SurveyEngine.Common;

namespace MyNamespace
{
    /// <summary>
    /// Summary description for Class.
    /// </summary>
    public class CustomCodeLibrary : CustomCode
    {
        private CustomCodeLibrary()
        {
        }

        /// <summary>
        /// Example method
        /// </summary>
        /// <returns>A string with the current PID</returns>
        public static string GetCurrentPID()
    }
}
```

Figure 5: The C# syntax for the custom code class

If your custom code library needs access to a web service then we need to assert that permission with an attribute on the custom code class as shown in Figure 6.

```
using System;
using Firmglobal.Confirmit.SurveyEngine.Common;
using System.Net;
using System.Security.Permissions;

namespace MyNamespace
{
    /// <summary>
    /// Summary description for Class.
    /// </summary>
    [WebPermission(SecurityAction.Assert, Unrestricted = true)]
    public class CustomCodeLibrary : CustomCode
    {
        private CustomCodeLibrary()
        {
        }
    }
}
```

Figure 6: The C# syntax for the custom code class accessing Web Services

If your custom code needs access to other resources that are protected by a permission, then you will need to Assert them also. The class definition can have a list of permissions, however you are recommended to assert only the permissions you need. Note that Asserting permissions is a required procedure as from Confirmit 12.

If you want your custom code to have unrestricted access to all resources then you can use the permission shown in Figure 7.

```
using System;
using Firmglobal.Confirmit.SurveyEngine.Common;
using System.Net;
using System.Security.Permissions;

namespace MyNamespace
{
    /// <summary>
    /// Summary description for Class1.
    /// </summary>
    [SecurityPermission(SecurityAction.Assert, Unrestricted = true)]
    public class CustomCodeLibrary : CustomCode
    {
        private CustomCodeLibrary()
        {
        }
    }
}
```

Figure 7: The C# syntax for the custom code class accessing Web Services

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Security;

//
// General Information about an assembly is controlled through the
// following set of attributes. Change these attribute values to
// modify the information associated with an assembly.
//
[assembly: AssemblyTitle("")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("")]
[assembly: AssemblyCopyright("")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]
[assembly: AllowPartiallyTrustedCallers()] // Needed from Confirmit 11

//
// Version information for an assembly consists of the following four
// values:
//
//     Major Version
//     Minor Version
//     Build Number
```

Figure 8: The C# syntax of the AssemblyInfo.cs file

3 Deploying the Custom Library Code DLL

The Custom Code Library DLL created in section 2.1 or 2.2 must be deployed to folders on the Confirmit servers. You must create the folders if they do not already exist. Ensure that you copy the Custom Code Library DLL to all applicable servers.

Confirmit supports project or company-specific Custom Code Library DLLs in addition to server-wide accessible Custom Code Library DLLs. Custom code libraries must have unique names.

Note: Adding the DLLs will cause the AppDomain to flush for the authoring and survey engine application. This will make the next request to these applications slower due to aspx compilation. You should also restart the task system on the batch servers.

3.1 Server-wide Accessible Custom Code Libraries

If you want some Custom Code Library functions to be available from all surveys across all companies, the Custom Code Library DLLs must be put into folders as listed in Table1 below.

Folder	Server
[confirmprogram]\web\wix\bin\CustomCode	Author & Deployment
[confirmprogram]\services\Tasks\CustomCode	Batch
[confirmprogram]\web\confirm_authoring\bin\CustomCode	Author

Table 1: The folders for deploying the server-wide Custom Code Library DLLs

3.2 Custom Code Libraries Available only for Specific Projects

If you want some Custom Code Library functions to be available only for specific surveys, the Custom Code Library DLLs must be put in folders as listed in Table2.

The ProjectCustomCode folders should contain a config file for all projects that are to use any of the DLLs in the bin directory. <projectid>.config. For example p01234567.config.

The config file should list all Custom Code Library DLLs used by a project, for example:

```
<?xml version="1.0"?>
<configuration>
<appSettings>
  <add key="assemblies" value="projectspecific_v1.dll;common_v3.dll"/>
</appSettings>
</configuration>
```

Folder	Server
[confirmprogram]\web\wix\bin\ProjectCustomCode\bin	Author & Deployment
[confirmprogram]\services\Tasks\ProjectCustomCode\bin	Batch
[confirmprogram]\web\confirm_authoring\bin\ProjectCustomCode\bin	Author

Table 2: The folders for deploying the project-specific Custom Code Library DLLs

3.3 Custom Code Libraries Available only for Specific Companies

If you want some Custom Code Library functions to be available only for specific companies, then the Custom Code Library DLLs must be put in folders as listed in Table3. (Note: Project-specific and company-specific Custom Code Library DLLs are put in the same folders).

The ProjectCustomCode folders must contain a config file for all companies that are to use any of the DLL's in the bin directory. The config file must be named **Company_<companyid>.config**. The companyid is found on the Details page for the Company under **Authoring > Admin > Accounts**. For example **Company_123.config**.

The config file should list all Custom Code Library DLLs used by a company. For example:

```
<?xml version="1.0"?>
<configuration>
  <appSettings>
    <add key="assemblies" value="companyspecific_v1.dll;common_v3.dll"/>
  </appSettings>
</configuration>
```

Folder	Server
[confirmprogram]\web\wix\bin\ProjectCustomCode\bin	Author & Deployment
[confirmprogram]\services\Tasks\ProjectCustomCode\bin	Batch
[confirmprogram]\web\confirm_authoring\bin\ProjectCustomCode\bin	Author

Table 3: The folders for deploying the company-specific Custom Code Library DLLs

4 Using Custom Code Library from Confirmit Script Code

To use your Custom Code Library from your code, you simply call the methods with `ClassName.MethodName(..)`. For example `CustomCodeLibrary.GetCurrentPID()`. You do not need to specify the namespace where the class is defined. Figure 9 below shows an example of an info node where `CustomCodeLibrary.GetCurrentPID()` is used in a text piping.

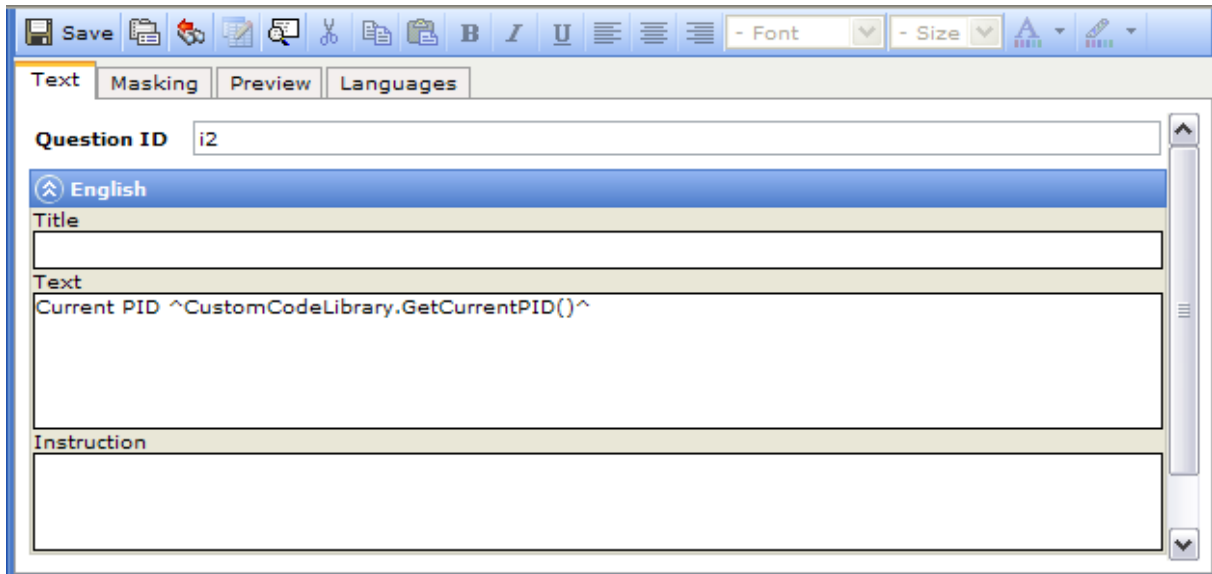


Figure 9: An info node that calls the custom code method `CustomCodeLibrary.GetCurrentPID()`