



# Web Intelligence Extension Points: Building Custom Functions

Web Intelligence BO XI 3.1 SP2

windows and linux



## Copyright

© 2009 SAP® BusinessObjects™. All rights reserved. SAP BusinessObjects and its logos, BusinessObjects, Crystal Reports®, SAP BusinessObjects Rapid Mart™, SAP BusinessObjects Data Insight™, SAP BusinessObjects Desktop Intelligence™, SAP BusinessObjects Rapid Marts®, SAP BusinessObjects Watchlist Security™, SAP BusinessObjects Web Intelligence®, and Xcelsius® are trademarks or registered trademarks of Business Objects, an SAP company and/or affiliated companies in the United States and/or other countries. SAP® is a registered trademark of SAP AG in Germany and/or other countries. All other names mentioned herein may be trademarks of their respective owners.

2009-07-17



# Contents

<b>Chapter 1</b>	<b>Overview of calculation extensions</b>	<b>5</b>
	External functions.....	6
	Deploying the custom functions.....	7
	The library declaration.....	7
	Using the Web Intelligence sample files.....	8
<b>Chapter 2</b>	<b>Defining a custom calculation</b>	<b>11</b>
	XML function objects.....	12
	Defining the XML function declaration.....	15
	Defining the XML catalog declaration.....	17
	Implementing the C++ file.....	17
	Compiling the source file in Microsoft Visual Studio 2005.....	19
	Copying the file into WebiCalcPlugin.....	20
<b>Chapter 3</b>	<b>Examples</b>	<b>21</b>
<b>Chapter 4</b>	<b>Error messages</b>	<b>25</b>
	#EXTERNAL error message.....	26
	Trace log message errors.....	26
<b>Appendix A</b>	<b>More Information</b>	<b>29</b>
<b>Index</b>		<b>33</b>

## Contents



# Overview of calculation extensions



1



chapter

Calculation extensions are custom Web Intelligence reporting calculations that enhance the list of existing Web Intelligence functions.

To use the Calculation Extension Library, create a C++ external library following a specific API.

For detailed information on the advanced calculation capabilities in Web Intelligence and a syntax reference to the Web Intelligence functions and operators, see *Using Functions, Formulas and Calculations in Web Intelligence*, available on the SAP Help portal.

## External functions

External functions are visible and usable like the other Web Intelligence standard functions. You can build a formula with functions that implement your own logic.

**Note:**

You can define as many functions as you need. Only functions that use single value parameters are supported. You can have a maximum of five single value parameters.

To define a function:

1. Declare in an XML file the description of the external function using a given XML structure.
2. Implement the function in a C++ library using a given API.
3. Copy the XML file and library to the appropriate folder in your Business Objects Enterprise installation directory folder for the server and the desktop client.
4. Restart the system to automatically add the external function to the list of the functions available for creating formulas.

The external function is based on a unique identifier so that when it is used in a report, it cannot be misinterpreted in case of using a different external library.

If the system cannot load a library or is missing information for an external function, has an inconsistent XML declaration, missing library, or duplicated function, an error message appears. The system also writes errors in the trace log.

**Related Topics**

- [#EXTERNAL error message](#) on page 26

## Deploying the custom functions

Deployment of custom functions requires a few manual steps. The BusinessObjects administrator must place the XML file and related library DLL file in the library folder for the server, as well as on any desktop rich-client.

**Caution:**

Replace or adding a library in the custom library folder can represent a threat to the system. Since the library is automatically loaded, an external library can access internal critical data or processes, putting the system in danger.

Ensure that the site administrator implements the appropriate security access to the related folder, so that only authorized people access the custom library folder.

## The library declaration

The library file extensions are different depending on the operating system:

- DLL for Windows
- SO for Linux or UNIX

The file types are:

Type	Description
XML catalogs declaration	There is only one file of this type, and it should be named <code>externalcatalogs.xml</code> . This file contains the list of all XML function definition files.

Type	Description
XML functions declaration	This file defines a list of functions and their associated library and is listed in the XML catalogs declaration file. <b>Note:</b> The catalogs file can either contain or reference the function declaration libraries.
library file	This file contains the code in C++ for the user functions. The library file contains the user function implementation as defined in the XML function declaration.

## Using the Web Intelligence sample files

You need to have installed the following applications:

- Visual Studio C++
- Web Intelligence XI 3.1 SP2

The examples in this document use the sample files in the `vs2005_samples.zip` file located in `[Install directory]\userlibs\WebI\Samples\`.

**Note:**

The samples also work with VS2008; however VS2008 migrates the samples when you open VS2008.

1. Uncompress `vs2005_samples.zip`.
2. To open the samples, launch `OpenSolution.bat`.

The `OpenSolution.bat` sets the temporary `WEBICALCPLUGINAPI` variable environment that is used by the solution to find Web Intelligence specific headers files.

**Caution:**

If the required version of Web Intelligence is not installed, you cannot use the `OpenSolution.bat`. Set manually the `WEBICALCPLUGINAPI` variable

environment with the path of the folder that contains the Web Intelligence headers files. Open VS2005\_Samples\WebICalcPlugIn.

### Related Topics

- [Examples](#) on page 21

# 1 | Overview of calculation extensions *Using the Web Intelligence sample files*



Defining a custom  
calculation



2

chapter



To customize a function within Web Intelligence:

1. Define the XML function declaration
2. Define the XML catalog declaration.
3. Implement the library in C++ using the specific API for external function.
4. Compile the source file.
5. Copy the XML definition and the library into the dedicated `WebiCalcPlugin` folder (server side and any rich client).
6. Restart the Web Intelligence server.

**Note:**

The chapter's examples use the sample files delivered with Web Intelligence.

The system automatically adds the function to the function list in the formula editor and formula bar contextual help.

If a formula is using a function for which no external library is available, the `#EXTERNAL` error message appears.

**Note:**

Only functions that use single value parameters are supported.

## XML function objects

The XML definition contains objects define the custom function. XML custom functions extend the function list of the formula language so that a formula using this function can be parsed according its XML signature and turn into a tokenized form. You assign the external function a global unique ID (GUID) so that it cannot be reused or confused with other custom libraries.

The XML definition contains the following objects:

Tag	XML attribute	XML definition object
<i>CATALOG</i>		The XML root

Tag	XML attribute	XML definition object
<i>LIBRARY</i>	file	<p>The name of the library file that contains the C++ implementation code</p> <p>The library file can contain several functions. The library extension should not be specified.</p>
<i>FUNCTION</i>	guid	<p>The unique function GUID</p> <p><b>Tip:</b> Define all GUIDs in advance and make sure that all GUIDs are unique from a global point of view.</p> <p>For Windows you can use the GUID tool provided with Visual Studio or download it from the Microsoft website. For Linux, the tool <code>usr/bin/uuidgen</code> can be found in the <code>libuuid1</code> (Debian) package.</p>
	name	<p>The function name that appears in the formula editor</p> <p>The function name must:</p> <ul style="list-style-type: none"> <li>• be a simple, unique name for the function</li> <li>• start with a letter</li> <li>• use lower and upper case letters, number characters, or the <code>_</code> character</li> <li>• not already exist in the Web Intelligence library</li> </ul> <p><b>Note:</b> The name will not be translated.</p>

Tag	XML attribute	XML definition object
<i>ARGLIST</i>		<p>The list of parameters</p> <p>The number should be less than five.</p>
<i>ARG</i>	type	<p>The parameter types</p> <p>The possible parameter types are as follows:</p> <ul style="list-style-type: none"> <li>• Numeric</li> <li>• Boolean</li> <li>• Date</li> <li>• String</li> </ul>
	name	<p>The name of each parameter as it should appear in the Formula Editor</p> <p>The name shows the prototype of the method to the user. Use only alphanumeric characters.</p>
<i>RETURN</i>	type	<p>The return values type</p> <p>Return values can be:</p> <ul style="list-style-type: none"> <li>• Numeric</li> <li>• Boolean</li> <li>• Date</li> <li>• String</li> </ul>

Tag	XML attribute	XML definition object
<i>CATEGORY</i>	type	The category in which the function will appear in the Formula Editor  Be consistent; place strings in the Character category and Numbers in the Numeric category. The available categories are: <ul style="list-style-type: none"> <li>• Character</li> <li>• Date</li> <li>• Document</li> <li>• DP</li> <li>• Misc</li> <li>• Logical</li> <li>• Num</li> </ul>
<i>HINT</i>	value	A hint to appear in the Formula Editor  The hint explains the use of the function.

## Defining the XML function declaration

The XML for the signature uses the following structure:

```
Function_list
```

The XML for the signature uses the following structure:

```
Function_list := [Function*]
Function := [name, GUID, data_type = Numeric|Boolean|Date|String, category = character|Date|Document|DP|Misc|Logical|Num, parameter_list, (online_help_signature?), (online_help_description?), library_name]
parameter_list := [parameter*]
parameter := [name, data_type =Numeric|Boolean|Date|String]
```

1. Set the XML root tag to `CATALOG`.
2. To the `CATALOG` add `LIBRARY` tags.

3. To the `LIBRARY` add the name of the library file without the DLL or SO file extension. This is the file attribute.
4. To the `LIBRARY` add `FUNCTION` tags.

A `FUNCTION` tag should have a unique GUID and an additional, unique attribute name which defines the name of the function.

The `FUNCTION` tag should contain:

- a `ARGLIST` tag with `ARG` tags. The `ARG` tags should have a first attribute type that defines the type of this parameter, and a second attribute that defines the name of this parameter.

The `ARG` type can be Boolean, Numeric, Date, or String. The `ARG` name contains only alphanumeric characters.

**Note:**

You are limited to five parameters.

- a `RETURN` tag which defines a type attribute.

The `RETURN` type can be Boolean, Numeric, Date, or String.

- a `CATEGORY` tag which defines a type attribute.

The `CATEGORY` type can be Character, Date, Document, DP, Misc, Logical, or Num.

- a `HINT` tag which defines a value attribute.

5. Place the XML definition into the dedicated folder (server side and any rich client).

**Example: SampleMath.xml**

```
<CATALOG>
  <LIBRARY file="SampleMath">
    <FUNCTION guid="CC3E9742-67A7-4844-9DBF-2CCD4F6ECABE"
name="MySquareFct">
      <ARGLIST>
        <ARG type="Numeric" name="input_number"/>
      </ARGLIST>
      <RETURN type="Numeric"/>
      <CATEGORY type="Num"/>
      <HINT value="My square function."/>
    </FUNCTION>
  </LIBRARY>
</CATALOG>
```

### Related Topics

- [Using the Web Intelligence sample files](#) on page 8

## Defining the XML catalog declaration

You can create the XML catalog declaration or add it to an existing catalogs declaration.

<CATALOG> references an XML function declaration file or directly define the <CATALOG> as is shown in the section which defines an XML functions declaration format.

To create a catalog declaration:

1. Name the declaration `externalcatalogs.xml`.
2. Set the XML root tag to `CATALOGS`.
3. To the `CATALOGS` add `CATALOG` tags.

This action defines the file name value of the XML functions declarations.

4. Place the XML library into the dedicated folder (server side and any rich client).

### Example: externalcatalogs.xml

```
<CATALOGS>
  <CATALOG file="SampleMath.xml"/>
</CATALOGS>
```

---

### Related Topics

- [Using the Web Intelligence sample files](#) on page 8

## Implementing the C++ file

1. In the file, add the `ibovariant.h` header.
2. For each method, start the declaration with the `BO_DECLARE_USER_FCT` macro.

The macro includes:

- the function name as it appears in the XML functions declaration file.
- the return value object name

- the parameter object name

**Note:**

The function returns a `BONOERROR` if everything is okay, otherwise the `#EXTERNAL` error message appears into the report.

**Example: Square.cpp**

```
// Headers file include of the WebI headers
#include <ibovariant.h>

// To not repeat BOExtFunct::
using namespace BOExtFunct;

BO_DECLARE_USER_FCT (// Name of function as it was defined in
the XML.
                    MySquareFct,
                    // Name of the return value object.
                    retVal,
                    // Name of the parameters object.
                    parameters
                    )
{
    try // Always used a try{}catch(...) to be sure no
        // exception was thrown outside this Web
        // Intelligence user function.
    {
        // Get the first parameter.
        const iBOValue&param0 = parameters[0];
        // Transform the parameter to the correct type.
        double valPar0(param0);
        // Assign value to the return value.
        retVal = valPar0 * valPar0;
    }
    catch(...)
    {
        return BOERROR; // Unkonwn exception so notify WebI
    }
    return BONOERROR; // It's OK
}
```

---

**Related Topics**

- [Using the Web Intelligence sample files](#) on page 8

## Compiling the source file in Microsoft Visual Studio 2005

1. To create a project, go to **File > New > Project**.
2. In "Project types", select **Visual C++ > General**.
3. In "Templates", select **Empty Project**.
4. Specify the name of the project.
5. Specify the destination folder for the project.
6. Click **OK**.
7. Right-click the project and select **Properties**.
8. In "Configuration", select **All configurations**.
9. In **Configuration Properties > General** set "Configuration Type" to **Dynamic Library (.dll)**.
10. Click **OK**.
11. Right-click the project and select **Add > New Item**.
12. In "Category", select **Code**.
13. In "Template", select **C++ File (.CPP)**.
14. Specify the name of the CPP file.
15. Click **Add**.
16. Right-click the project and select **Properties**.
17. In "Configuration", select **All configurations**.
18. In **Configuration Properties > C/C++**, add the folder which contains the Business Objects file headers.
19. Click **Apply**.
20. In "Configuration", select **Debug**.
21. In **Configuration Properties > C/C++ > Code generation**, set it to **Multi-threaded Debug (/MTd)**.
22. Click **Apply**.
23. In "Configuration", select **Release**.
24. In **Configuration Properties > C/C++ > Code generation** set it to **Multi-threaded (/MT)**.
25. Click **OK**.
26. Add the code to the CPP file.
27. Compile.

## Copying the file into *WebiCalcPlugin*

- Copy the XML functions declaration, the XML catalogs declaration, and the DLL/SO file into the `WebiCalcPlugin` section of the bin folder.

The file is available in a Windows deployment at:

```
[installation directory]\[BusinessObjects  
Version]\[OS]_[PLATFORM]\WebiCalcPlugin
```

where `[BusinessObjects Version]` is the version of the product, for example `BusinessObjects Enterprise 12.0`, and `[OS]` is the operating system, for example `win32` for Windows Operating System or `linux` for Linux Operating System, and `[PLATFORM]` is the platform, for example `x86` on an Intel 32-bit CPU.



Examples



3  
chapter

The examples use the sample files in the `VS2005_Samples.zip` file, which is located in `[Install directory]\userlibs\WebI\Samples\`.

### Example: XML catalog declaration for the `externalcatalogs.xml`

```
<CATALOGS>
  <CATALOG file="SampleString.xml"/>
</CATALOGS>
```

### Example: XML function declaration in `SampleString.xml`

```
<CATALOG>
  <LIBRARY file="SampleString">
    <FUNCTION guid="A91BD526-B8EB-4b09-90F2-FFCD350776A8"
name="MyHelloWorld">
      <RETURN type="String"/>
      <CATEGORY type="Num"/>
      <HINT value="My simple hello world function."/>
    </FUNCTION>
  </LIBRARY>
</CATALOG>
```

### Example: C++ file declaration in `HelloWorld.cpp`

```
// Headers file include of the Web Intelligence headers
#include <ibovariant.h>

// To not repeat BOExtFunc::
using namespace BOExtFunc;

BO_DECLARE_USER_FCT(
    // Name of function as it was defined in
the XML.
    MyHelloWorld,
    // Name of the return value object.
    retVal
    // Don't use parameter.
    /*parameters*/
)
{
    try // Always used a try{}catch(...) to be sure no
        // exception was thrown outside this
        // Web Intelligence user function.
    {
        // Create an std::wstring with wide char Hello world.
```

```
std::wstring helloWorldStr = L"Hello world!!!";  
// Initialise the return value.  
retVal = helloWorldStr;  
}  
catch(...)  
{  
    // Unkonwn exception so notify Web Intelligence  
    return BOERROR;  
}  
return BONOERROR;    // It's OK  
}
```

---

### Related Topics

- [Using the Web Intelligence sample files](#) on page 8





Error messages



4

chapter



## #EXTERNAL error message

The #EXTERNAL error message is caused by the following problems:

- A formula refers to an external function that is not in the external library folder.
- A document contains an external method and the system cannot load it. The library file is not found, or there is an inconsistent declaration.
- An external method does not initialize the return value.
- An external method initialized the return type with bad type. For example, a double was set to a string.
- An external method returns an error code.

Ask the BusinessObjects administrator to deploy the correct library that implements this function.

## Trace log message errors

If an error appears during XML parsing/validation, a message appears to the user and errors are entered in the trace logs.

Log type	Error messages
XML logs	File cannot be read or is missing. Bad XML structure due to: <ul style="list-style-type: none"><li>• Parent/Children relation invalid.</li><li>• Missing field (ID function, name function).</li><li>• Invalid field value.</li></ul>
DLL logs	File is missing. DLL cannot be loaded. Function is not found in the DLL.

Log type	Error messages
Function logs	Function name is already in use. Function ID is already used. Function name is missing. Return type is invalid. ID is invalid. Number of parameters is invalid.
Parameters logs	Parameter name is missing. Parameter type is invalid.
Runtime logs	The user function does not initialize the return value. The user function initializes the return value with a bad type. The user function returns the BOERROR error code.

## 4 | Error messages

*Trace log message errors*



More Information



---



appendix

Information Resource	Location
SAP BusinessObjects product information	<a href="http://www.sap.com">http://www.sap.com</a>
SAP Help Portal	<p>Select <a href="http://help.sap.com">http://help.sap.com</a> &gt; SAP BusinessObjects.</p> <p>You can access the most up-to-date documentation covering all SAP BusinessObjects products and their deployment at the SAP Help Portal. You can download PDF versions or installable HTML libraries.</p> <p>Certain guides are stored on the SAP Service Marketplace and are not available from the SAP Help Portal. These guides are listed on the Help Portal accompanied by a link to the SAP Service Marketplace. Customers with a maintenance agreement have an authorized user ID to access this site. To obtain an ID, contact your customer support representative.</p>
SAP Service Marketplace	<p><a href="http://service.sap.com/bosap-support">http://service.sap.com/bosap-support</a> &gt; Documentation</p> <ul style="list-style-type: none"> <li>• Installation guides: <a href="https://service.sap.com/bosap-inst-guides">https://service.sap.com/bosap-inst-guides</a></li> <li>• Release notes: <a href="http://service.sap.com/releasenotes">http://service.sap.com/releasenotes</a></li> </ul> <p>The SAP Service Marketplace stores certain installation guides, upgrade and migration guides, deployment guides, release notes and Supported Platforms documents. Customers with a maintenance agreement have an authorized user ID to access this site. Contact your customer support representative to obtain an ID. If you are redirected to the SAP Service Marketplace from the SAP Help Portal, use the menu in the navigation pane on the left to locate the category containing the documentation you want to access.</p>
Developer resources	<p><a href="https://boc.sdn.sap.com/">https://boc.sdn.sap.com/</a></p> <p><a href="https://www.sdn.sap.com/irj/sdn/businessobjects-sdklibrary">https://www.sdn.sap.com/irj/sdn/businessobjects-sdklibrary</a></p>

Information Resource	Location
SAP BusinessObjects articles on the SAP Community Network	<p><a href="https://www.sdn.sap.com/irj/boc/businessobjects-articles">https://www.sdn.sap.com/irj/boc/businessobjects-articles</a></p> <p>These articles were formerly known as technical papers.</p>
Notes	<p><a href="https://service.sap.com/notes">https://service.sap.com/notes</a></p> <p>These notes were formerly known as Knowledge Base articles.</p>
Forums on the SAP Community Network	<p><a href="https://www.sdn.sap.com/irj/scn/forums">https://www.sdn.sap.com/irj/scn/forums</a></p>
Training	<p><a href="http://www.sap.com/services/education">http://www.sap.com/services/education</a></p> <p>From traditional classroom learning to targeted e-learning seminars, we can offer a training package to suit your learning needs and preferred learning style.</p>
Online customer support	<p><a href="http://service.sap.com/bosap-support">http://service.sap.com/bosap-support</a></p> <p>The SAP Support Portal contains information about Customer Support programs and services. It also has links to a wide range of technical information and downloads. Customers with a maintenance agreement have an authorized user ID to access this site. To obtain an ID, contact your customer support representative.</p>
Consulting	<p><a href="http://www.sap.com/services/bysubject/businessobjectscounseling">http://www.sap.com/services/bysubject/businessobjectscounseling</a></p> <p>Consultants can accompany you from the initial analysis stage to the delivery of your deployment project. Expertise is available in topics such as relational and multidimensional databases, connectivity, database design tools, and customized embedding technology.</p>



# Index

#EXTERNAL [12](#), [26](#)

## B

BO\_DECLARE\_USER\_FCT [17](#)

## C

C++ file  
    implementing [17](#)

## E

errors  
    in functions [26](#)  
    trace logs [26](#)  
ExternalFunc.xml [6](#)

## F

Formula Editor [12](#)  
functions  
    errors in [26](#)  
    structure [12](#)  
    XML catalog declaration [17](#)

functions (*continued*)  
    XML declaration [15](#)

## G

GUID [6](#), [12](#)

## L

library files  
    structure [7](#)

## S

source files  
    compiling [19](#)

## T

trace logs  
    error messages [26](#)

## W

WebiCalcPlugIn [7](#), [20](#)

## Index