Server Extension User's Guide
SAP BusinessObjects Planning and Consolidation 10.0, version for the Microsoft platform

PUBLIC
Document Version: 1.1 [September 9, 2016]

Copyright

Contents

Icons in Body Text

Icon	Meaning
⚠	Caution
❖	Example
💡	Note
⬆	Recommendation
‹›	Syntax

Additional icons are used in SAP Library documentation to help you identify different types of information at a glance. For more information, see *Help on Help* → *General Information Classes and Information Classes for Business Information Warehouse* on the first page of any version of *SAP Library*.

Typographic Conventions

Type Style	Description
Example text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation.
`Example text`	Emphasized words or phrases in body text, graphic titles, and table titles.
EXAMPLE TEXT	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
`Example text`	Output on the screen. This includes file and directory names and their paths, messages, names of variables and options +arguments, source text, and names of installation, upgrade and database tools.
`Example text`	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, `F2` or `ENTER`.

Document History

Date	Version	Description
December 21, 2012	1.0	Initial version
September 9, 2016	1.1	Changed product name from SAP Business Planning and Consolidation to SAP BusinessObjects Planning and Consolidation.

Overview

SAP BusinessObjects Planning and Consolidation 10.0, version for the Microsoft platform provides server extensions that you can use to extend the functionality of processing dimensions, modifying models, and optimizing models. The three extensions have two methods - PRE and POST - and they aid in the manipulation of imported member table data and other data based on custom logic.

This document provides information on extending standard BusinessObjects Planning and Consolidation functionality that you can include in all Server Extension implementations.

About Server Extensions

Server-side extensions in SAP BusinessObjects Planning and Consolidation are part of a callback-based framework with which you develop your own business logic at particular junctions of a standard workflow. SAP BusinessObjects Planning and Consolidation provides a super class and an SDK, while you provide the plug-in logic. Once your codes are registered, SAP BusinessObjects Planning and Consolidation is responsible for their management and execution.

Server Extension SDK

The SAP BusinessObjects Planning and Consolidation Server Extension SDK is a public DLL that your code needs to reference for the following items:

- ServerExtension base class

  All of your server extensions should be derived from this, and it includes the methods shown in the next topic.
- Data structure and types

  Parameters of each method and return object

Server Extension Methods in SDK

There are both PRE and POST methods for each server extension. The following table shows the methods of the server extensions:

Function	Extension	Description
All	GetOrder	This returns the execution order when multiple server extensions are present. Without this extension, ServerExtensionManager would arbitrarily determine the order when the orders are the same.
Process Dimension	PreDimensionProcess	This method is called after the member data has been imported from the client and before executing the dimension processing steps.
	PostDimensionProcess	This method is called after processing the OLAP dimension and before processing the OLAP cube.
Modify Model	PreModelProcess	This method is called after reassigning the SQL index.
	PostModelProcess	This method is called after processing the OLAP cube.
Optimize Model	PreModelOptimization	This method is called before copying the master data.
	PostModelOptimization	This method is called at the end of the optimization function.

Parameters of the Server Extension Methods

The following table shows the parameters of each server extension method:

Extension Name	Parameter Type	Description
PreDimensionProcess	string env_name	Current environment name.
	string dimension_name	Current dimension name.
	string source_table	Name of the source member table. Name is PreProcess{dimension}.
	DimensionProcessOption process_option	Indicates the current processing option: 0 = IncrementalProcess 1 = FullProcess
	WorksheetOption worksheet_option	Indicates whether processing is performed from the member sheet.
PostDimensionProcess	string env_name	Current environment name.
	string dimension_name	Current dimension name.
	DimensionProcessOption process_option	Indicates the current processing option: 0 = IncrementalProcess 1 = FullProcess
	WorksheetOption worksheet_option	Indicates whether processing is performed from the member sheet.
PreModelProcess	string env_name	Current environment name.
	string model_name	Current model name.
	bool process_cube	Indicates whether the OLAP cube will be processed; options are True or False
PostModelProcess	string env_name	Current environment name.
	string model_name	Current model name.
	bool process_cube	Indicates whether the OLAP cube will be processed; options are True or False
PreModelOptimization	string env_name	Current environment name.
	string model_name	Current model name.
	OptimizeModelOption optimize_option	Indicates the optimization option, such as Full, Incremental, or Lite.
PostModelOptimization	string env_name	Current environment name.
	string model_name	Current model name.
	OptimizeModelOption optimize_option	Indicates the optimization option, such as Full, Incremental, or Lite.

Deployment

- Server extension directory on the file share
  This is the directory in which you deploy the server extensions you have developed.
  [FileServer]\Data\Webfolders\AdminTemplates\ServerExtension\

- Working directory on the application server

  SAP BusinessObjects Planning and Consolidation copies the file from the deployment directory on the file server into this local working directory on the application server.

  [ApplicationServer]\Websrvr\bin\ServerExtension\
- Server Extension SDK

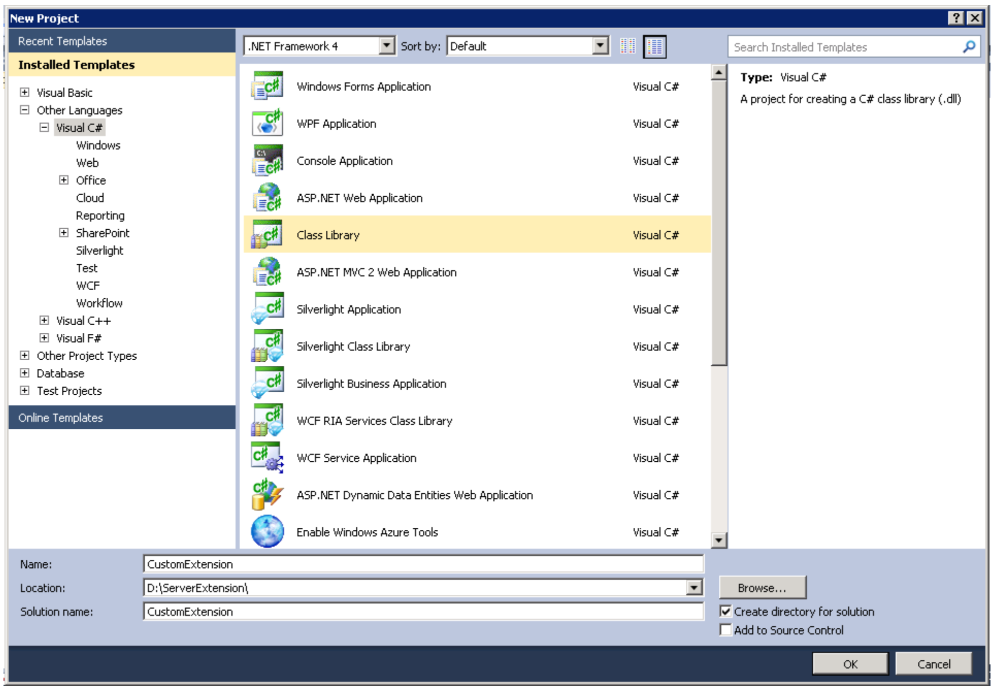
  BPCServerExtensionSDK.dll will be distributed under the Server Extension directory. You must add a reference to this DLL to develop you own server extension. The DLL includes the ServerExtension base class and the data structures that are used as method parameters and return object.

Step-by-Step Procedure

Creating a project and referencing the server extension SDK

You can develop projects to accomplish a custom task. You must develop the custom server extensions.NET C# or VB.NET, version 4.0.

1. Run Microsoft Visual Studio 10.0 and create a Class Library type project.



New Project

Recent Templates

Installed Templates

- ⊞ Visual Basic
- ⊟ Other Languages
  - ⊟ Visual C#
    - Windows
    - Web
    - ⊞ Office
    - Cloud
    - Reporting
    - ⊞ SharePoint
    - Silverlight
    - Test
    - WCF
    - Workflow
  - ⊞ Visual C++
  - ⊞ Visual F#
- ⊞ Other Project Types
- ⊞ Database
- ⊞ Test Projects

Online Templates

.NET Framework 4 Sort by: Default

Search Installed Templates

Windows Forms Application	Visual C#
WPF Application	Visual C#
Console Application	Visual C#
ASP.NET Web Application	Visual C#
Class Library	Visual C#
ASP.NET MVC 2 Web Application	Visual C#
Silverlight Application	Visual C#
Silverlight Class Library	Visual C#
Silverlight Business Application	Visual C#
WCF RIA Services Class Library	Visual C#
WCF Service Application	Visual C#
ASP.NET Dynamic Data Entities Web Application	Visual C#
Enable Windows Azure Tools	Visual C#

Type: Visual C#

A project for creating a C# class library (.dll)

Name:	CustomExtension
Location:	D:\ServerExtension\
Solution name:	CustomExtension

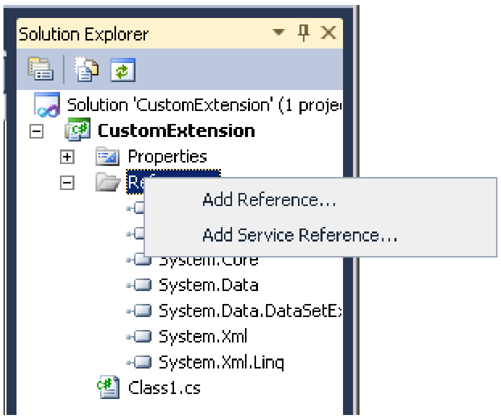
Browse...

☑ Create directory for solution

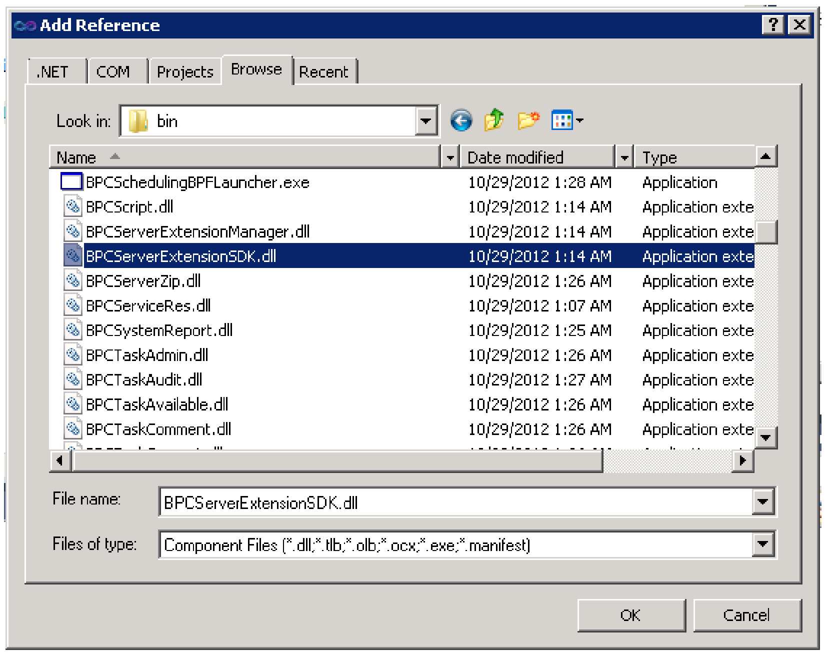
☐ Add to Source Control

OK Cancel

2. Right-click on the *References* tree node and choose *Add Reference*.



3. Add the BPCServerExtensionSDK.dll.
   A. On the *Browse* tab, browse to the [ApplicationServer]\Websrvr\bin\ folder in which the BPCServerExtensionSDK.dll exists.
   B. Select *BPCServerExtensionSDK.dll* and choose *OK*.



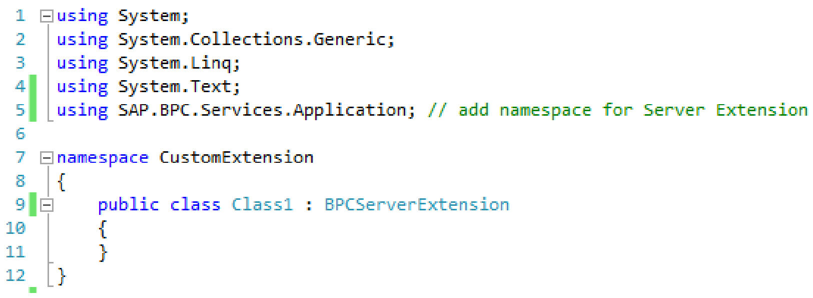
Look in: bin

Name	Date modified	Type
BPCSchedulingBPFLauncher.exe	10/29/2012 1:28 AM	Application
BPCScript.dll	10/29/2012 1:14 AM	Application exte
BPCServerExtensionManager.dll	10/29/2012 1:14 AM	Application exte
BPCServerExtensionSDK.dll	10/29/2012 1:14 AM	Application exte
BPCServerZip.dll	10/29/2012 1:26 AM	Application exte
BPCServiceRes.dll	10/29/2012 1:07 AM	Application exte
BPCSystemReport.dll	10/29/2012 1:25 AM	Application exte
BPCTaskAdmin.dll	10/29/2012 1:26 AM	Application exte
BPCTaskAudit.dll	10/29/2012 1:27 AM	Application exte
BPCTaskAvailable.dll	10/29/2012 1:26 AM	Application exte
BPCTaskComment.dll	10/29/2012 1:26 AM	Application exte

File name: BPCServerExtensionSDK.dll

Files of type: Component Files (*.dll;*.tlb;*.olb;*.ocx;*.exe;*.manifest)

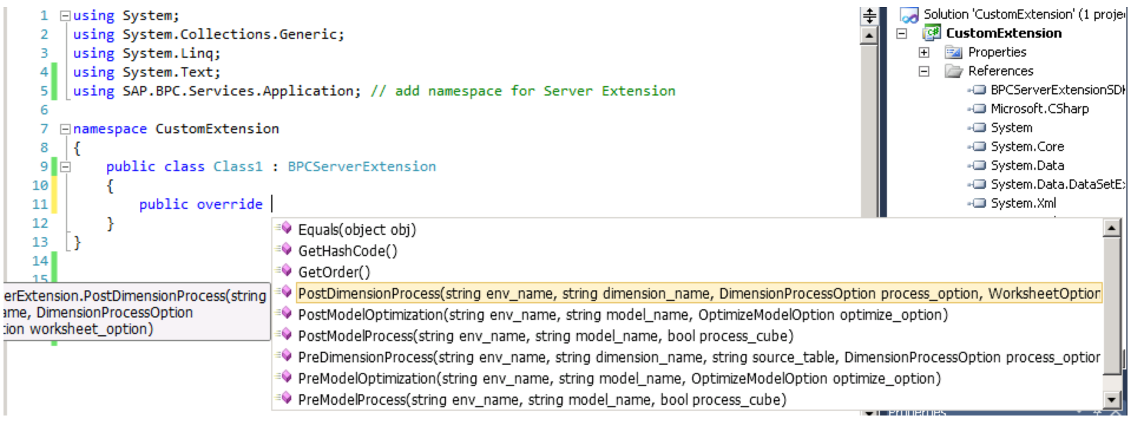
4. Add a namespace and inherit your class from the BPCServerExtension.
   A. Manually add the SAP.BPC.Services.Application namespace manually. This gives you easy access to the BPCServerExtensionSDK and other data structures that are used as method parameters and return object.
   B. Inherit your class from BPCServerExtension.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using SAP.BPC.Services.Application; // add namespace for Server Extension

namespace CustomExtension
{
    public class Class1 : BPCServerExtension
    {
    }
}
```

5. Override methods with the override keyword. In this example, `PostDimensionProcess` is used.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using SAP.BPC.Services.Application; // add namespace for Server Extension

namespace CustomExtension
{
    public class Class1 : BPCServerExtension
    {
        public override |
    }
}
```

```
erExtension.PostDimensionProcess(string
ıme, DimensionProcessOption
:ion worksheet_option)
```

```
Equals(object obj)
GetHashCode()
GetOrder()
PostDimensionProcess(string env_name, string dimension_name, DimensionProcessOption process_option, WorksheetOptior
PostModelOptimization(string env_name, string model_name, OptimizeModelOption optimize_option)
PostModelProcess(string env_name, string model_name, bool process_cube)
PreDimensionProcess(string env_name, string dimension_name, string source_table, DimensionProcessOption process_optior
PreModelOptimization(string env_name, string model_name, OptimizeModelOption optimize_option)
PreModelProcess(string env_name, string model_name, bool process_cube)
```

```
Solution 'CustomExtension' (1 proje
  CustomExtension
    Properties
    References
      BPCServerExtensionSDK
      Microsoft.CSharp
      System
      System.Core
      System.Data
      System.Data.DataSetE:
      System.Xml
```

6. Implement your own custom logic in the derived method.
   A. Create an instance of ServerExtensionResult that is used for the return object.
   B. Implement your own logic in the *try { }* block as shown in line 14 in the example below.
   C. In the *catch { }* block, as shown in line 18, set an exception for the *SetException* property of the return object.

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Text;
 5  using SAP.BPC.Services.Application; // add namespace for Server Extension
 6
 7  namespace CustomExtension
 8  {
 9      public class Class1 : BPCServerExtension
10      {
11          public override ServerExtensionResult PostDimensionProcess(string env_name, string dimension_name, DimensionProc
12          {
13              ServerExtensionResult ret = new ServerExtensionResult();
14              try
15              {
16                  // implement your own custom logic here
17              }
18              catch (Exception ex)
19              {
20                  ret.Success = false;
21                  ret.SetExcetpion = ex;
22              }
23              return ret;
24          }
25      }
26  }
```

Adding status messages

You can show your custom messages through the status dialog in the Admin console, which you can do by adding your message with an Add method like the following:

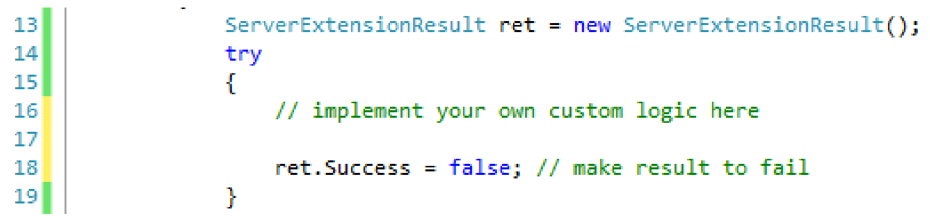
```
ServerExtensionResult ret = new ServerExtensionResult();
try
{
    // implement your own custom logic here

    // Add your custom message,
    // and they will be shown in Status Dialog in admin client
    ret.messages.Add("Your messages here");

    ret.Success = true;
}
catch (Exception ex)
{
    ret.Success = false;
    ret.SetExcetpion = ex;
}
return ret;
```

Making a result fail intentionally

Since you can create your own logic using a server extension, you may want your method to fail according to your custom logic without an exception occurring. To do so, set the Success property of the return object to `false`.



```
ServerExtensionResult ret = new ServerExtensionResult();
try
{
    // implement your own custom logic here

    ret.Success = false; // make result to fail
}
```

Deploying your custom server extension

After creating your custom server extension DLL, copy it to the server extension folder [FileServer]\Data\Webfolders\AdminTemplates\ServerExtension\ on the file share. SAP BusinessObjects Planning and Consolidation does not create the server extension folder by default; you must create it manually on a file share.

To apply your changes after copying your DLL to the file share, IISRESET is required of all application servers.

Making a database connection

SAP BusinessObjects Planning and Consolidation does not provide any connection object for MSSQL or SSAS. If you want to create your own logic against MSSQL or SSAS, you need to make the connection in your custom server extension code.

Please be careful to release any connection objects when making a connection to MSSQL or SSAS. Not doing so may impact performance or reliability of your data.