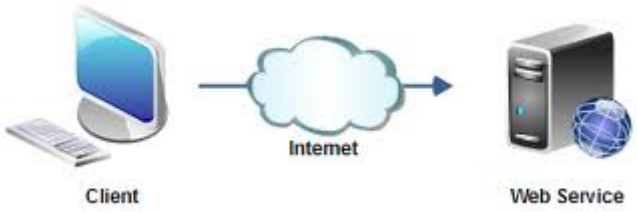
WEB SERVICES

TECHNICAL GUIDE FOR DEVELOPERS



1. Introduction

This Technical Manual considers a development project that need to integrate with the Web Services Central Bank of Chile provides, to obtain data (observations) of the series of daily, monthly, quarterly and annual frequencies that are available through this technology.

You should consider that the source codes examples are written in c# and PHP languages, however to take these examples to java or other languages should be no problem for a developer.

To start using this Web Service our company must request a login account through the following forms:

- Acceptance of terms and conditions
- Certificate of incumbency (Certificate 2)

The url for the Web Services WSDL (Web Service Description Language) is https://si3.bcentral.cl/SieteWS/sietews.asmx?wsdl

Please review the following website for more information

http://si3.bcentral.cl/estadisticas/Principal1/Web_Services/index_EN.htm

2. Web service reference and data collection

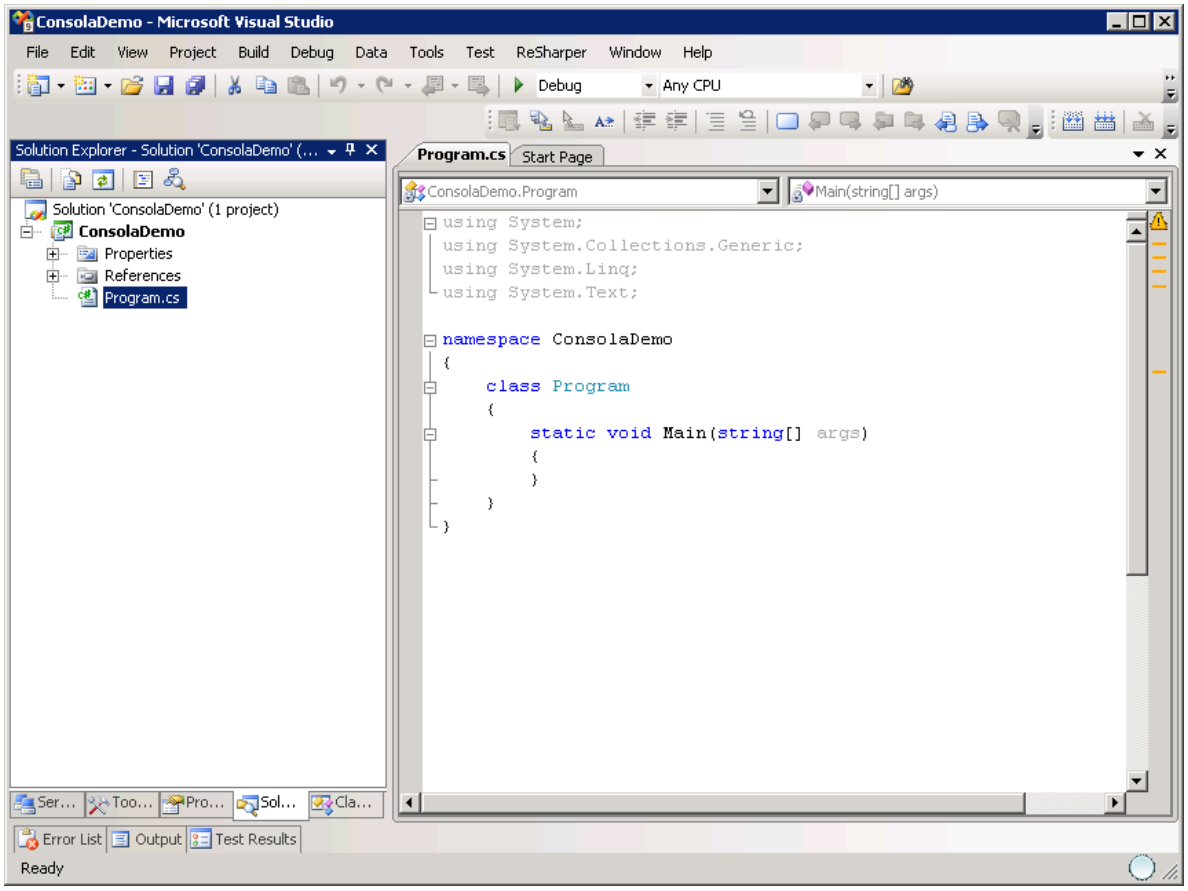
The code and sample application below is built using visual studio 2008 and c# languages.

In the next chapter the code for both c# to PHP is provided. Both perform the same operations, except that c# solution runs as a console application and PHP solution runs as a web page.

For development in PHP steps 3.1 and 3.2 can be omitted.

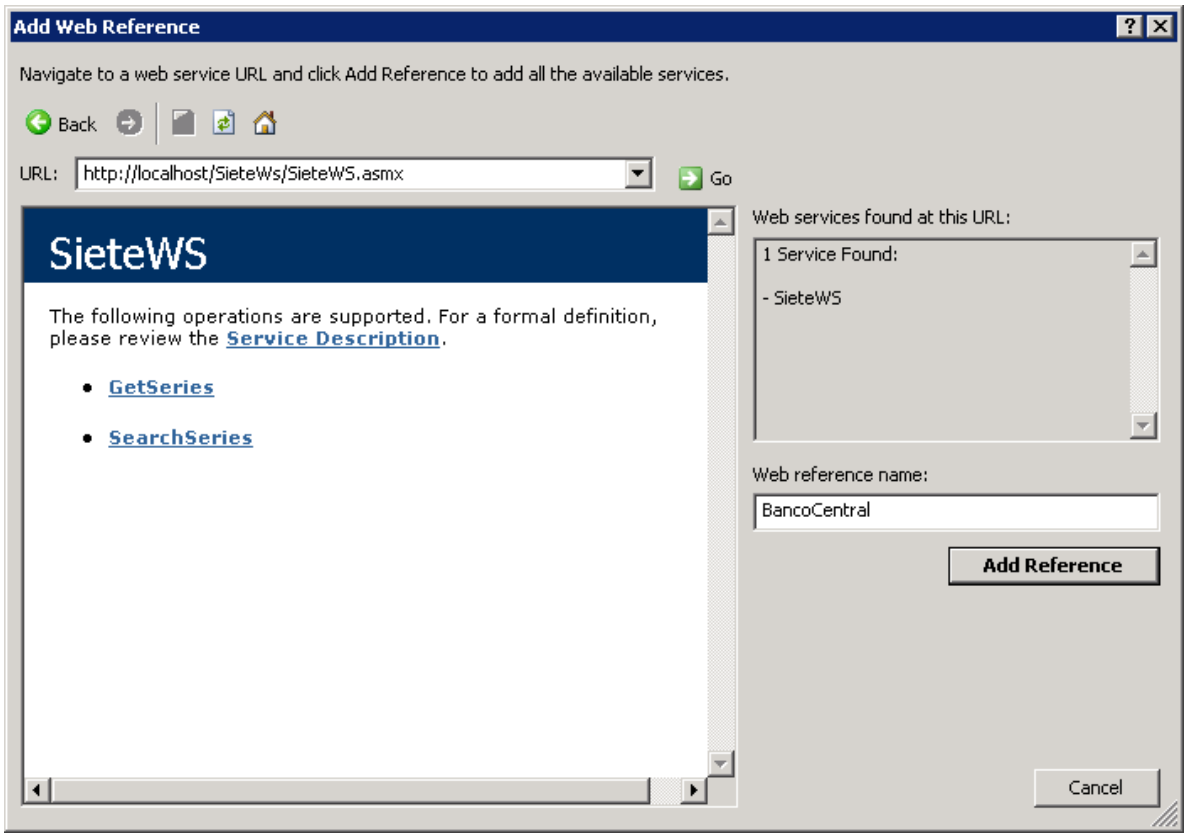
2.1 Creation of application

To create an application you must have a .Net project solution. In this example a console application will be used.



2.2 Add reference to web service

With the created application, you must add a reference to the web service available on https://si3.bcentral.cl/SieteWS/sietews.asmx?wsdl route.



Once visual studio finish adding references and create the necessary classes, they can be used from the console application.

For this example purposes the reference is created with the name BancoCentral.

2.3 Get available series

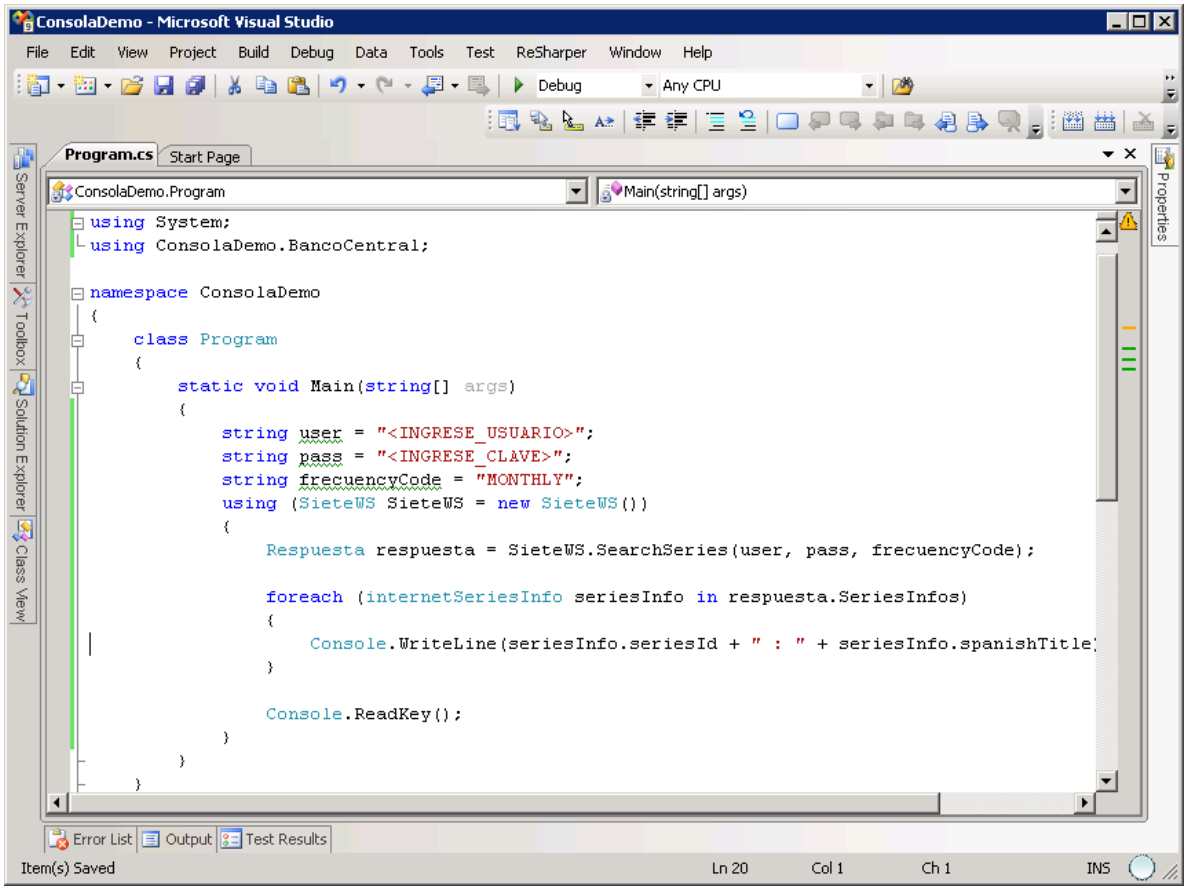
For the series available SearchSeries method is provided, which takes three parameters. These parameters are:

• user: username given by the bank to access the system
• password: user password
• frequencyCode: filter results to only those series that have that frequency

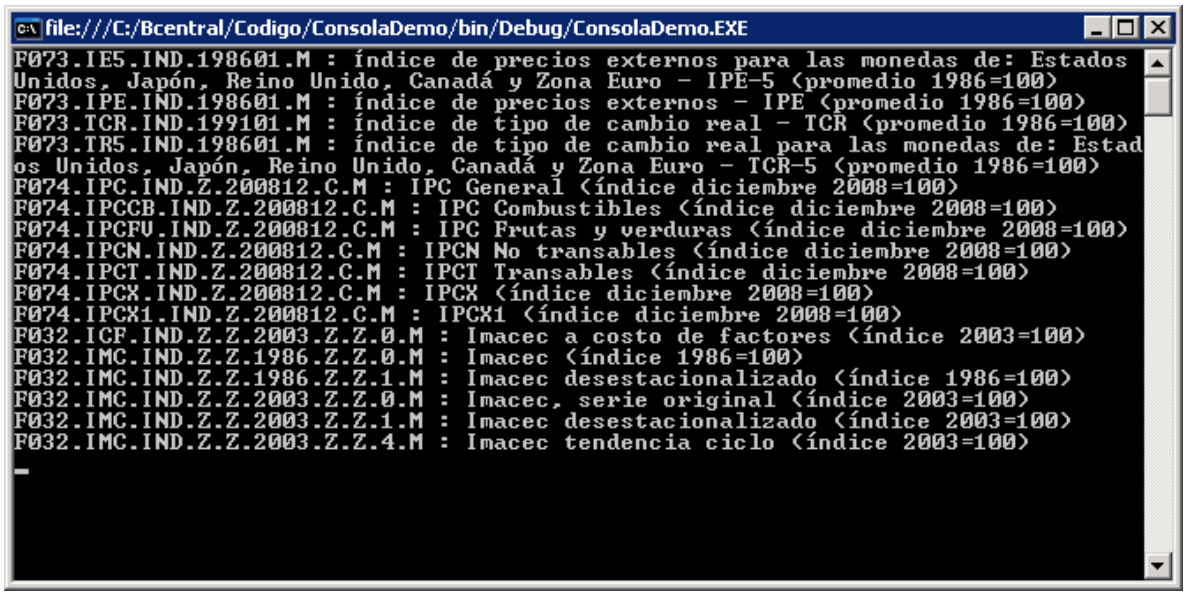
Valid FrecuencyCode are listed below:

- DAILY
- MONTHLY
- QUARTERLY
- ANNUAL

The following figure shows the block of code needed to get the data series with the selected frequency (MONTHLY for example).



Running this code displays the following list of series. In addition to the id of the series and the name in Spanish, information on the dates of observations for each series are also obtained.

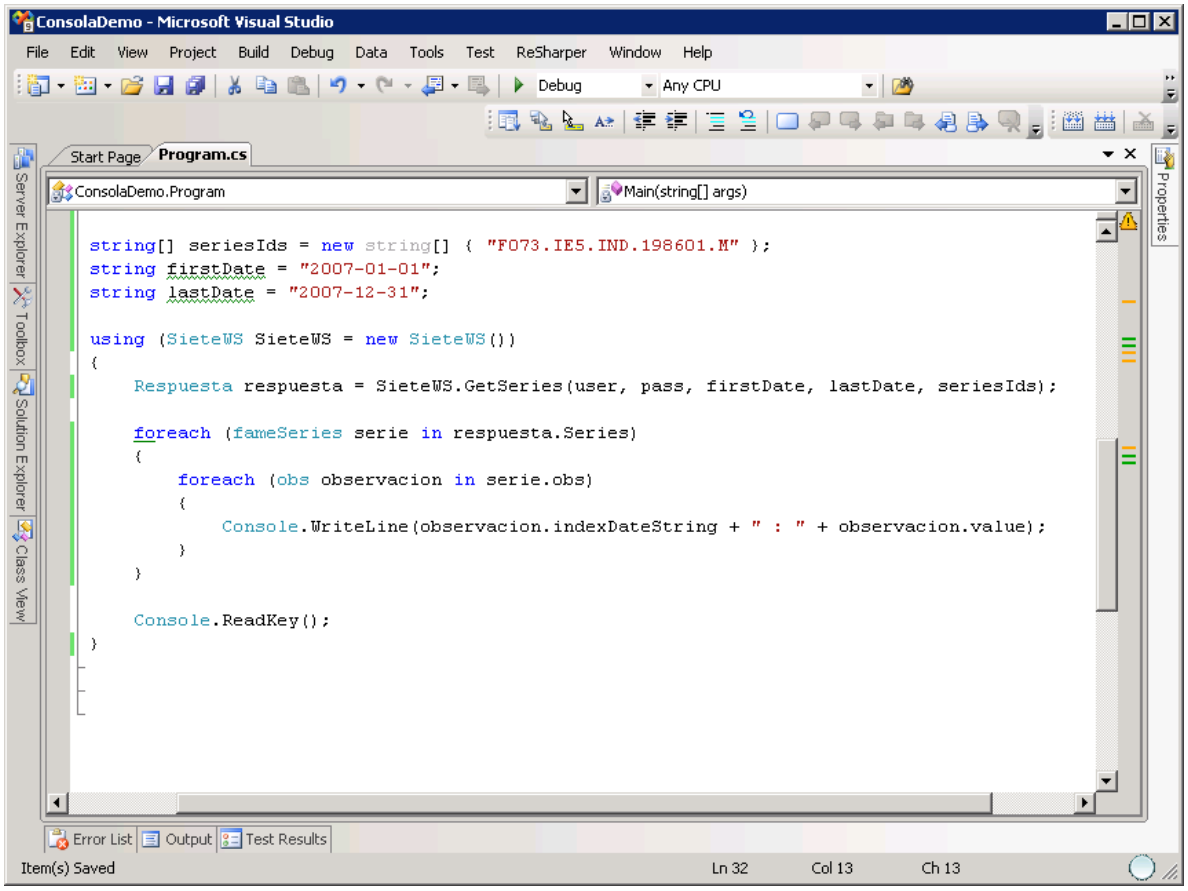


2.4 Obtaining the observations for one series

For the observations for one series GetSeries method is provided, which receives five parameters. These parameters are:

- user: username given by the bank to access the system
- password: user password
- firstDate : filter observations from a start date. The parameter is optional. The format should be YYYY–MM–DD
- lastDate : filter observations to an end date . The parameter is optional. The format should be YYYY-MM–DD
- seriesIds: filter the series of which the observations are to be obtained . The parameter is required.

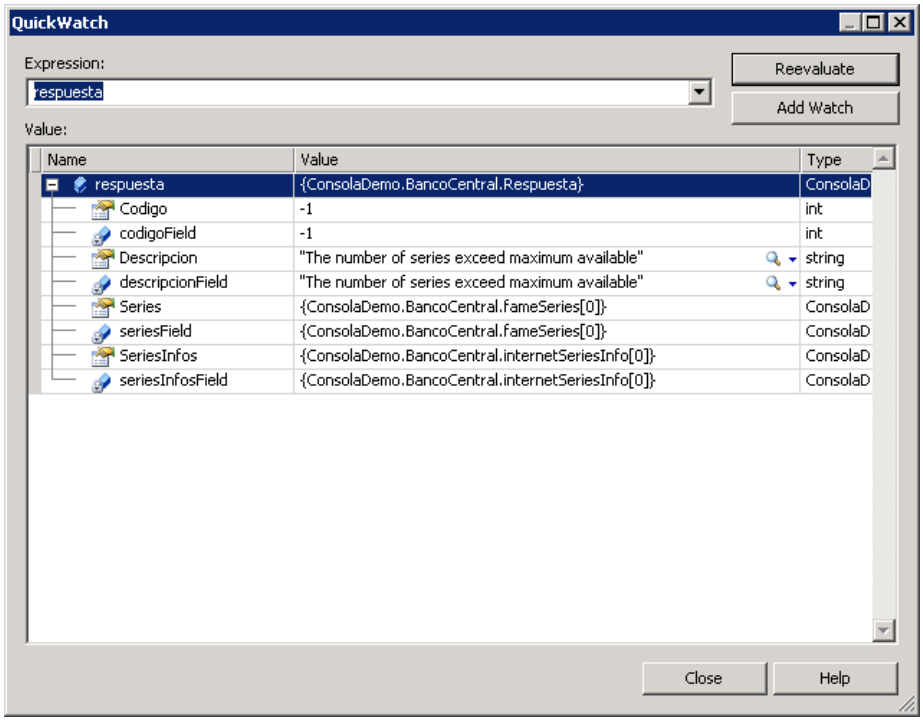
For example, if you want to get the observations for the first series of the list above, use the followind seriesId: "F073.IE5.IND.198601.M" . If you want to get all observations for year 2007, call the web service as shown in the code below.



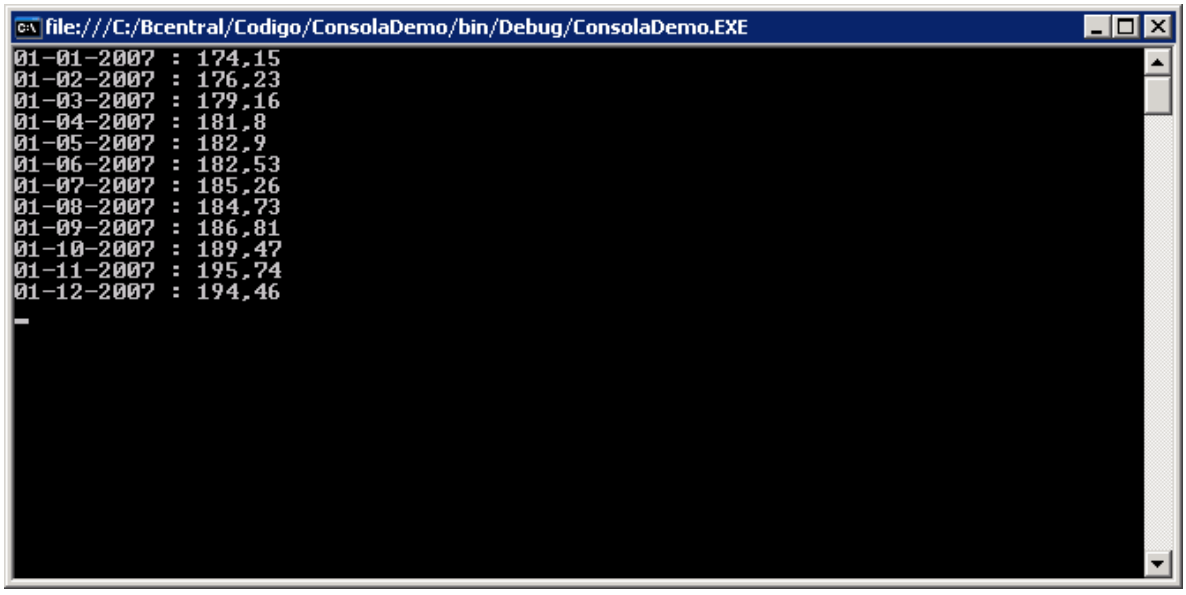
The code shows a foreach and nested foreach cycle. This is necessary because the GetSeries method allows you to select one series in seriesIds parameter. The first foreach cycles the series and the second nested foreach cycles on every timeseries observations.

There is a maximum of one (1) series which can be obtained in one call to GetSeries. If you request more than one timeseries, a message will be returned in the response indicating that the maximum amount allowed was exceeded.

Of course, you can loop through a list of sereisIds, calling every time to GetSeries with one series to obtain each series from the list.



The returning result is displayed below, whith the date of observation, one per month since it is a monthly timeseries, and the value of the observation for each period.

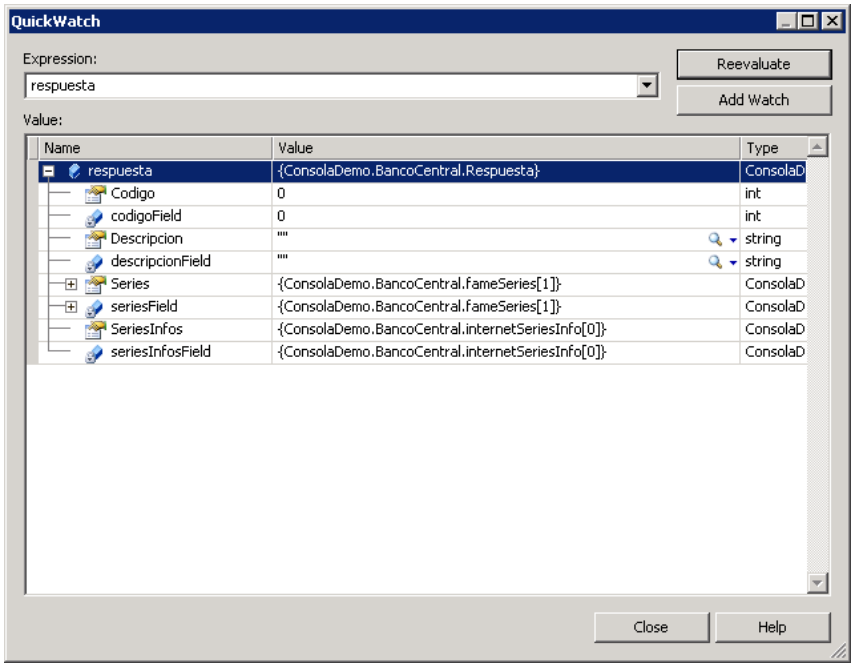


2.5 Error Handling

As seen in the example above, if you are requesting more timeseries than allowed an error will be returned.

The response contains two properties that indicate the success or failure of the execution. These properties are Code and Description. In case of success, the values will be zero and will return an empty string respectively.

This can be seen in the following image.



3. Application code

As mentioned in the previous chapter, the following code blocks fulfill similar functionality, although the execution environments are different.

3.1 C# console application code.

```
using System;
using ConsolaDemo.BancoCentral;

namespace ConsolaDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            string user = "<INGRESE_USUARIO>";
            string pass = "<INGRESE_CLAVE>";
            string frecuencyCode = "MONTHLY";
            using (SieteWS SieteWS = new SieteWS())
            {
                Respuesta respuesta = SieteWS.SearchSeries(user, pass, frecuencyCode);

                foreach (internetSeriesInfo seriesInfo in respuesta.SeriesInfos)
                {
                    Console.WriteLine(seriesInfo.seriesId + " : " + seriesInfo.spanishTitle);
                }

                Console.ReadKey();
            }

            string[] seriesIds = new string[] { "F073.IE5.IND.198601.M" };
            string firstDate = "2007-01-01";
            string lastDate = "2007-12-31";

            using (SieteWS SieteWS = new SieteWS())
            {
                Respuesta respuesta = SieteWS.GetSeries(user, pass, firstDate, lastDate, seriesIds);

                foreach (fameSeries serie in respuesta.Series)
                {
                    foreach (obs observacion in serie.obs)
                    {
                        Console.WriteLine(observacion.indexDateString + " : " + observacion.value);
                    }
                }

                Console.ReadKey();
            }
        }
    }
}
```

3.2 Php Web page code

```
<?
$user = '<INGRESE_USUARIO>';
$password='<INGRESE_CLAVE>';
$frequencyCode ='MONTHLY';

$wsdl="<DIRECCION_SITIO_WEB>?WSDL";

$client = new soapclient($wsdl);
$params = new stdClass;
$params->user = $user;
$params->password = $password;
$params->frequencyCode = $frequencyCode;
$result = $client->SearchSeries($params)->SearchSeriesResult;

foreach ($result->SeriesInfos->internetSeriesInfo as $serie)
{
   echo $serie->seriesId . " : " . $serie->spanishTitle . "<br/>";
}

$seriesIds = array ("F073.IE5.IND.198601.M");
$firstDate = "2007-01-01";
$lastDate = "2007-12-31";

$client = new soapclient($wsdl);
$params = new stdClass;
$params->user = $user;
$params->password = $password;
$params->firstDate = $firstDate;
$params->lastDate = $lastDate;
$params->seriesIds = $seriesIds;

$result = $client->GetSeries($params)->GetSeriesResult;

$fameSeries =$result->Series->fameSeries;

//Cuando se solicita una sola serie, la respuesta no es interpretada como un arreglo
if (is_array($fameSeries ) != 1) $fameSeries = array($fameSeries);

foreach ($fameSeries as $serieFame)
{
   echo $serieFame->seriesKey->seriesId . " : " . $serieFame->precision . "<br/>";
   foreach ($serieFame->obs as $obs)
   {
      echo $obs->indexDateString . " : " . $obs->value . "<br/>";
   }
}
?>
```