



## **SOAP client application development**

## Contents

1. Abbreviations/Definition/Glossary/Symbols.....	3
Warning signs.....	5
2. Scope of the document.....	6
3. OPENcontrol Web Services Interface.....	7
CNC Remote Names.....	7
The WSDL File.....	7
Backward Compatibility with Legacy Applications.....	7
4. Software Development Examples.....	8
Source codes.....	8
C# example with Visual Studio 2008 for legacy applications.....	9
Java example with NetBeans 7.3 on Ubuntu Linux.....	11
C/C++ Legacy Application Example.....	13
C# example with Visual Studio 2012 Express on Windows 7.....	14
Another C# example with Visual Studio 2012 Express on Windows 7.....	17
5. Web Service Methods defined in the WSDL file.....	20
Brief comparison of the method calls.....	21
Functions for the control of CNC bootstrap phase.....	22
Variable management.....	22
PLC oriented functions.....	22
Process dedicated functions.....	22
Functions for the “Through Mode”.....	23
Generic Functions.....	23
File system management functions.....	24




## 1. Abbreviations/Definition/Glossary/Symbols

The following is a list of abbreviation used in the document:

Reference	Description
Web Service	<p>A web service is a method of communication between two electronic devices over the World Wide Web.</p> <p>A web service is a software function provided at a network address over the web or the cloud.</p> <p><a href="http://en.wikipedia.org/wiki/Web_Service">http://en.wikipedia.org/wiki/Web_Service</a></p>
WSDL	<p>Web Services Description Language: an XML-based interface description language that is used for describing the functionality offered by a web service.</p> <p>A WSDL description of a web service (also referred to as a WSDL file) provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns.</p> <p><a href="http://en.wikipedia.org/wiki/Web_Services_Description_Language">http://en.wikipedia.org/wiki/Web_Services_Description_Language</a></p>
XML	<p>Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.</p> <p>It is defined in the XML 1.0 Specification produced by the W3C, and several other related specifications, all gratis open standards.</p> <p><a href="http://en.wikipedia.org/wiki/XML">http://en.wikipedia.org/wiki/XML</a></p>
W3C	<p>The World Wide Web Consortium (W3C) is the main international standards organization for the World Wide Web (abbreviated WWW or W3).</p> <p>Founded by Tim Berners-Lee at MIT and currently headed by him, the consortium is made up of member organizations which maintain full-time staff for the purpose of working together in the development of standards for the World Wide Web.</p> <p><a href="http://en.wikipedia.org/wiki/W3C">http://en.wikipedia.org/wiki/W3C</a></p>
SOAP	<p>SOAP, originally defined as Simple Object Access Protocol, is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks.</p> <p>It relies on Extensible Markup Language (XML) for its message format, and usually relies on other Application Layer protocols, most notably Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission.</p> <p><a href="http://en.wikipedia.org/wiki/SOAP">http://en.wikipedia.org/wiki/SOAP</a></p>
HTTP	<p>The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web (WWW).</p> <p><a href="http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol">http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol</a></p>
WWW World Wide Web	<p>The World Wide Web (abbreviated as WWW or W3, commonly known as the Web), is a system of interlinked hypertext documents accessed via the Internet. With a web browser, one can view web pages that may contain text, images, videos, and other multimedia, and navigate between them via hyperlinks.</p> <p><a href="http://en.wikipedia.org/wiki/World_Wide_Web">http://en.wikipedia.org/wiki/World_Wide_Web</a></p>
Internet	<p>The Internet is a global system of interconnected computer networks that use the standard Internet protocol suite (TCP/IP) to serve billions of users worldwide.</p> <p>It is a network of networks that consists of millions of private, public, academic, business, and government networks, of local to global scope, that are linked by a broad array of electronic, wireless and optical networking technologies.</p> <p>The Internet carries an extensive range of information resources and</p>

	<p>services, such as the inter-linked hypertext documents of the World Wide Web (WWW) and the infrastructure to support email.</p> <p><a href="http://en.wikipedia.org/wiki/Internet">http://en.wikipedia.org/wiki/Internet</a></p>
Internet protocol suite (TCP/IP)	<p>The Internet protocol suite is the set of communications protocols used for the Internet and similar networks, and generally the most popular protocol stack for wide area networks.</p> <p>It is commonly known as TCP/IP, because of its most important protocols: Transmission Control Protocol (TCP) and Internet Protocol (IP), which were the first networking protocols defined in this standard.</p> <p><a href="http://en.wikipedia.org/wiki/Internet_protocol_suite">http://en.wikipedia.org/wiki/Internet_protocol_suite</a></p>
IDE	<p>An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger.</p> <p><a href="http://en.wikipedia.org/wiki/Integrated_development_environment">http://en.wikipedia.org/wiki/Integrated_development_environment</a></p>
Visual Studio	<p>Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft.</p> <p>It is used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight it can also develop windows presentation foundation(WPF) application.</p> <p><a href="http://en.wikipedia.org/wiki/Microsoft_Visual_Studio">http://en.wikipedia.org/wiki/Microsoft_Visual_Studio</a></p>
NetBeans	<p>NetBeans is an integrated development environment (IDE) for developing primarily with Java, but also with other languages, in particular PHP, C/C++, and HTML5. It is also an application platform framework for Java desktop applications and others.</p> <p><a href="http://en.wikipedia.org/wiki/Netbeans">http://en.wikipedia.org/wiki/Netbeans</a></p>
Eclipse	<p>In computer programming, Eclipse is a multi-language software development environment comprising a base workspace and an extensible plug-in system for customizing the environment. It is written mostly in Java. It can be used to develop applications in Java and, by means of various plug-ins, other programming languages including Ada, C, C++, COBOL, Fortran, Haskell, JavaScript, Perl, PHP, Python, R, Ruby (including Ruby on Rails framework), Scala, Clojure, Groovy, Scheme, and Erlang.</p> <p><a href="http://en.wikipedia.org/wiki/Eclipse_(software)">http://en.wikipedia.org/wiki/Eclipse_(software)</a></p>
URL	<p>A uniform resource locator, abbreviated URL, also known as web address, is a specific character string that constitutes a reference to a resource.</p> <p>In most web browsers, the URL of a web page is displayed on top inside an address bar.</p> <p>An example of a typical URL would be  <a href="http://en.example.org/wiki/Main_Page">"http://en.example.org/wiki/Main_Page"</a>.</p> <p>A URL is technically a type of uniform resource identifier (URI), but in many technical documents and verbal discussions, URL is often used as a synonym for URI.</p> <p><a href="http://en.wikipedia.org/wiki/Url">http://en.wikipedia.org/wiki/Url</a></p>

## Warning signs

	<p><b>WARNING</b> Draws attention to facts or circumstances that may cause damage to the control, to the machine or to operators.</p>
	<p><b>CAUTION</b> Indicates information to be followed in order to avoid damage to equipment in general.</p>
	<p><b>IMPORTANT</b> Indicates information that must be followed carefully in order to ensure full success of the application</p>

## 2. Scope of the document

---

This document describes the new method of communication with the CNC over the network, introduced in **OPENcontrol version 3.2**.

The new method is based on cross platform Web Services (WSDL file and SOAP protocol).

Modern software integrated development environments (such as Visual Studio, NetBeans, Eclipse) can easily manage Web Services.

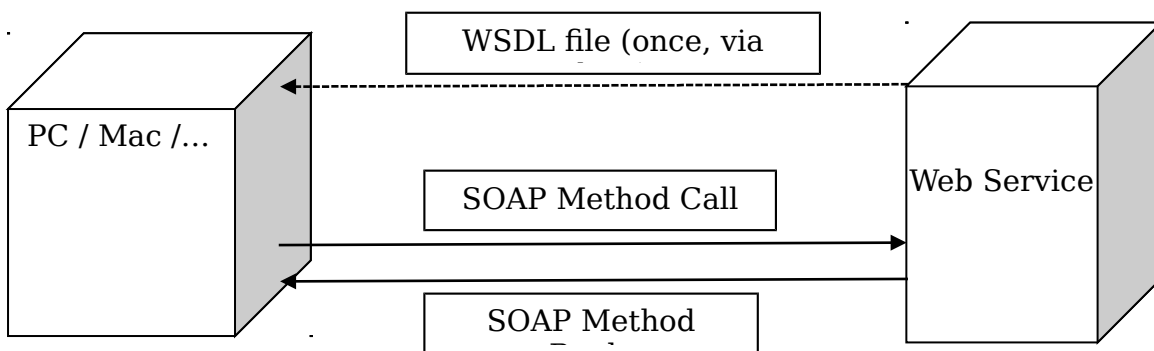
### 3. OPENcontrol Web Services Interface

The new method of communication uses Web Service by the SOAP protocol over HTTP.

Web Services are, by design, **cross platform** communication methods.



There are no a-priori limitations regarding cpu addressing (32/64 bits), cpu architecture (x86, ARM, ...), operating system (Windows, MacOSX, GNU/Linux, Android, ...), software language (C#, Java, ...).



#### CNC Remote Names

The software applications access the OPENcontrol CNC Web Service using the http url at port 8080, i.e. in this way:



<http://cnc-hostname:8080>

#### The WSDL File

A WSDL file describes the functionality offered by the CNC web service itself at the url:



<http://cnc-hostname:8080/?wsdl>

The WSDL file is somehow human readable, but it is intended as input to the modern software integrated development environments (such as Visual Studio, NetBeans, Eclipse).

#### Backward Compatibility with Legacy Applications

Legacy applications over 32bit Windows versions can still use CndexLink.dll, specifying "http://cnc-hostname:8080" as parameter RemoteName in function OpenSession\_C() and as parameter SourceTargetName and/or DestTargetName in functions LogFSTransferFile\_C() and LogFSTransferFileW\_C(). [In future releases we may add the "SP." or the "WS." prefix]

### System Settings for a SOAP Client Application

Since SOAP is built over the HTTP protocol, it is subject to the same rules that apply to a plain HTTP connection.

One aspect that requires particular attention is proxy configuration. An OPENcontrol SOAP client application generally does not want its SOAP connection(s) to be redirected to a proxy server, since the SOAP server (the CNC) is usually located in the LAN. Therefore, the operating system must be instructed to ignore the proxy server for local network connections, or better, for connections to the subnet where the CNC is located.

For example, if the CNC has an IP address of 192.168.157.2 and a subnet mask of 255.255.255.0, then the operating system must use direct connections to the 192.168.157.\* subnet.

### Configuring the Proxy Options

From the Control Panel, open Internet Properties. From the Connections tab, click LAN settings.

Now 3 different configurations can be encountered:

1. All checkboxes are disabled. In this case the system is not using a proxy server and no further action is required.
2. The "Automatically detect settings" is checked. In this case, the operating system is loading proxy settings from a network server named "wpad". You can see the settings by typing "http://wpad/wpad.dat" in your web browser. If you experience connection problems with your SOAP client application, you should contact your system administrator to help you work around the problem. Or, if you know the IP address and TCP port of the proxy server, you can use custom settings as described at the next point.
3. The "Use a proxy server for your LAN [...]" checkbox is checked, and the edit boxes below it contain the IP address and TCP port of the proxy server. In this case, click "Advanced". In the popup dialog box, under "Exceptions", add the subnet of the CNC, for example "192.168.157.\*" (without quotation marks).

Although the three cases described above cover most of the real cases, there may be more complex network configurations not covered in this document. In such cases it is better to contact the system administrator.

### Proxy Options for .NET Applications

A .NET application can be configured to automatically bypass the proxy server by modifying the App.config file.

A sample App.config with proxy bypass looks like this:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.net>
    <defaultProxy>
      <proxy bypassonlocal="true" usesystemdefault="false" />
    </defaultProxy>
  </system.net>
</configuration>
```



The<defaultProxy> tag can be placed under the<system.net> tag in an existing App.config file.

## 4. Software Development Examples

---

Each software integrated development environment manages Web Services in a slightly different way, but the **common steps** are:

- get the Web Service description, i.e. the WSDL file
- auto-generate some software “glue”
- specify the real CNC hostname, i.e. the URL content
- transparently use the generated software

Hereafter you'll find brief examples with **C#** in Visual Studio 2008 on Windows XP, with **Java** in NetBeans on Ubuntu and with **C/C++** in Visual Studio 2008 on Windows XP. Afterwards there is a bigger example with C# in Visual Studio 2012 Express on Windows 7.

### Source codes

All development projects described in this document are distributed with WinNBI v4.4 and later. Project files are located in <WinNBI-installation-directory>\UserExamples\SOAP.

## C# example with Visual Studio 2008 for legacy applications

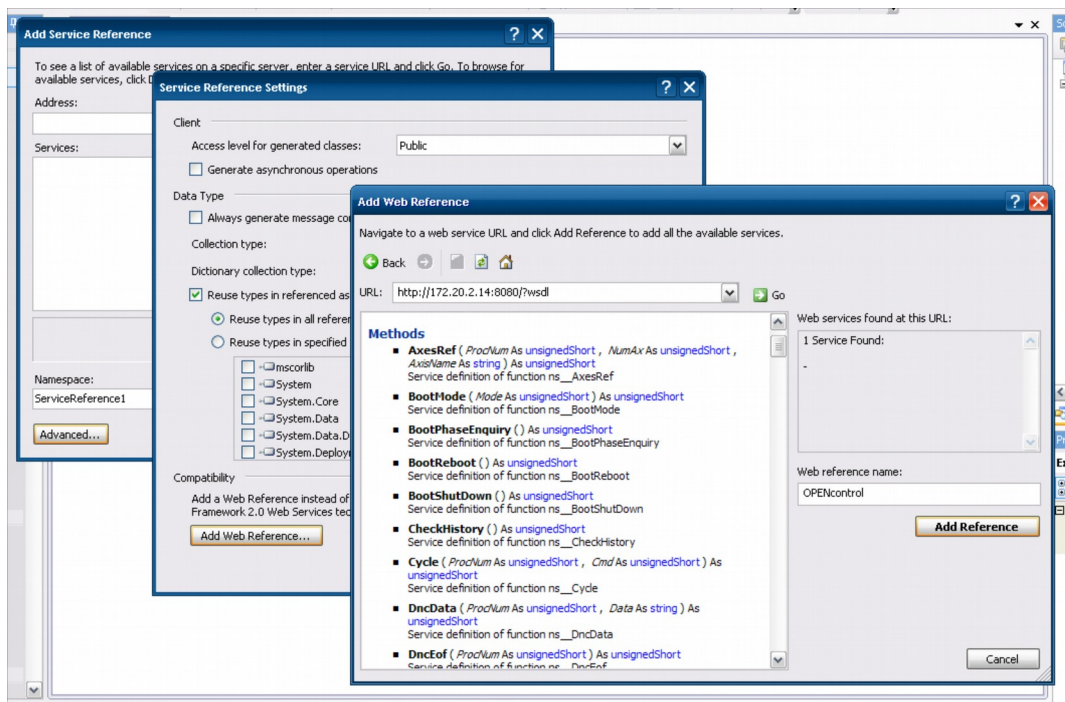
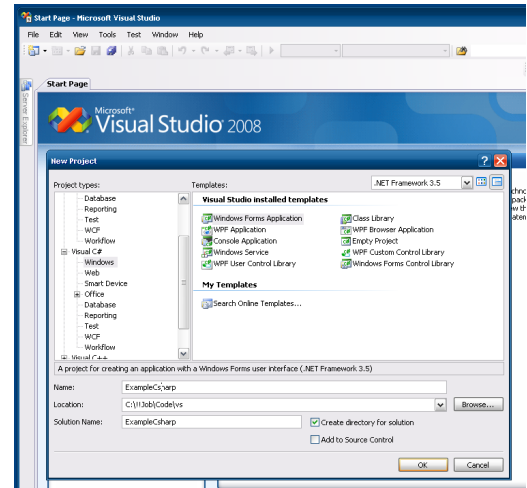
Source codes: UserExamples\SOAP\ExampleCsharp2.zip

Create a new project:

- File > New > Project...
- .NET Framework 3.5

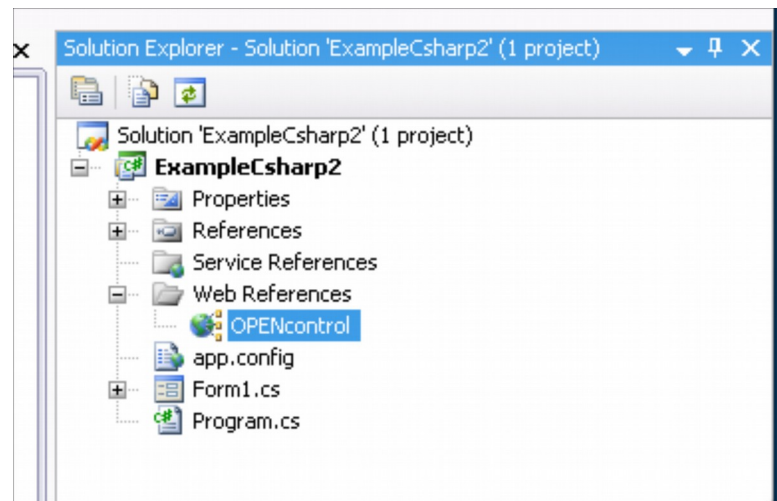
Add the Web Service Reference:

- Project > Add Service Reference ...
- Advanced ...
- Add **Web Reference** ...
- URL: <http://cnc-hostname:8080/?wsdl>
- Press "Go"
- Web Reference Name: OPENcontrol
- Add Reference



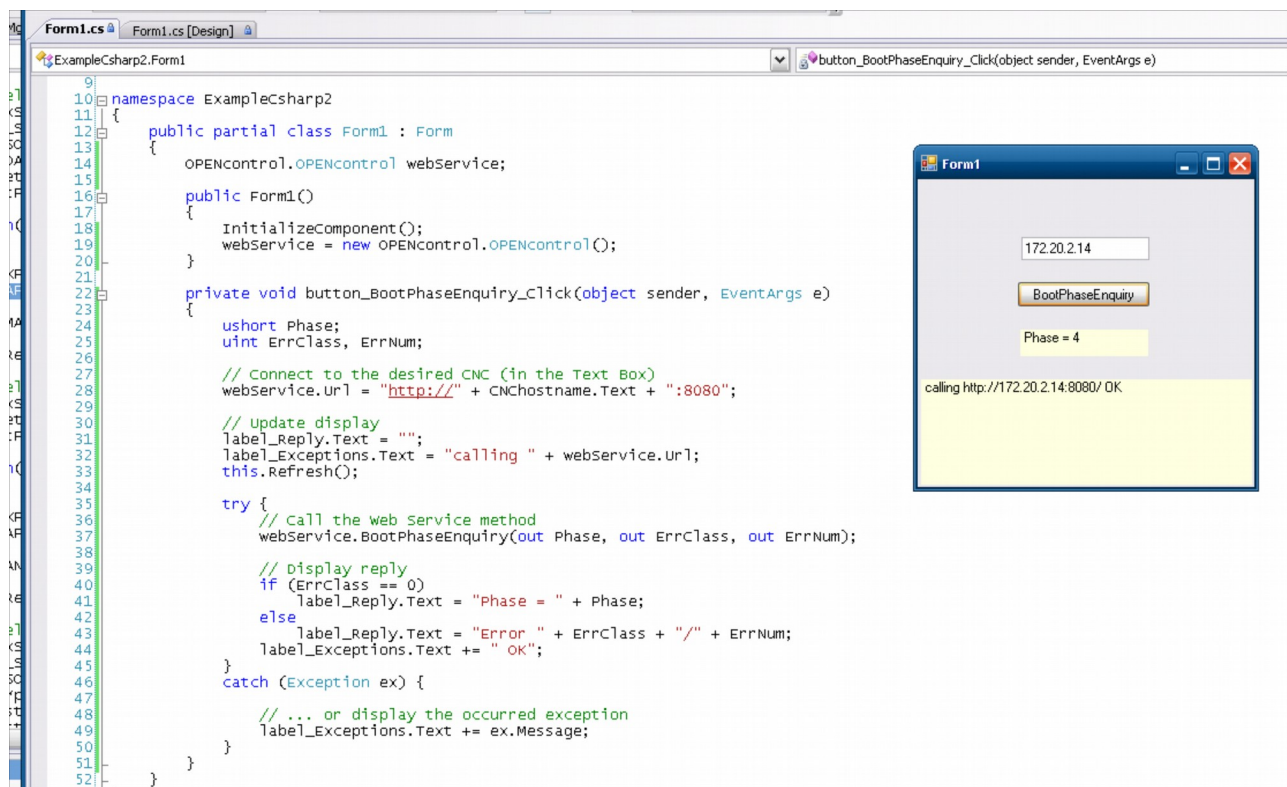
In the Solution Explorer there should be now an OPENcontrol Web Reference.

If, while creating the project, you choose the .NET Framework 2.0, then the “Add Web References” is directly available, but it is better to proceed as suggested.



Add some simple C# controls (TextBox, Button, Labels).

Call the BootPhaseEnquiry method in the Button click callback.

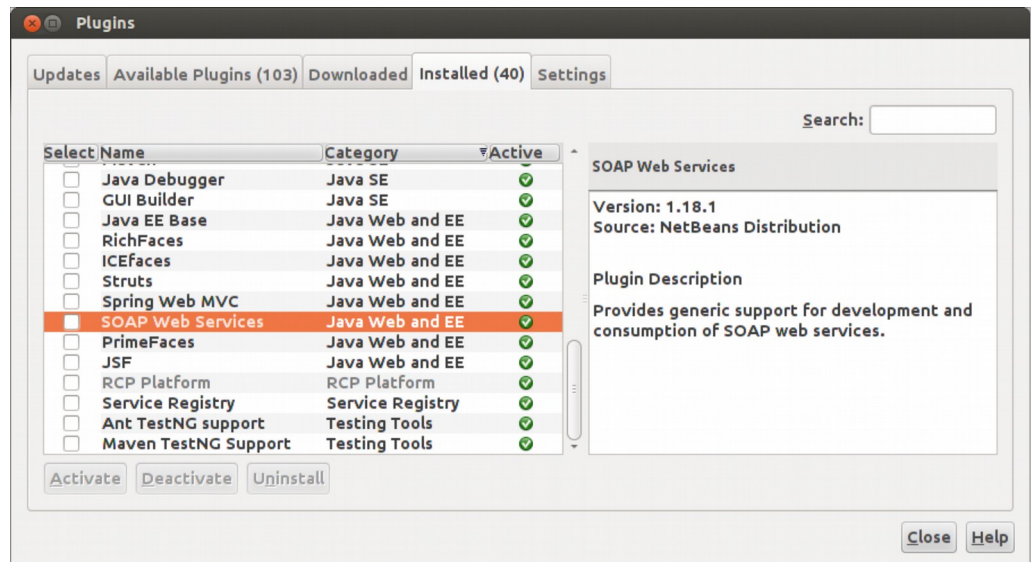


## Java example with NetBeans 7.3 on Ubuntu Linux

Source codes: `UserExamples\SOAP\ExampleJava.zip`

Install the “SOAP Web Services” Plugin (if not already present).

Create a new “Java Application” project.

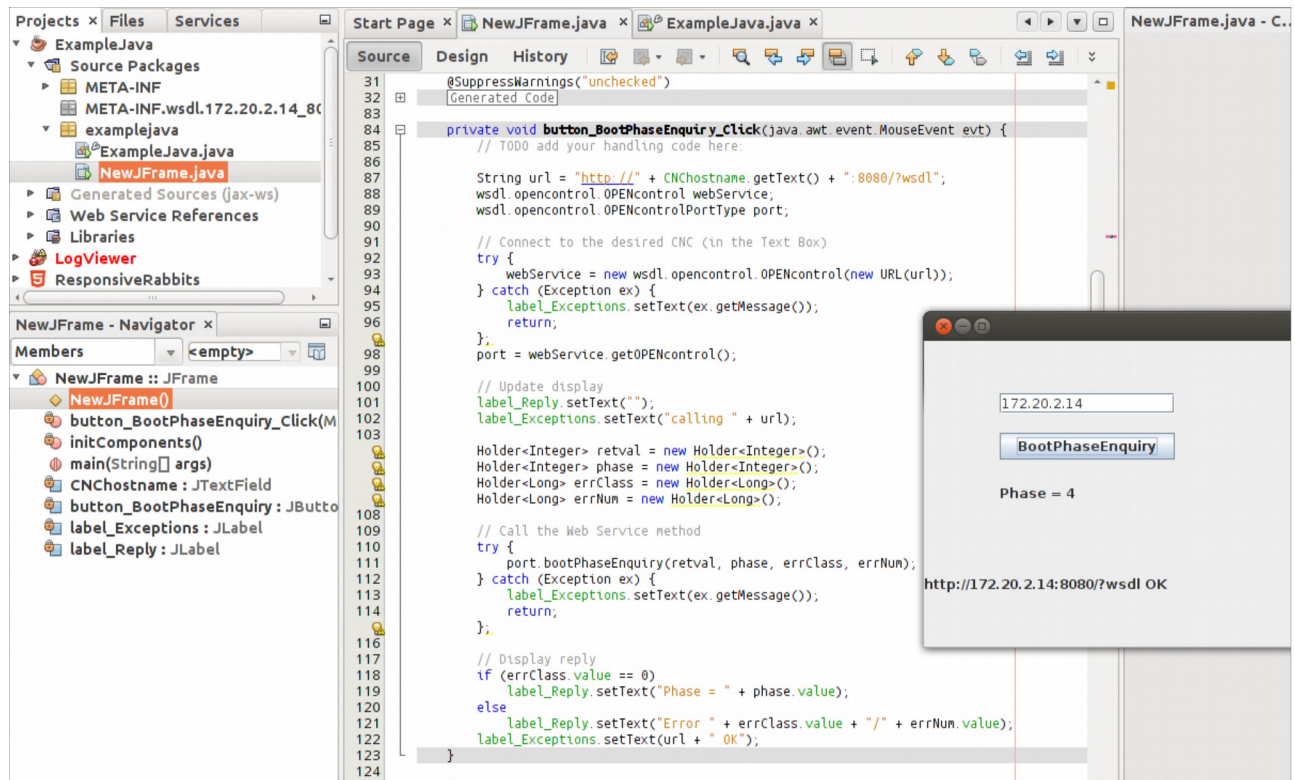


Add a new **Web Service Client**:

- from the popup on the project icon: New > Web Service Client...
- choose the WSDL URL: `http://cnc-hostname:8080/?wsdl`

Add some simple C# controls (TextBox, Button, Labels).

Call the `BootPhaseEnquiry` method in the Button click callback.



## C/C++ Legacy Application Example

Source codes: UserExamples\SOAP\ExampleC++.zip

The Cndex Link User library provides backward compatibility for legacy C++/C# custom applications: just use "http://cnc-hostname:8080" as RemoteName in OpenSession\_C() call.

```
// ExampleC++.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"
#include "windows.h"

#include "C:\Program Files\OSAI\WinNBI\UserLibrary\CndexLinkUser.h"
// Project properties>Linker>Input>Additional Dependencies:
// "C:\Program Files\OSAI\WinNBI\UserLibrary\CndexLink.lib"

int _tmain(int argc, _TCHAR* argv[])
{
    // Project Properties > General > Character Set : Using MultiByte Charater Set
    char *cnc_hostname = (argc > 1) ? argv[1] : "172.20.2.14";

    DWORD ErrClass, ErrNum;
    char RemoteName[256];
    WORD UserSession;
    WORD Phase;

    // Specify the Web Service connection to the desired CNC
    sprintf_s(RemoteName, sizeof(RemoteName), "http://%s:8080", cnc_hostname);

    // Connect to the desired CNC
    printf("OpenSession_C(%) ... ", RemoteName);
    if (!OpenSession_C(RemoteName, &UserSession, &ErrClass, &ErrNum)) {
        printf("Error %u/%u\n", ErrClass, ErrNum);
        return 1;
    }
    printf("OK, UserSession=%u\n", UserSession);

    // Call the Web Service method via DLL call
    printf("BootPhaseEnquiry_C() ... ");
    if (!BootPhaseEnquiry_C(UserSession, &Phase, &ErrClass, &ErrNum)) {
        printf("Error %u/%u\n", ErrClass, ErrNum);
        return 1;
    }
    printf("OK, Phase=%u\n", Phase);

    // Close Connection
    printf("CloseSession_C() ... ");
    if (!CloseSession_C(UserSession, &ErrClass, &ErrNum)) {
        printf("Error %u/%u\n", ErrClass, ErrNum);
        return 1;
    }
    printf("OK\n");
    return 0;
}
```

An example of usage of the sample legacy application:

```
C:\...\ExampleC++\Debug>ExampleC++.exe
OpenSession_C(http://172.20.2.14:8080) ... OK, UserSession=16512
BootPhaseEnquiry_C() ... OK, Phase=4
CloseSession_C() ... OK
```

```
C:\...\ExampleC++\Debug>ExampleC++.exe not-existent-cnc
OpenSession_C(http://not-existent-cnc:8080) ... OK, UserSession=16512
BootPhaseEnquiry_C() ... Error 11/28
```

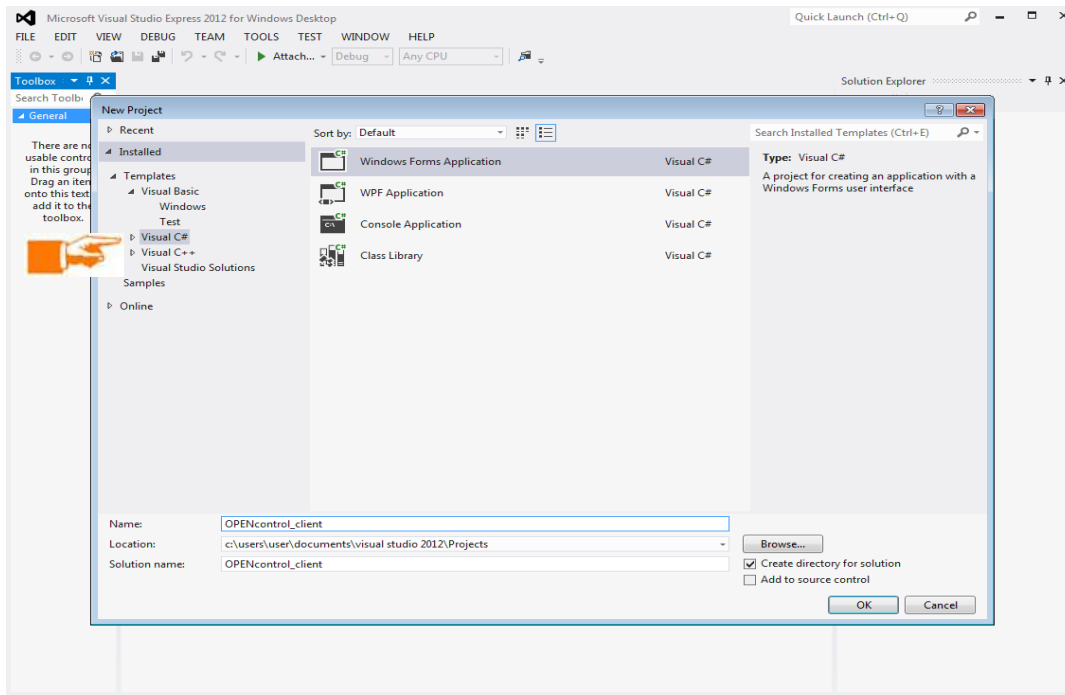
```
C:\...\ExampleC++\Debug>
```



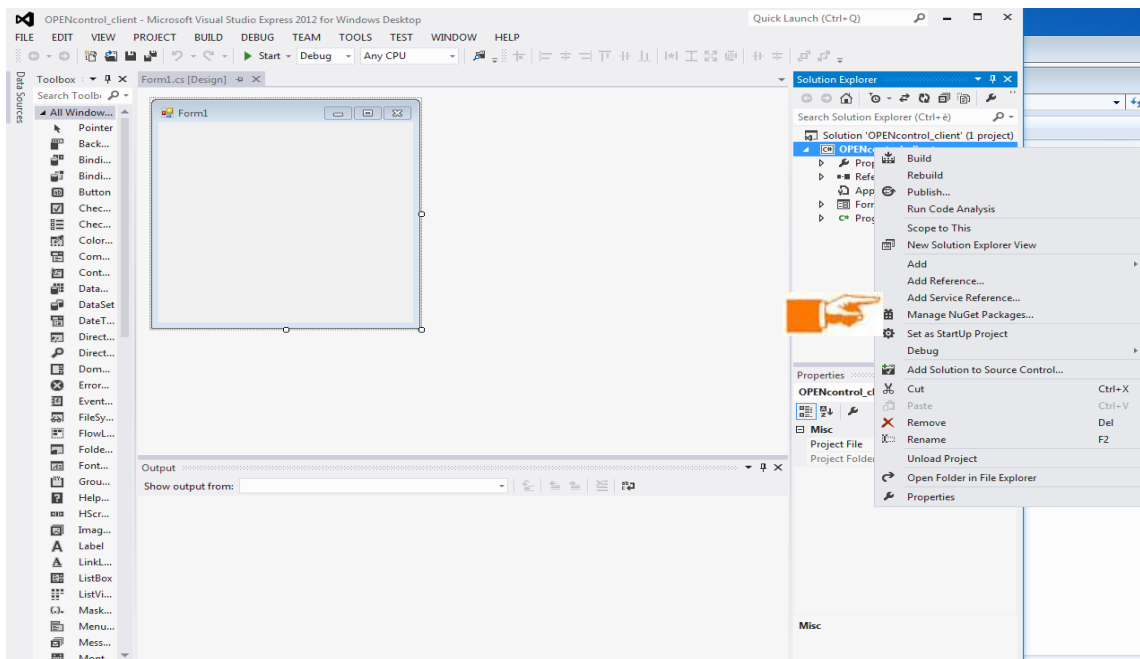
## C# example with Visual Studio 2012 Express on Windows 7

Source codes: [UserExamples\SOAP\ ExampleCsharpVS2012E\\_lite.zip](#)

Start a new C# Windows Form Application:

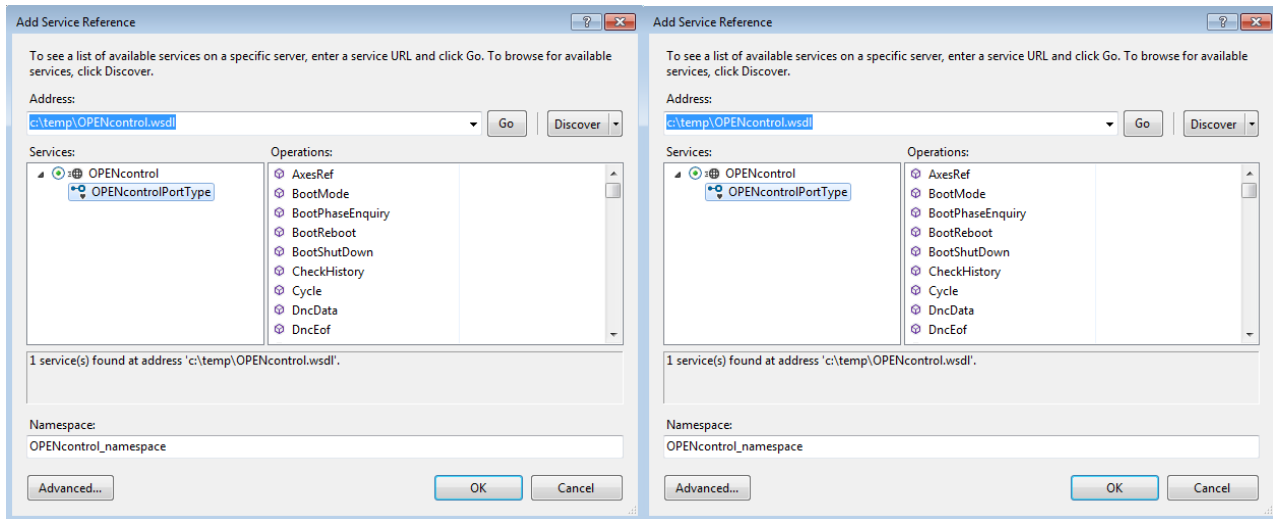


Add a **Service Reference** to the OPENcontrol Web Service:



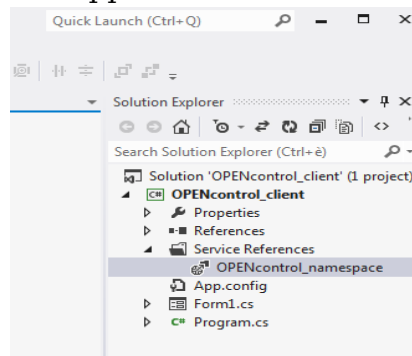


You can specify the WSDL either by the WSDL file on the computer or by the http link to a running OPENcontrol CNC: in both cases click Go for acquiring the service.

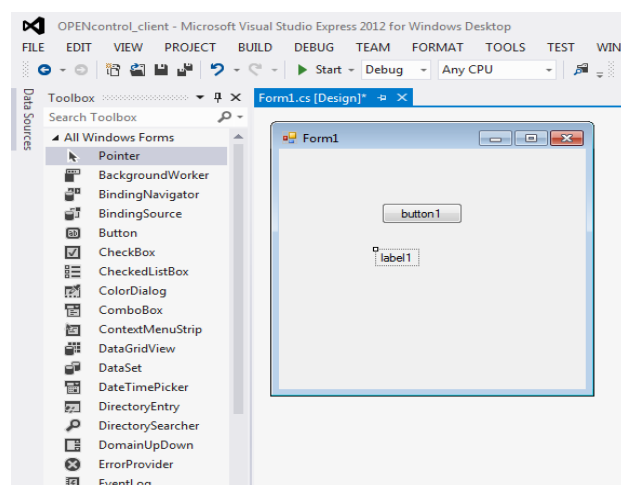


Then specify a useful namespace (for instance “OPENcontrol\_namespace”) and click “OK”.

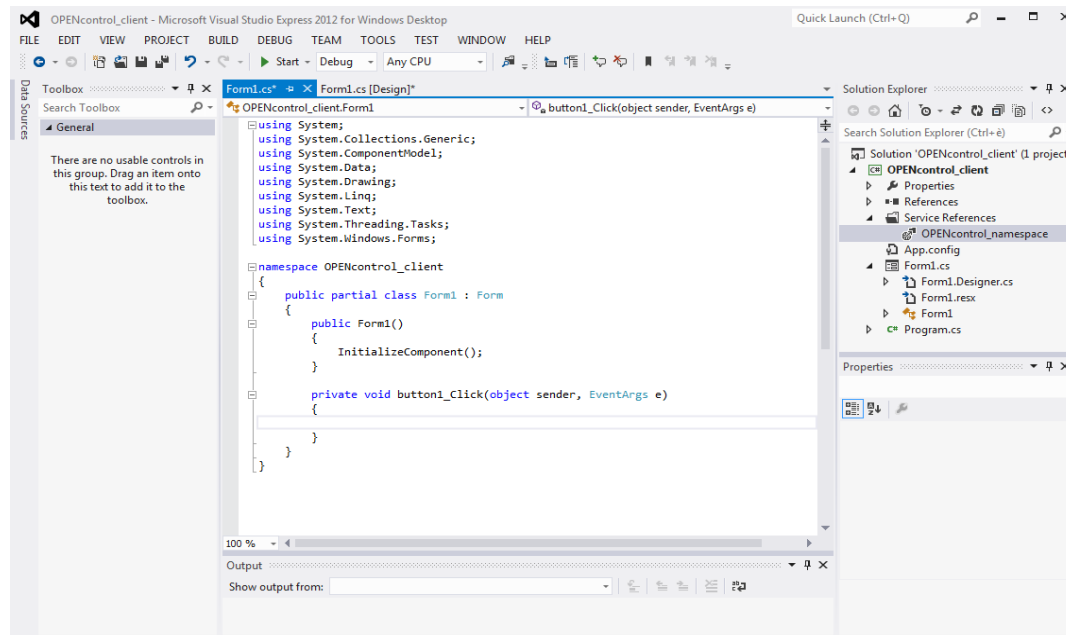
The OPENcontrol Web Service appears now in the Solution Explorer.



Now add a Button and a Label:



Double clicking the Button, Visual Studio creates the “click” callback and opens the



editor on it.

Add a “using” statement for the Web Service namespace:

```
using ExampleCsharp.OPENcontrol_namespace;
```

Then add the following code in the button1\_Click callback:

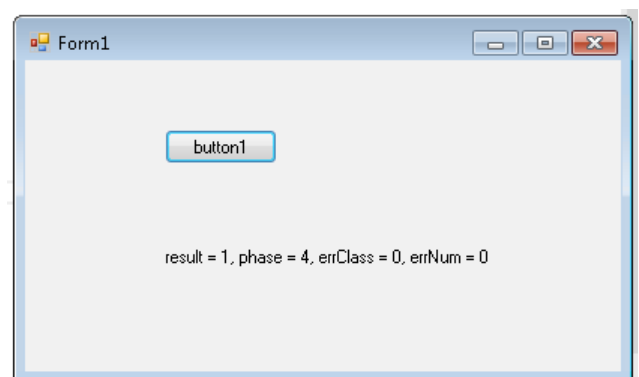
```
// create a new web service client
OPENcontrolPortTypeClient webService;
string conf = "OPENcontrol";
string url = "http://172.20.2.14:8080";

webService = new OPENcontrolPortTypeClient(conf, url);

// call the BootPhaseEnquiry method
ushort phase;
ushort result;
uint errClass, errNum;

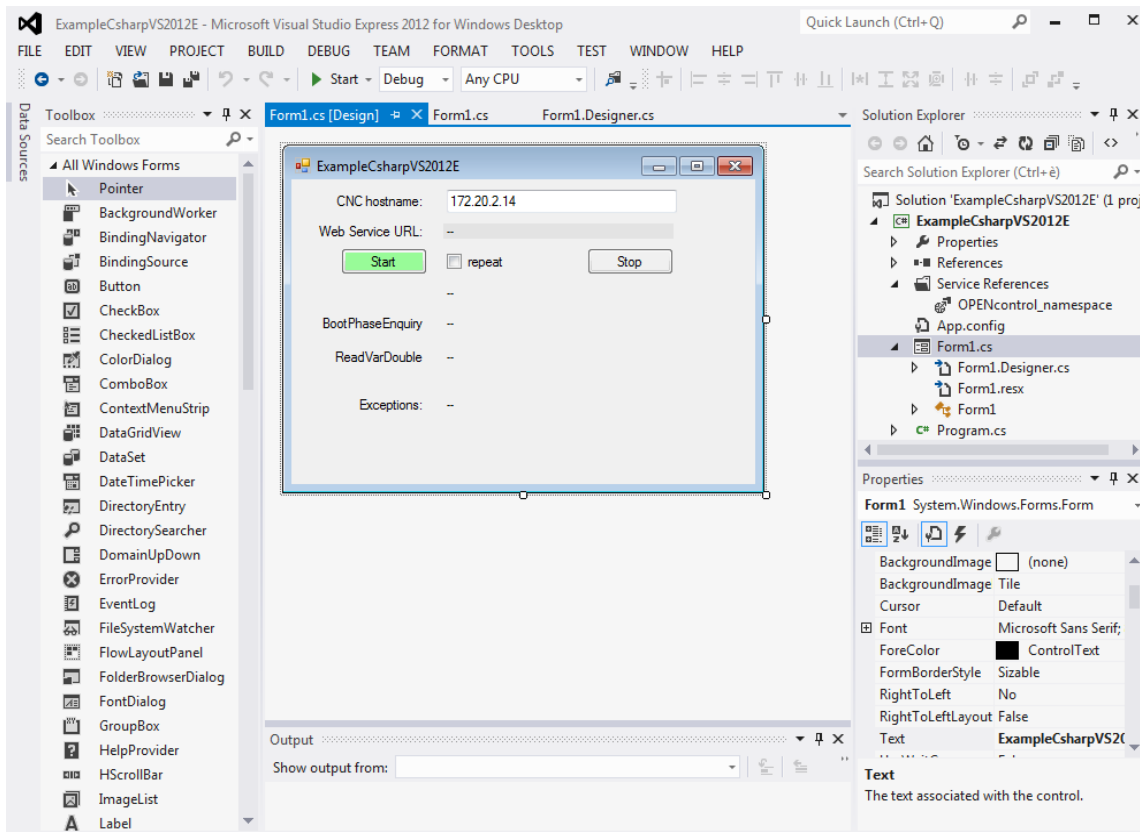
result =
webService.BootPhaseEnquiry(out
phase, out errClass, out errNum);
textBox1.Text =
    "result = " + result
    + ", phase = " + phase
    + ", errClass = " + errClass
    + ", errNum = " + errNum;
```

Now build and run with **F5**.



## Another C# example with Visual Studio 2012 Express on Windows 7

Source codes: [UserExamples\SOAP\ ExampleCsharpVS2012E\\_full.zip](#)



Let's write a more realistic example managing loops and exceptions.

The program in this project:

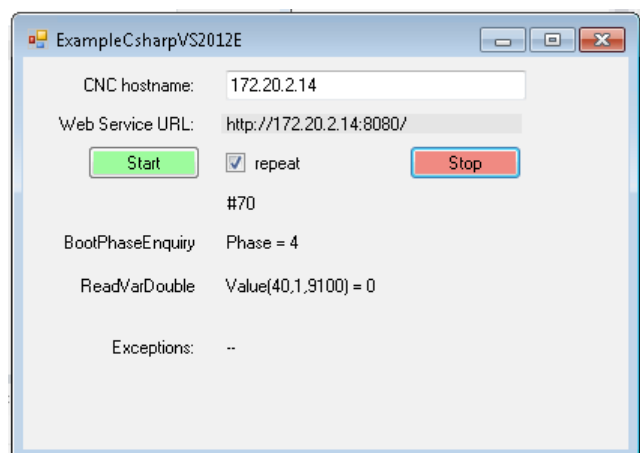
- let you change the CNC hostname at run time
- let you continuously call the methods
- manages the exceptions
- translates OPENcontrol hostnames "IP.hostname" to urls "http://hostname:8080"

The C# variables for the user interface controls are:

```

TextBox CNChostname;
Label label_URL;
Label label_Reply1;
Button Start;
CheckBox repeat;
Button Stop;
Label label_Counter;
Label label_Exceptions;
Label label_Reply2;

```



The important parts of the source code are:

```
// the Service Reference
using ExampleCsharpVS2012E.OPENcontrol_namespace;
```

The url creation function:

```
// prepare the url from the hostname
private string new_url(string hostname)
{
    string url = String.Copy(hostname);
    if (url.ToUpper().Equals("IP."))
        url = "IP.127.0.0.1";
    if (url.ToUpper().StartsWith("IP."))
        url = url.Substring(3);
    if (!url.ToLower().StartsWith("http://"))
        url = "http://" + url;
    if (!url.ToLower().EndsWith(":8080"))
        url = url + ":8080";
    return url;
}
```

The web service client creation function:

```
private OPENcontrolPortTypeClient new_client(string hostname)
{
    OPENcontrolPortTypeClient retval;
    string url = new_url(hostname); ;

    // update the user interface before
    label_URL.Text = "trying " + url;
    Application.DoEvents();

    // create the web service client
    retval = new OPENcontrolPortTypeClient("OPENcontrol", url);

    // update the user interface after
    label_URL.Text = retval.InnerChannel.RemoteAddress.ToString();
    return retval;
}
```

The “repeat” checkbox and “Stop” button callback functions:


```
private bool doStop;


private void repeat_CheckedChanged(object sender, EventArgs e)
{
    Stop.Enabled = repeat.Checked;
    if (Stop.Enabled)
        Stop.BackColor = Color.LightCoral;
    else
        Stop.BackColor = SystemColors.Control;
}


private void Stop_Click(object sender, EventArgs e)
{
    doStop = true;
}
```

The “Start” button callback function:

```
private void Start_Click(object sender, EventArgs e)
{
```

```
// user interface management
Start.Enabled = false;
doStop = false;
label_Counter.Text = "";
label_Reply1.Text = "";
label_Reply2.Text = "";
try
{
    // create a new web service client
     OPENcontrolPortTypeClient webService = new_client(CNChostname.Text);
    int counter = 1;
    do
    {
        // common variables
        ushort result;
        uint ErrClass, ErrNum;
        label_Counter.Text = "#" + counter;

        // call BootPhase Enquiry
        ushort Phase;
         result = webService.BootPhaseEnquiry(out Phase, out ErrClass, out ErrNum);
        if (result > 0)
            label_Reply1.Text = "Phase = " + Phase;
        else
            label_Reply1.Text = "Error = " + ErrClass + "/" + ErrNum;

        // call ReadVarDouble
        ushort Code = 40;
        ushort Process = 1;
        ushort Index = 9100;
        ushort NumVar = 1;
        doublearray Value;
         result = webService.ReadVarDouble(Code, Process, Index, NumVar,
                                     out Value, out ErrClass, out ErrNum);
        if (result > 0)
            label_Reply2.Text = "Value(" + Code + "," + Process + "," +
                               Index + ") = " + Value[0];
        else
            label_Reply2.Text = "Error = " + ErrClass + "/" + ErrNum;

        // wait a bit and loop
        Application.DoEvents();
        Thread.Sleep(50);
        ++counter;
    } while (repeat.Checked && !doStop);
}
catch (Exception ex)
{
    label_Exceptions.Text = ex.Message;
}
Start.Enabled = true;
}
```

---

## 5. Web Service Methods defined in the WSDL file

---

The Web Service methods are strictly related to the **CndexLinkUser** functions: whenever possible method names, arguments names and order are the same<sup>1</sup>. Developers can use the **CndexLinkUser.pdf** (italian) or **CndexLinkUserEnglish.pdf** (english) manual as reference for web service methods, with a few caveats.

The general rules for Web Service methods are:

- The method names are equal to **CndexLinkUser** function names without the “\_C” suffix<sup>2</sup>
- There are no `OpenSession()` and `CloseSession()` methods<sup>3</sup>
- There is no `UserSession` argument
- The type definitions have no “\_C4” suffix and no middle “\_”
- If a method returns a memory area then the method itself allocates the area<sup>4</sup>
- The new methods `PutFile` and `GetFile` replace the two irreproducible functions `LogFSTransferFile_C()` and `LogFSTransferFileW_C()`<sup>5</sup>.

Some more for the Java scenario:

- The first character of method names is lowercase
- The return value is the first “out” argument, “retval”
- The “out” arguments need to use the “Holder” wrapper<sup>6</sup>

---

<sup>1</sup> But, as you may have noticed in the examples, each IDE slightly transforms the methods names and arguments.

<sup>2</sup> The obsolete 10 Series functions have been omitted.

<sup>3</sup> But they are present in the backward compatibility library.

<sup>4</sup> This comes easy in C# and Java and avoids the useless transmission of zeroed buffers to the soap server

<sup>5</sup> the `LogFSTransferFile*_C()` methods are irreproducible in a web service because the software in the soap server cannot directly access the files in the client pc; anyhow the `LogFSTransferFile*_C()` functions in the backward compatibility library accept web service targets (“http://cnc-hostname:8080”).

<sup>6</sup> With SOAP it is possible to return multiple values in a single request. This is impossible in Java as a method can only return one object. JAX-WS solves this problem with the concept of Holders. A `javax.xml.ws.Holder` is a simple wrapper object that can be passed into the `@WebService` method as a parameter. The application sets the value of the holder during the request and the server will send the value back as an OUT parameter.

<http://tomee.apache.org/examples-trunk/webservice-holder/>

## Brief comparison of the method calls

```
// -----
// direct DLL CndexLink in C++                                OLD, BACKWARD COMPATIBLE
#include "CndexLinkUser.h"

LPSTR RemoteName; WORD UserSession;
WORD Retval; DWORD ErrClass, ErrNum;
WORD Code = 20; WORD Process = 1; WORD Index = 19100;
WORD *pValue; WORD NumVar = 42;

Retval = OpenSession_C(RemoteName, &UserSession, &ErrClass, &ErrNum);
pValue = (WORD *)malloc(NumVar * sizeof(WORD));
Retval = ReadVarWord_C(UserSession, Code, Process, Index, NumVar, pValue, &ErrClass, &ErrNum);
// the variable 0 value is in: pValue[0]

// -----
// DLL CndexLink import in C#                                OLD, DEPRECATED
using CndexLinkUserDotNet;

string RemoteName; ushort UserSession;
ushort Retval; uint ErrClass, ErrNum;
ushort Code = 20; ushort Process = 1; ushort Index = 19100;
ushort[] Value; ushort NumVar = 42;

Cndex ds = new Cndex();
Retval = ds.OpenSession_C(RemoteName, out UserSession, out ErrClass, out ErrNum);
Value = new ushort[NumVar];
Retval = ds.ReadVarWord_C(UserSession, Code, Process, Index, NumVar, Value, out ErrClass, out ErrNum);
// the variable 0 value is in: Value[0]

// -----
// direct Web Service method call in C#                        NEW
// Web Reference to OPENcontrol

string RemoteName;
ushort Retval; uint ErrClass, ErrNum;
ushort Code = 20; ushort Process = 1; ushort Index = 19100;
ushort[] Value; ushort NumVar = 42;

OPENcontrol.OPENcontrol ws = new OPENcontrol.OPENcontrol();
ws.Url = RemoteName;
Retval = ws.ReadVarWord(Code, Process, Index, NumVar, out Value, out ErrClass, out ErrNum);
// the variable 0 value is in: Value[0]

// -----
// direct Web Service method call in Java                      NEW
// Web Service Reference to OPENcontrol

String remoteName;
Holder<Integer> retval = new Holder<Integer>();
Holder<Long> errClass = new Holder<Long>(); Holder<Long> errNum = new Holder<Long>();
int code = 20; int process = 1; int index = 19100;
Holder<opencontrol.UnsignedShortArray> value = new Holder<opencontrol.UnsignedShortArray>();
int numVar = 42;

wsdl.opencontrol.OPENcontrol ws = new wsdl.opencontrol.OPENcontrol(new URL(remoteName));
wsdl.opencontrol.OPENcontrolPortType port = ws.getOPENcontrol();
port.readVarWord(code, process, index, numVar, retval, value, errClass, errNum);
// the variable 0 value is in: value.value.getItem().get(0)
```

### Functions for the control of CNC bootstrap phase

- BootPhaseEnquiry
- BootReboot
- BootShutDown
- BootMode
- GetHWKey

### Variable management

- ReadVarWord
- ReadVarDouble
- WriteVarWord
- WriteVarWordBit
- WriteVarDouble
- ReadVarText
- WriteVarText

### PLC oriented functions

- ReadWarningMsg
- ResetSingleTableII
- LockTableII
- UnLockTableII
- GetOriginTabRecordII
- SetOriginTabRecordII
- GetToolTabRecordII
- SetToolTabRecordII
- GetOffsetTabRecordII
- SetOffsetTabRecordII
- GetUserTabRecordII
- SetUserTabRecordII
- SaveTables
- RestoreSingleTable
- SaveSingleTable
- SaveBackupMemory
- RestoreBackupMemory

### Process dedicated functions

- Cycle
- SyncroCycle
- Reset
- Hold
- SetFeedManOver
- SetFeedRateOver
- SetFeedRapidOver



- SetSpeedRateOver
- SetManMovDirection
- GetVarJOG
- SetVarJOG
- SetVarUAS
- GetVarRCM
- SetVarRCM
- GetProcVarWord
- SetProcVarWord
- GetProcVarDouble
- SetProcVarDouble
- SetMdiString
- SetProcessMode
- SelectProcess
- GetSelectedProcess
- SelectProcAxis
- SelectPartProgram
- SelectPartProgramFromDrive
- GetActivePartProgram
- GetPartProgramLines
- GetAxOriginNum
- GetAxesPosition
- GetNcInfo1
- GetNcInfo2
- GetToolNames
- GetProcessStatus
- GetBlkNum
- ReadErrMsg
- ReadPartProgramMsg
- GetGCode
- GetMCode
- SkipPProgBlock
- Ese
- EseEx
- Exe
- AxesRef

### Functions for the “Through Mode”

- DncInit
- DncData
- DncEof
- DncStop

### Generic Functions

- GetAxesInfo3

- GetCodeNumber
- GetOptions
- GetDateTime
- SetDateTime

### File system management functions

- PutFile
- GetFile
- PutBinaryFile
- GetBinaryFile
- LogFSSetSecurityLevel
- LogFSGetSecurityLevel
- LogFSLongFileNames
- LogFSGetNumDrive
- LogFSGetDriveList
- LogFSGetHiddenDriveList
- LogFSGetDrivePath
- LogFSAddDrive
- LogFSRemoveDrive
- LogFSReloadDriveList
- LogFSCreateDir
- LogFSCreateFile
- LogFSGetFileSize
- LogFSGetFileAttrib
- LogFSSetFileAttrib
- LogFSChangeFileAttrib
- LogFSFindFirst
- LogFSFindNext
- LogFSFindClose
- LogFSRemoveFile
- LogFSRemoveDir
- LogFSRename
- LogFSCopyFile
- LogFSGetInfo