

# How to consume a Lawson Web Service from a Personalized Script in Smart Office

Thibaud Lopez Schneider  
Lawson Software  
June 15, 2010

In this paper I illustrate a new solution to consume a Lawson Web Service from a Personalized Script in JScript.NET in Lawson Smart Office. This new solution complements the other three known solutions. This one is interesting because it minimizes code source surface while still ensuring SOAP validation. And the solution does not involve any C# coding, nor does it require Microsoft Visual Studio. For this new solution, we will use the free Microsoft Web Services Description Language Tool (wsdl.exe) to generate a proxy class in C# that we'll use from JScript.NET.

## **Table of contents**

Background

Step 1. Install the .NET SDK

Step 2. Test the Lawson Web Service

Step 3. Generate a proxy class

Step 4. Compile the C#

Step 5. Publish the DLL

Step 6. Create a Personalized Script

Step 7. Test

Summary

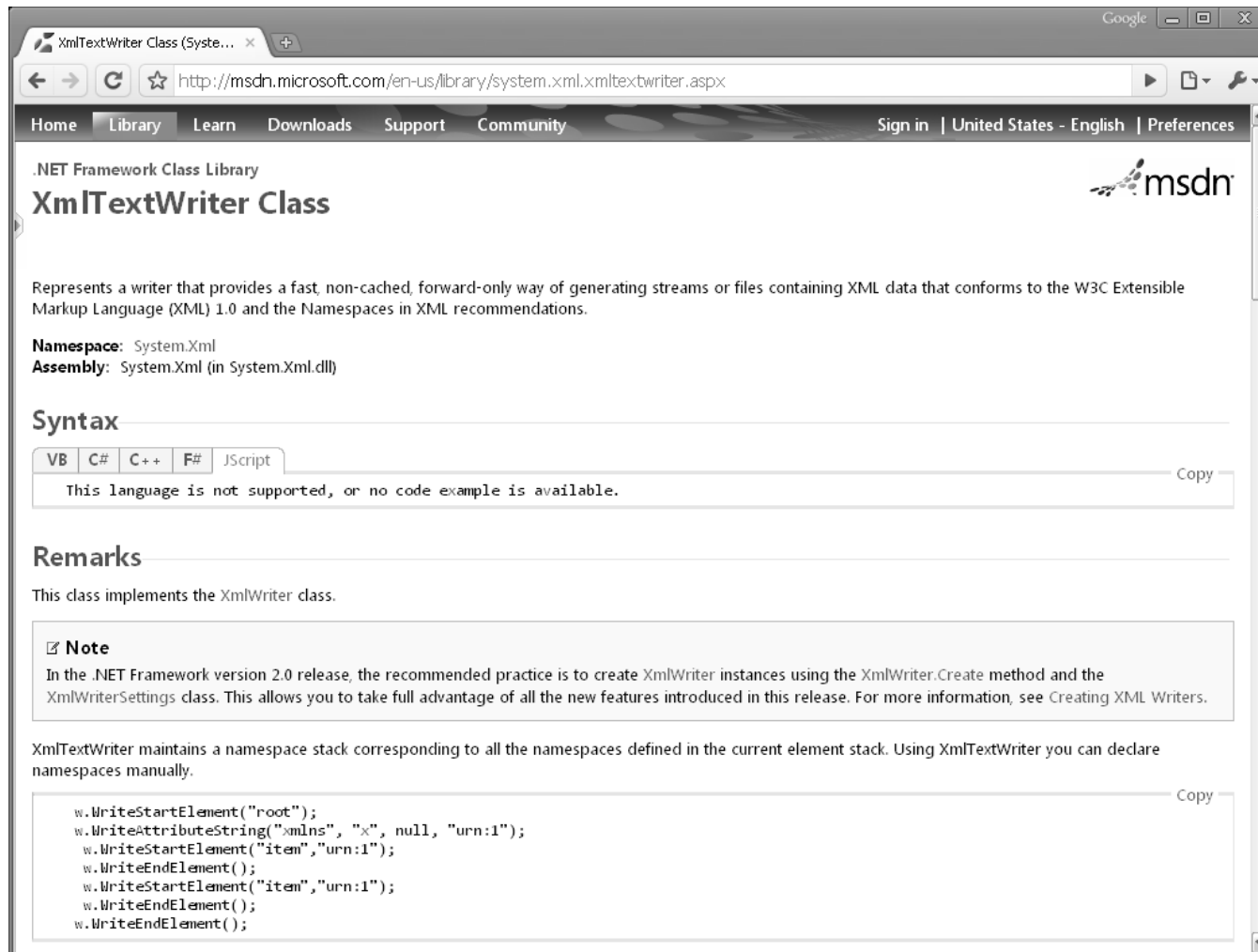
# Background

As of today, there are three known solutions to call a Lawson Web Service from a Personalized Script in Lawson Smart Office: 1) the “Big string”, 2) the XML writer, and 3) the C# proxy written with Microsoft Visual Studio. Each of these solutions has its advantages and disadvantages.

**Known Solution 1:** The “Big String” solution creates the SOAP Request in a long String or StringBuilder in JScript.NET, and sends it to the SOAP server with XmlHttpRequest or WebRequest. This solution is easy to implement because it doesn’t require any special skills nor tool, but it doesn’t provide SOAP validation, it also requires handling the Request/Response, and it’s hard to maintain.

```
xml += '<?xml version="1.0" encoding="utf-8"?>';
xml += '<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"';
xml += ' xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"';
xml += ' xmlns:xsd="http://www.w3.org/2001/XMLSchema">';
xml += '<soap:Header>';
xml += '<mws xmlns="http://mws.intentia.net/mws2">';
xml += '<user>' + user + '</user>';
xml += '<password>' + password + '</password>';
xml += '</mws>';
xml += '</soap:Header>';
xml += '<soap:Body>';
xml += '<List xmlns="http://sales.lawson.com/MMSO60MI/List">';
xml += '<ListItem>';
xml += '<Warehouse>' + warehouse + '</Warehouse>';
xml += '<ItemNumber>' + itemNumber + '</ItemNumber>';
xml += '</ListItem>';
xml += '</List>';
xml += '</soap:Body>';
xml += '</soap:Envelope>';
```

**Known Solution 2:** The XML writer solution uses any XML writer class such as `System.Xml.XmlTextWriter` to create the SOAP Request. I have not tested this solution myself. It's similar to the "Big String" solution, in better because it adds XML validation, but it requires more skills in the programming.



The screenshot shows a web browser window displaying the MSDN page for the `XmlTextWriter` class. The browser's address bar shows the URL `http://msdn.microsoft.com/en-us/library/system.xml.xmltextwriter.aspx`. The page has a navigation bar with links for Home, Library, Learn, Downloads, Support, and Community. The main heading is ".NET Framework Class Library" followed by "XmlTextWriter Class". A description states: "Represents a writer that provides a fast, non-cached, forward-only way of generating streams or files containing XML data that conforms to the W3C Extensible Markup Language (XML) 1.0 and the Namespaces in XML recommendations." Below this, it specifies the namespace as `System.Xml` and the assembly as `System.Xml (in System.Xml.dll)`. The "Syntax" section has tabs for VB, C#, C++, F#, and JScript, with a message stating "This language is not supported, or no code example is available." The "Remarks" section notes that this class implements the `XmlWriter` class. A "Note" box mentions that in the .NET Framework version 2.0 release, the recommended practice is to create `XmlWriter` instances using the `XmlWriter.Create` method and the `XmlWriterSettings` class. At the bottom, a code block shows an example of using `XmlTextWriter` to write XML elements.

XmlTextWriter Class (System... x)

http://msdn.microsoft.com/en-us/library/system.xml.xmltextwriter.aspx

Home Library Learn Downloads Support Community Sign in | United States - English | Preferences

.NET Framework Class Library

## XmlTextWriter Class

Represents a writer that provides a fast, non-cached, forward-only way of generating streams or files containing XML data that conforms to the W3C Extensible Markup Language (XML) 1.0 and the Namespaces in XML recommendations.

**Namespace:** System.Xml  
**Assembly:** System.Xml (in System.Xml.dll)

### Syntax

VB C# C++ F# JScript

This language is not supported, or no code example is available.

### Remarks

This class implements the `XmlWriter` class.

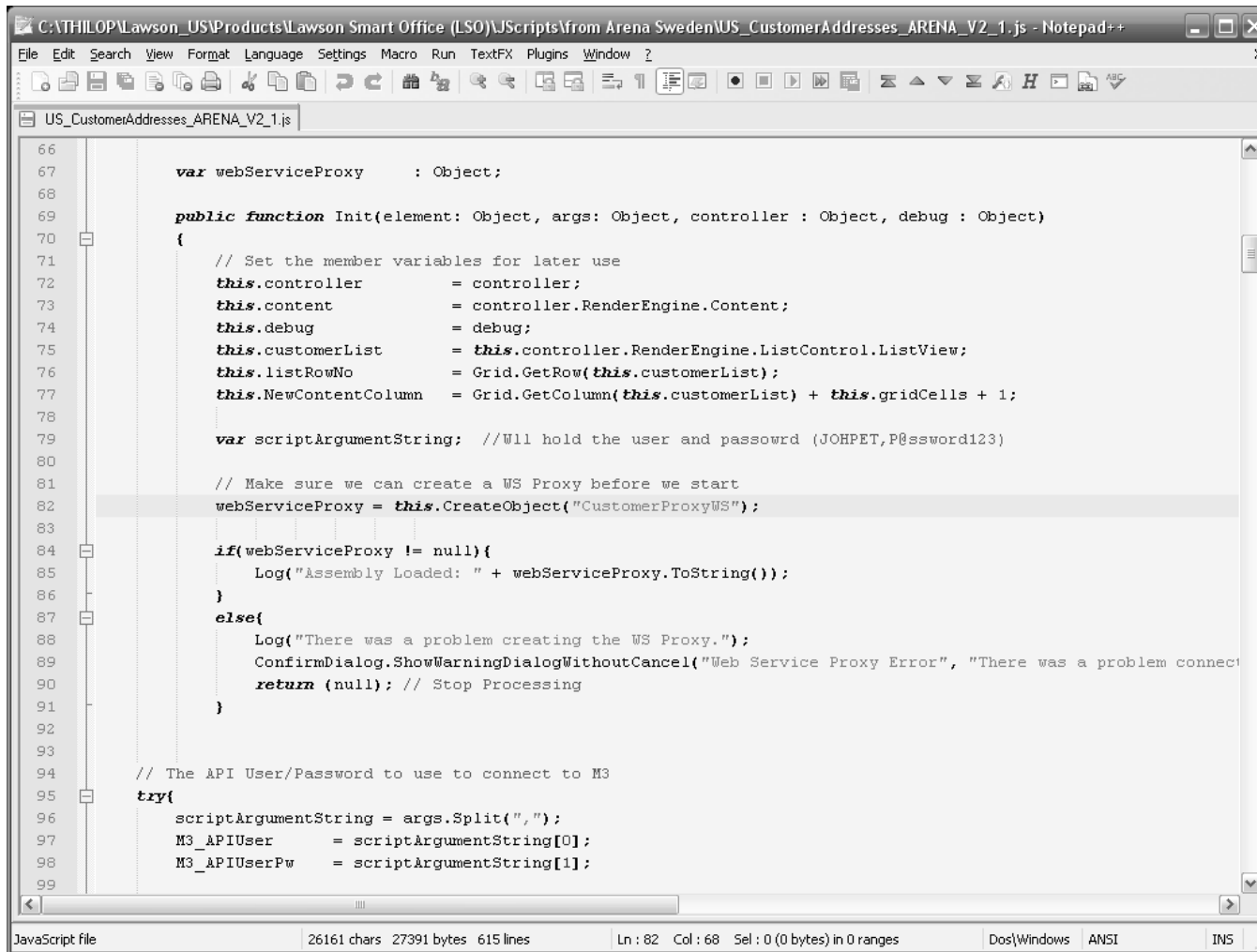
**Note**

In the .NET Framework version 2.0 release, the recommended practice is to create `XmlWriter` instances using the `XmlWriter.Create` method and the `XmlWriterSettings` class. This allows you to take full advantage of all the new features introduced in this release. For more information, see [Creating XML Writers](#).

`XmlTextWriter` maintains a namespace stack corresponding to all the namespaces defined in the current element stack. Using `XmlTextWriter` you can declare namespaces manually.

```
w.WriteStartElement("root");
w.WriteAttributeString("xmlns", "x", null, "urn:1");
w.WriteStartElement("item", "urn:1");
w.WriteEndElement();
w.WriteStartElement("item", "urn:1");
w.WriteEndElement();
w.WriteEndElement();
```

**Known Solution 3:** Peter A Johansson found a great solution by creating a C# proxy in Microsoft Visual Studio and generating a DLL that hides all the complexity. (Also, Peter suggests storing the DLL in the instance cache to minimize network traffic.) Visual Studio is easy to learn and generates most of the coding, but this solution requires C# programming skills, and Visual Studio skills.



The screenshot shows a Notepad++ window with the title bar "C:\ATHIOP\Lawson\_US\Products\Lawson Smart Office (LSO)\JScripts\from Arena Sweden\US\_CustomerAddresses\_ARENA\_V2\_1.js - Notepad++". The menu bar includes File, Edit, Search, View, Format, Language, Settings, Macro, Run, TextFX, Plugins, Window, and Help. The toolbar contains various icons for file operations, editing, and running. The editor window displays the file "US\_CustomerAddresses\_ARENA\_V2\_1.js" with the following JavaScript code:

```
66
67     var webServiceProxy      : Object;
68
69     public function Init(element: Object, args: Object, controller : Object, debug : Object)
70     {
71         // Set the member variables for later use
72         this.controller        = controller;
73         this.content            = controller.RenderEngine.Content;
74         this.debug              = debug;
75         this.customerList       = this.controller.RenderEngine.ListControl.ListView;
76         this.listRowNo          = Grid.GetRow(this.customerList);
77         this.NewContentColumn    = Grid.GetColumn(this.customerList) + this.gridCells + 1;
78
79         var scriptArgumentString; //Will hold the user and password (JOHPET,P@ssword123)
80
81         // Make sure we can create a WS Proxy before we start
82         webServiceProxy = this.CreateObject("CustomerProxyWS");
83
84         if(webServiceProxy != null){
85             Log("Assembly Loaded: " + webServiceProxy.ToString());
86         }
87         else{
88             Log("There was a problem creating the WS Proxy.");
89             ConfirmDialog.ShowWarningDialogWithoutCancel("Web Service Proxy Error", "There was a problem connect
90             return (null); // Stop Processing
91         }
92
93
94     // The API User/Password to use to connect to M3
95     try{
96         scriptArgumentString = args.Split(",");
97         M3_APIUser           = scriptArgumentString[0];
98         M3_APIUserPw         = scriptArgumentString[1];
99     }
```

The status bar at the bottom indicates "JavaScript file", "26161 chars 27391 bytes 615 lines", "Ln : 82 Col : 68 Sel : 0 (0 bytes) in 0 ranges", "Dos/Windows", "ANSI", and "INS".

Here is an excerpt of the discussions from the Lawson Web Services Forum in the Community Of Interest (COI) illustrating that there is still no official solution to call a Lawson Web Service from a Personalized Script in Lawson Smart Office.

<http://fsops.lawson.com/sites/fsadm/field/Solutions/Knowledge/COI/Meintegration/Lists/General%20Discussion10/DispForm.aspx?ID=1&Source=http%3A%2F%2Ffsops%2Elawson%2Ecom%2Fsites%2Ffsadm%2Ffield%2FSolutions%2FKnowledge%2FCOI%2FMeintegration%2Fpages%2FLWS%2Easpx>

Integration (M3) - Microsoft Internet Explorer provided by Lawson

http://fsops.lawson.com/sites/fsadm/field/Solutions/Knowledge/COI/Meintegration/pages/LWS.aspx

Integration (M3)

Home Help Up to Communities of Interest (COI)

Integration (M3)

Home M3 EDI and BMs MEC M3 ACO LWS Useful Links Re-usable Content

Welcome to the Lawson Web Services discussion forum Lawson Web Services

- From: Bernd Herrmann  
Posted At: 4/1/2009 9:31 AM  
Subject: Run an LWS from a .net-Application (e.g. JScript in Smart Office)

Hi,  
I think that LWS is a very powerful tool. In combination with a script that was developed in Smart Office it would be most useful. So when you would be able to run a web service directly from a panel in Smart Office where you could pass on some values would be great. I managed to run a standard web service already but I am struggling how to run a web service that runs an M3 API. Does someone has an example maybe how to run an LWS from JScript .net? This would be exciting...

Regards,  
Bernd

- From: Thibaud Lopez Schneider  
Posted At: 4/6/2009 7:13 PM  
Subject: Run an LWS from a .net-Application (e.g. JScript in Smart Office)

I think several colleagues have succeeded to call standard web services from Lawson Smart Client, but all have failed to call Lawson Web Services. There seems to be an incompatibility between .NET and LWS.  
The very ugly workaround is to hard-code the SOAP Request as a mega-string and send it using an HTTP Request.

- From: Johan Lofgren  
Posted At: 4/7/2009 5:33 AM  
Subject: Run an LWS from a .net-Application (e.g. JScript in Smart Office)

There are no incompatibilities between .net and LWS.

The problem is more of the nature "how to generate .net proxy classes and make them available on the smart office server". LPD is looking into creating an example.

Local intranet 100%

The fourth solution



**Step 1.** Install the free Microsoft .NET SDK. In the SDK, we're particularly interested in the Web Services Description Language Tool (wsdl.exe) which is a proxy class generator for the command line.

<http://www.microsoft.com/downloads/details.aspx?FamilyID=fe6f2099-b7b4-4f47-a244-c96d69c35dec&displaylang=en>

The screenshot shows the Microsoft Download Center page for the .NET Framework 2.0 Software Development Kit (SDK) (x86). The page is displayed in a web browser window with the address bar showing the URL: <http://www.microsoft.com/downloads/details.aspx?FamilyID=fe6f2099-b7b4-4f47-a244-c96d69c35dec&displaylang=en>. The page features a Microsoft logo, a search bar, and a navigation sidebar on the left. The main content area displays the product name, a brief description, and a table of quick details. A download button is prominently displayed.

**Download details: .NET Fra...**

Click Here to Install Silverlight

United States | Change | All Microsoft Sites

Microsoft

Search Microsoft.com

bing Web

Download Center

Download Center Home

Search All Downloads Go Advanced Search

**Product Families**

- Windows
- Office
- Servers
- Business Solutions
- Developer Tools
- Windows Live
- MSN
- Games & Xbox
- Windows Mobile
- All Downloads

**Download Categories**

- Games
- DirectX
- Internet
- Windows Security & Updates
- Windows Media
- Drivers
- Home & Office
- Mobile Devices
- Mac & Other Platforms
- System Tools
- Development Resources

**Download Resources**

- Microsoft Update Services
- Download Center FAQ
- Related Sites

**Download Notifications**

- Notifications Signup

**Worldwide Downloads**

**.NET Framework 2.0 Software Development Kit (SDK) (x86)**

**Brief Description**

The Microsoft .NET Framework Software Development Kit (SDK) version 2.0 includes tools, documentation and samples developers need to write, build, test, and deploy .NET Framework applications on x86 platforms.

**On This Page**

- Quick Details
- System Requirements
- Related Resources
- Related Downloads
- Overview
- Instructions
- What Others Are Downloading

**Download**

**Quick Details**

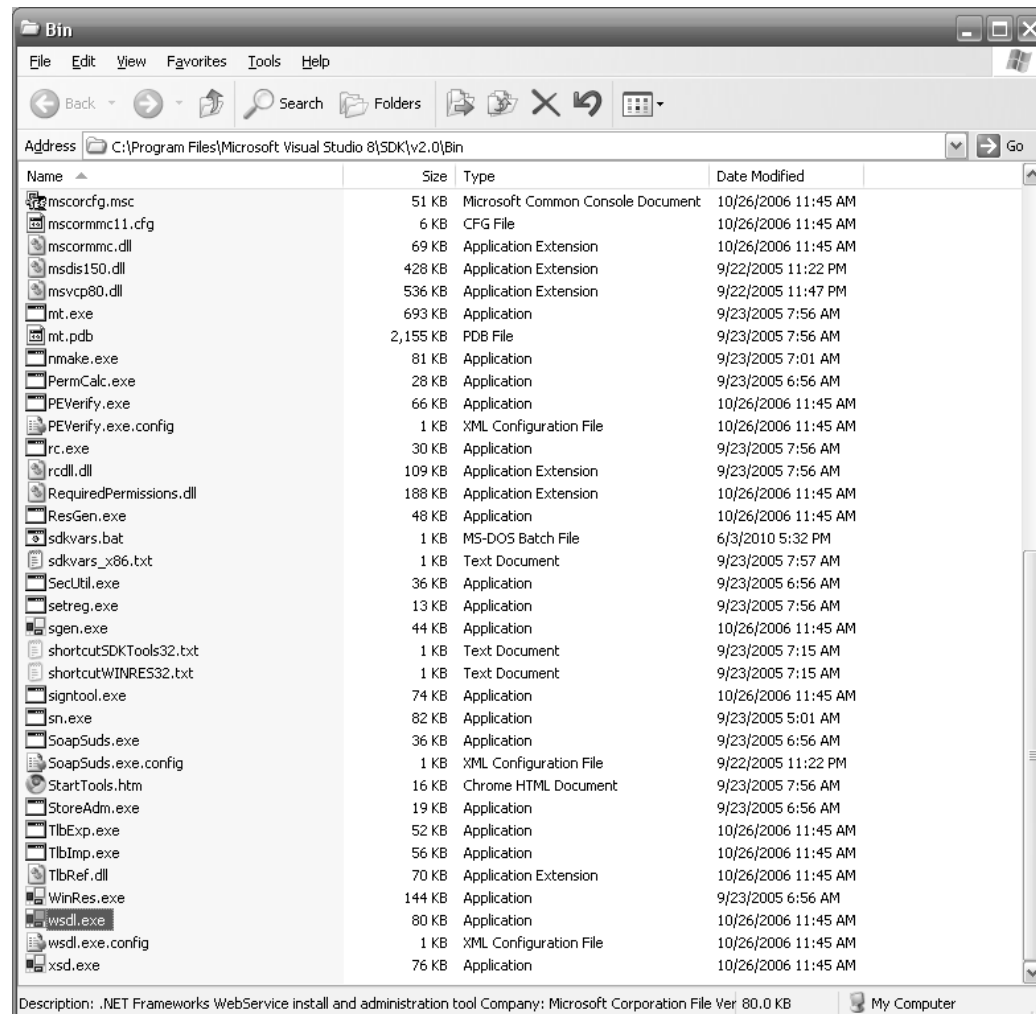
File Name:	setup.exe
Version:	1
Date Published:	11/29/2006
Language:	English
Download Size:	354.0 MB
Estimated Download Time:	14 hr 24 min 56K

**Change Language:** English Change

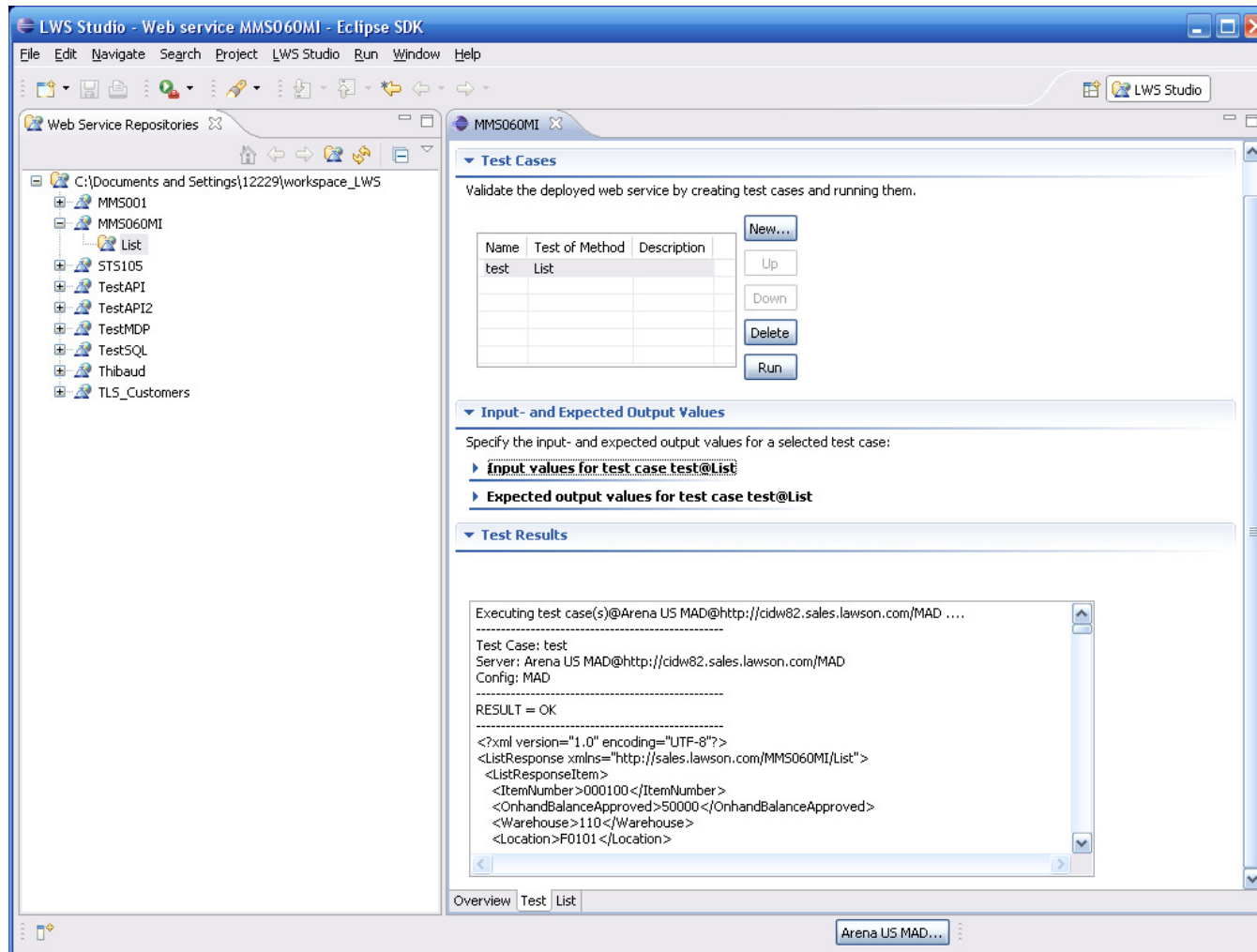
**Overview**

Verify that the SDK correctly installed the Web Services Description Language Tool (wsdl.exe).

C:\Program Files\Microsoft Visual Studio 8\SDK\v2.0\Bin\



**Step 2.** Test your web service with Lawson Web Services Studio.  
Make sure that it works and returns RESULT = OK.

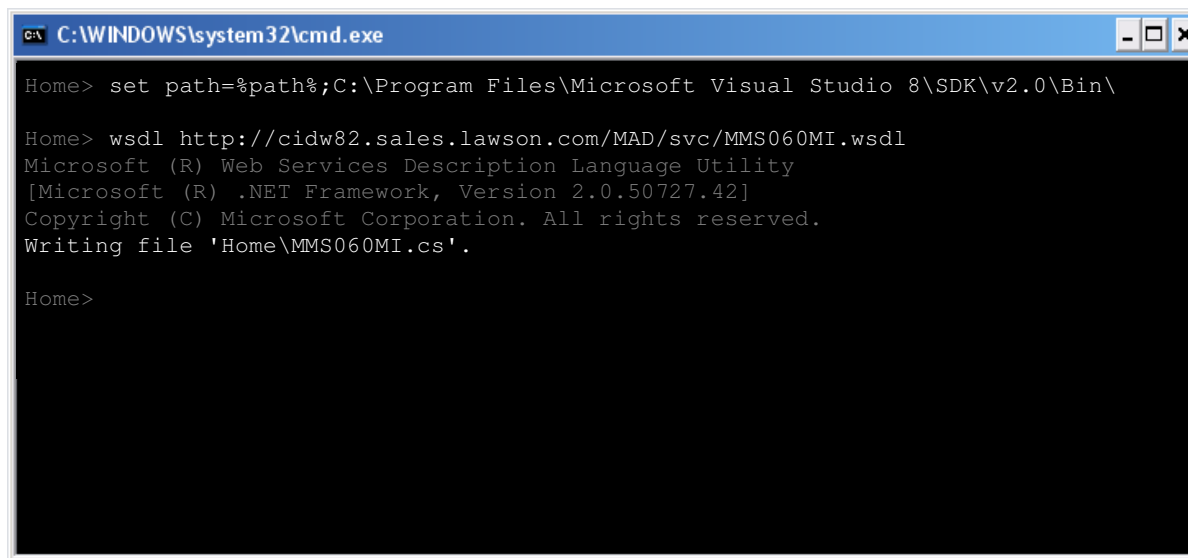


Copy the URL of your web service's WSDL. (For that, open the Lawson Web Services Administration page, click on *List Web Services*, locate your web service, click on the *wsdl* link on its right, and copy that URL to the clipboard.)  
For example: <http://cidw82.sales.lawson.com/MAD/svc/MMS060MI.wsdl>

The image shows two overlapping browser windows. The background window is the 'Lawson Web Services - Microsoft Internet Explorer' page. It has a sidebar with the following links: 'List Web Services' (with a sub-link 'View a list of deployed web services'), 'View Log' (with a sub-link 'View the server log'), 'Download Server Certificate' (with a sub-link 'Download the Server certificate. This certifies web service calls'), 'Deploy Services' (with a sub-link 'Deploy web services. You can only deploy a certificate that has been uploaded to the server'), 'Server Setup' (with a sub-link 'Check the server environment'), 'Undeploy Services' (with a sub-link 'Undeploy web services'), and 'Manage Configurations' (with a sub-link 'Create and edit service configurations for...'). The foreground window is a Microsoft Internet Explorer browser displaying the WSDL file at the URL <http://cidw82.sales.lawson.com/MAD/svc/MMS060MI.wsdl>. The WSDL content is as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:mws="http://mws.intentia.net/mws2" xmlns:tns="http://sales.lawson.com/MMS060MI"
  xmlns:list="http://sales.lawson.com/MMS060MI/List" name="MMS060MI"
  targetNamespace="http://sales.lawson.com/MMS060MI">
- <wsdl:documentation>
  <id>8A80808A-0128-0000-28B3-AEA5B3B76F5B</id>
  <version>1.0</version>
</wsdl:documentation>
- <wsdl:types>
- <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://sales.lawson.com/MMS060MI/List" elementFormDefault="qualified">
- <xsd:annotation>
  <xsd:documentation>Api: Detailed item balances interface Transaction: List</xsd:documentation>
</xsd:annotation>
  <xsd:element name="List" type="list:ListCollection" />
- <xsd:complexType name="ListCollection">
- <xsd:sequence>
  <xsd:element name="ListItem" minOccurs="0" maxOccurs="unbounded" type="list:ListItem" />
</xsd:sequence>
  <xsd:attribute name="maxRecords" use="optional" default="100" type="xsd:int" />
</xsd:complexType>
- <xsd:complexType name="ListItem">
- <xsd:sequence>
- <xsd:element name="Company" type="xsd:decimal" nillable="true" maxOccurs="1" minOccurs="0">
- <xsd:annotation>
  <xsd:documentation>Company (CONO)</xsd:documentation>
</xsd:annotation>
</xsd:element>
- <xsd:element name="Warehouse" nillable="false">
- <xsd:annotation>
  <xsd:documentation>Warehouse (WHLO)</xsd:documentation>
</xsd:annotation>
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:maxLength value="3" />
</xsd:restriction>
</xsd:simpleType>
</xsd:sequence>
</xsd:complexType>
```

**Step 3.** Generate a C# proxy class for your web service with the following command: `wsdl MyWSDL`.



```
C:\WINDOWS\system32\cmd.exe

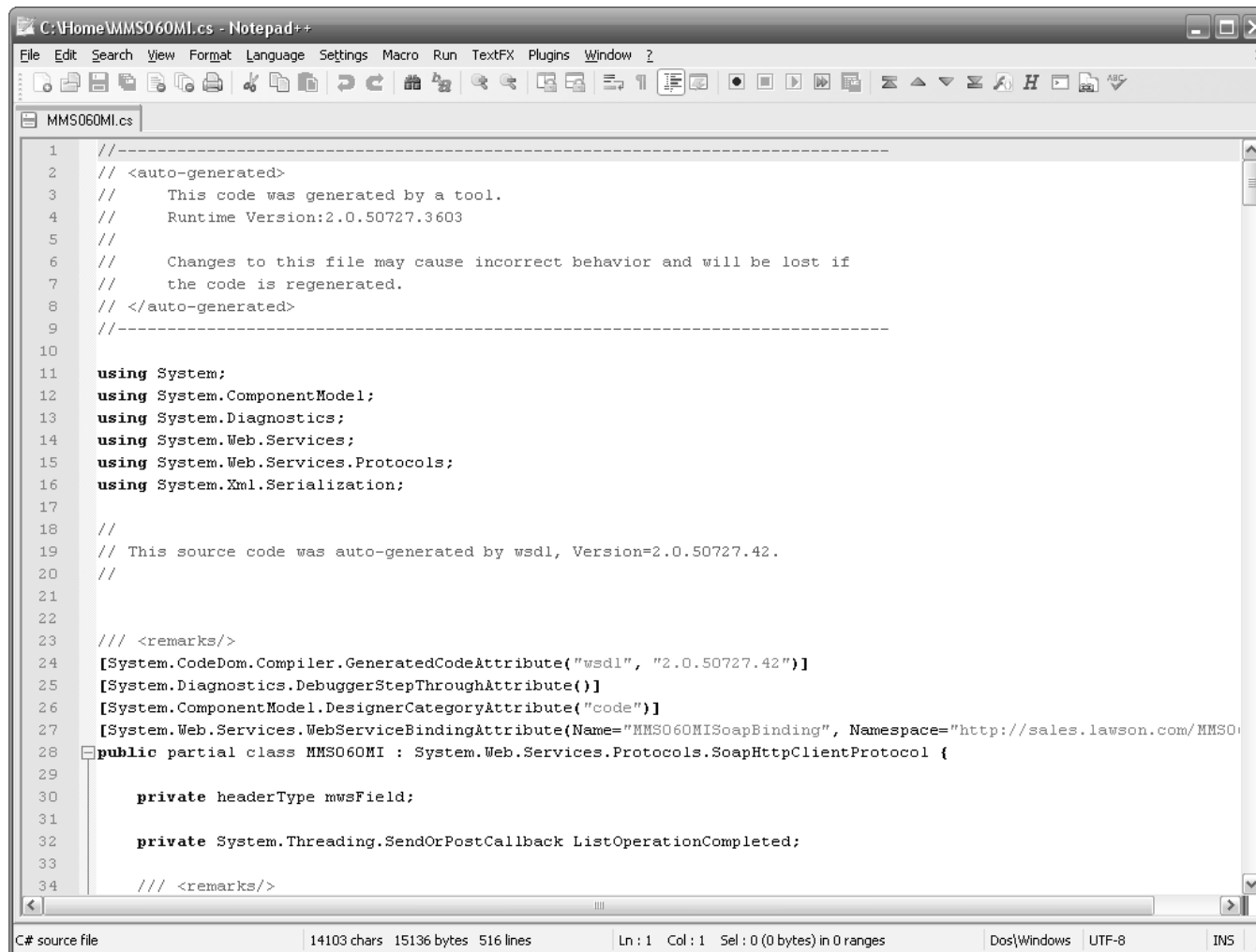
Home> set path=%path%;C:\Program Files\Microsoft Visual Studio 8\SDK\v2.0\Bin\

Home> wsdl http://cidw82.sales.lawson.com/MAD/svc/MMS060MI.wsdl
Microsoft (R) Web Services Description Language Utility
[Microsoft (R) .NET Framework, Version 2.0.50727.42]
Copyright (C) Microsoft Corporation. All rights reserved.
Writing file 'Home\MMS060MI.cs'.

Home>
```

Note 1: You can also try to generate a proxy in JScript.NET, but I tried the three types of Lawson Web Services (API, MDP, and SQL) and I always got the error *“Custom attributes on parameter declarations are not supported by JScriptCodeProvider.”* Note 2: You can also save the WSDL as a file in your computer and use a filepath instead of a URL.

Verify that the wsdl.exe tool correctly generated the C# proxy class.



```
1 //-----
2 // <auto-generated>
3 //   This code was generated by a tool.
4 //   Runtime Version:2.0.50727.3603
5 //
6 //   Changes to this file may cause incorrect behavior and will be lost if
7 //   the code is regenerated.
8 // </auto-generated>
9 //-----
10
11 using System;
12 using System.ComponentModel;
13 using System.Diagnostics;
14 using System.Web.Services;
15 using System.Web.Services.Protocols;
16 using System.Xml.Serialization;
17
18 //
19 // This source code was auto-generated by wsdl, Version=2.0.50727.42.
20 //
21
22
23 /// <remarks/>
24 [System.CodeDom.Compiler.GeneratedCodeAttribute("wsdl", "2.0.50727.42")]
25 [System.Diagnostics.DebuggerStepThroughAttribute()]
26 [System.ComponentModel.DesignerCategoryAttribute("code")]
27 [System.Web.Services.WebServiceBindingAttribute(Name="MMSO60MISoapBinding", Namespace="http://sales.lawson.com/MMSO60MI")]
28 public partial class MMSO60MI : System.Web.Services.Protocols.SoapHttpClientProtocol {
29
30     private headerType mwsField;
31
32     private System.Threading.SendOrPostCallback ListOperationCompleted;
33
34     /// <remarks/>
35 }
```

C# source file 14103 chars 15136 bytes 516 lines Ln : 1 Col : 1 Sel : 0 (0 bytes) in 0 ranges Dos\Windows UTF-8 INS

For more information on the wsdl.exe tool, refer to its documentation.

[http://msdn.microsoft.com/en-us/library/7h3ystb6\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/7h3ystb6(VS.80).aspx)

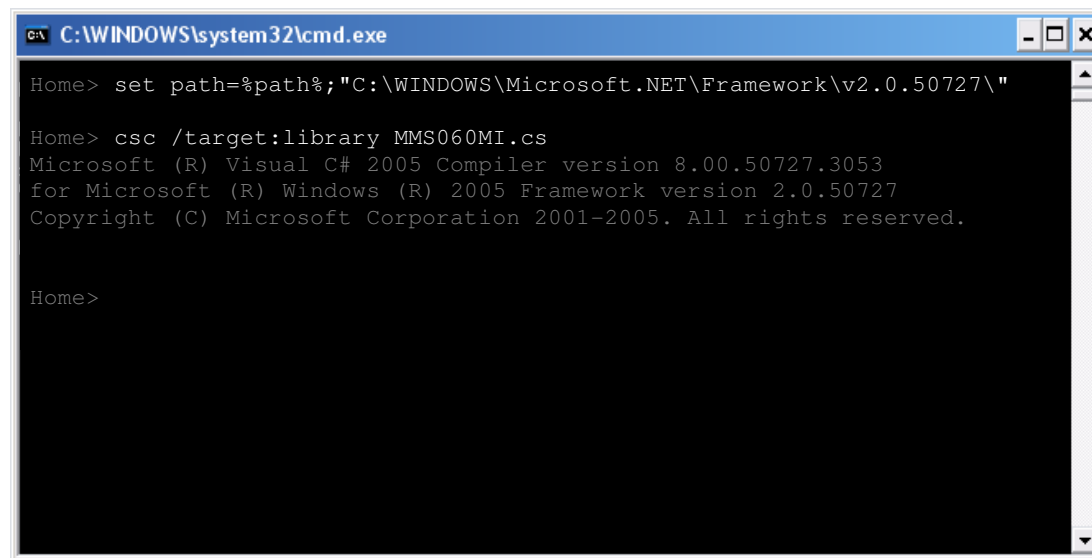
The screenshot shows a web browser window with the address bar displaying `http://msdn.microsoft.com/en-us/library/7h3ystb6(VS.80).aspx`. The page title is ".NET Framework Tools Web Services Description Language Tool (Wsd.exe)". The main content area describes the tool's function: "The Web Services Description Language tool generates code for XML Web services and XML Web service clients from WSDL contract files, XSD schemas, and .discomap discovery documents." Below this is a command line example: `wsdl [options] {URL | path}` with a "Copy" button. Two tables follow, detailing arguments and options.

Argument	Description
<i>URL</i>	The URL to a WSDL contract file (.wsdl), XSD schema file (.xsd), or discovery document (.disco). Note that you cannot specify a URL to a .discomap discovery document.
<i>Path</i>	The path to a local WSDL contract file (.wsdl), XSD schema file (.xsd), or discovery document (.disco or .discomap).

Option	Description
<code>/appsettingurlkey:key</code> or <code>/urlkey:key</code>	Specifies the configuration key to use in order to read the default value for the URL property when generating code. When using the <code>/parameters</code> option, this value is the <code>&lt;appSettingUrlKey&gt;</code> element and contains a string.
<code>/appsettingbaseurl:baseurl</code> or <code>/baseurl:baseurl</code>	Specifies the base URL to use when calculating the URL fragment. The tool calculates the URL fragment by converting the relative URL from the <code>baseurl</code> argument to the URL in the WSDL document. You must specify the <code>/appsettingurlkey</code> option with this option. When using the <code>/parameters</code> option, this value is the <code>&lt;appSettingBaseUrl&gt;</code> element and contains a string.

**Step 4.** Compile the C# proxy class with the following command line: `csc /target:library MyProxy.cs`



```
C:\WINDOWS\system32\cmd.exe

Home> set path=%path%;"C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\"

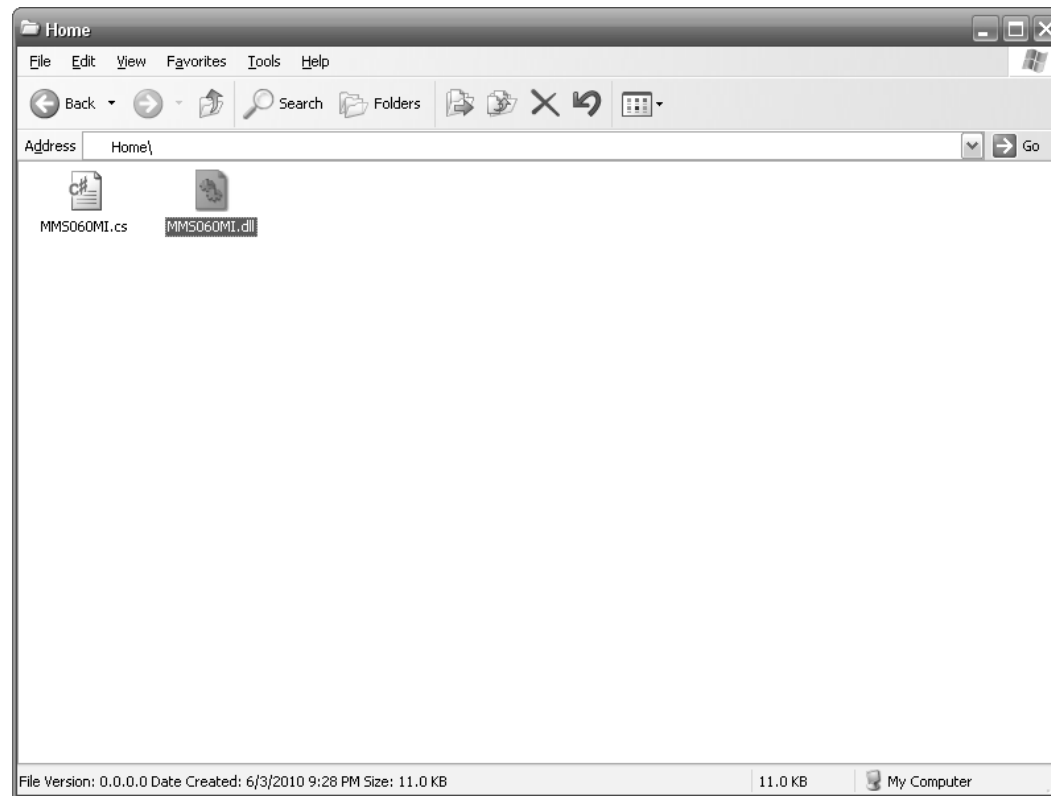
Home> csc /target:library MMS060MI.cs
Microsoft (R) Visual C# 2005 Compiler version 8.00.50727.3053
for Microsoft (R) Windows (R) 2005 Framework version 2.0.50727
Copyright (C) Microsoft Corporation 2001-2005. All rights reserved.

Home>
```

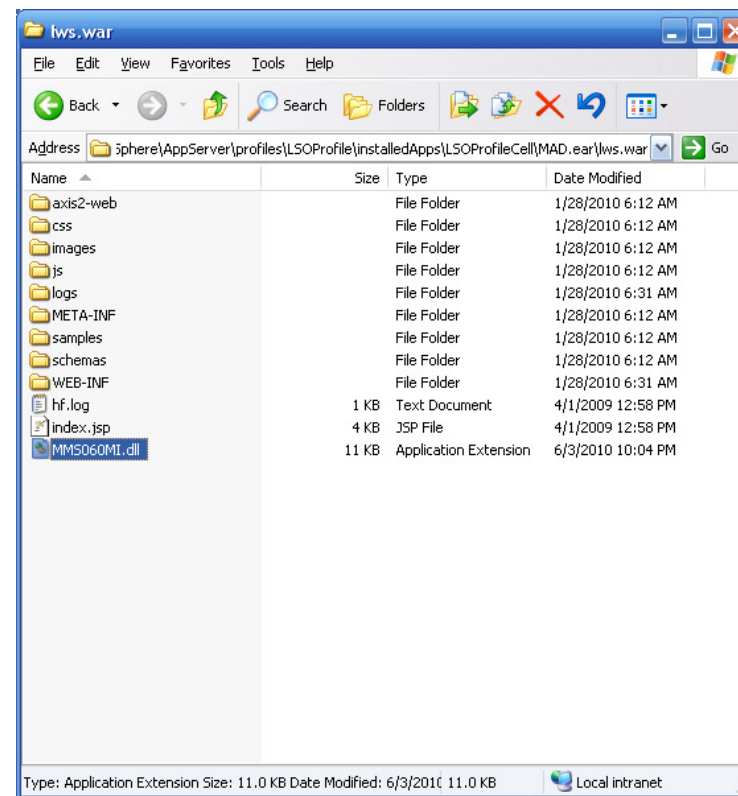
Note 1: The C# compiler (csc.exe) is provided with the .NET Framework so you should have already had it before installing the SDK. Note 2: I'm using the .NET Framework 2.0.x because that's the same one used by the wsdl.exe tool that generated the proxy class, but it also compiled with the .NET Framework 4.0.x.



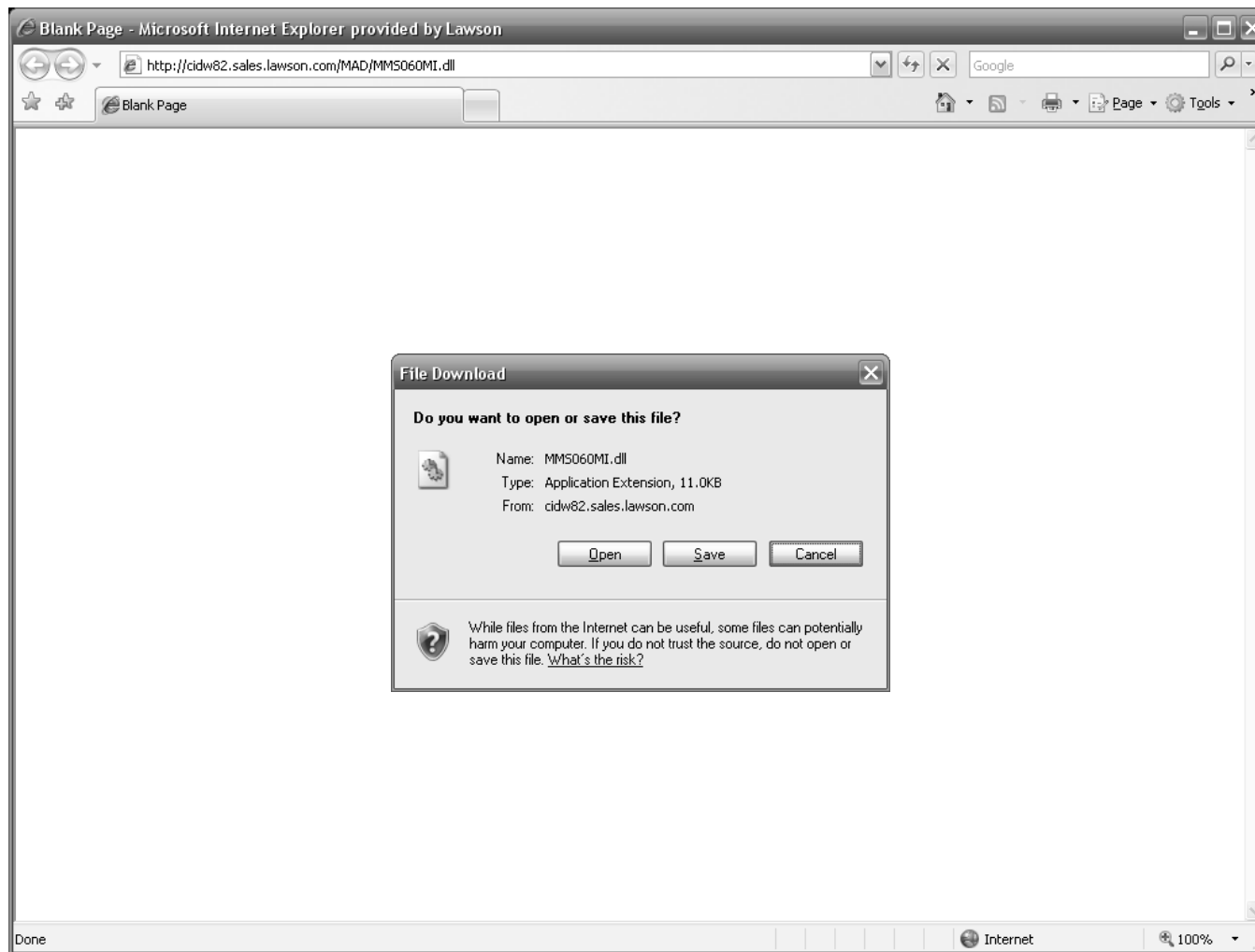
Verify that the compiler correctly  
created the DLL.



**Step 5.** Publish the DLL into a location that is accessible to your users, for example in the lws.war folder in WebSphere.



Verify that you have access to the DLL as a user.



Note: You can use a URL like in the screenshot, or shared folder, or a UNC path like \\host\c\$\path\file.dll, etc.

## Step 6. Create a Personalized Script in JScript.NET.

```
import System;
import System.Reflection;

package MForms.JScript {
    class TestMyWebService {
        public function Init(element: Object, args: Object, controller : Object, debug : Object) {

            // load the DLL
            var assembly = Assembly.LoadFrom("http://cidw82.sales.lawson.com/MAD/MMS060MI.dll");

            // create a proxy
            var proxy = assembly.CreateInstance("MMS060MI");

            // SOAP Request Header (user and password)
            var header = assembly.CreateInstance("headerType");
            header.user = "secret";
            header.password = "secret";
            proxy.mws = header;

            // SOAP Request Body (input parameters)
            var body = assembly.CreateInstance("ListItem");
            body.Company = new Decimal(701);
            body.Warehouse = "110";

            var collection = assembly.CreateInstance("ListCollection");
            collection.maxRecords = 100;
            var ListItem = body.GetType();
            collection.ListItem = new ListItem[1];
            collection.ListItem[0] = body;

            // call the web service
            var response = proxy.List(collection);

            // SOAP Response
            debug.WriteLine(response.length + " records returned");
            for (var i in response) {
                debug.WriteLine(response[i].Location + "=" + response[i].StatusBalanceID);
            }
        }
    }
}
```

```

// create a proxy
var proxy = assembly.CreateInstance("MMS060MI");

// SOAP Request Header (user and password)
var header = assembly.CreateInstance("headerType");
header.user = "secret";
header.password = "secret";
proxy.mws = header;

// SOAP Request Body (input parameters)
var body = assembly.CreateInstance("ListItem");
body.Company = new Decimal(701);
body.Warehouse = "110";

var collection = assembly.CreateInstance("ListCollection");
collection.maxRecords = 100;
var ListItem = body.GetType();
collection.ListItem = new ListItem[1];
collection.ListItem[0] = body;

// call the web service
var response = proxy.List(collection);

// SOAP Response
debug.WriteLine(response.length + " records returned");
for (var i in response) {
    debug.WriteLine(
        response[i].Location + "=" +
        response[i].StatusBalanceID);
}

```

Web Service name

Method name

LWS Studio - Web service MMS060MI - Eclipse SDK

File Edit Navigate Search Project LWS Studio Run Window Help

Web Service Rep

C:\Documents and Settings\12\... MMS001 MMS060MI List STS105 TestAPI TestAPI2 TestMDP TestSQL Thibaud TLS\_Customers

### Web Service Method List wrapping an M3 API Program T

#### General Information

General information about this method.

Name:	List	Short Description:	Detail
M3 API Program:	MMS060MI	Long Description:	Detail
Wrapped Transaction:	List		
Alias:	List		

#### Input to List

Specify aliases for the input fields of this method.

Filter Field on

Field	Alias	Type	Length	Offset	Constraint
BANO	LotNumber	alphanumeric	20	46	Optional
CAMU	Container	alphanumeric	20	66	Optional
CONO	Company	numeric	3	15	Optional
ITNO	ItemNumber	alphanumeric	15	21	Optional
REPN	ReceivingNumber	numeric	10	86	Optional
WHLO	Warehouse	alphanumeric	3	18	Mandatory
WHSL	Location	alphanumeric	10	36	Optional

#### Output from List

Specify aliases for the output fields of this method.

Available Outputs:

Filter Field on

Field	Alias
ABCD	ABCClassVolume
ALOC	Allocatable
ALQT	AllocatedQuantityBasicUM
ATNB	AttributeNumberLot
ATNR	AttributeNumber
AUDD	AutomaticDeletionDelay
AUDE	AutomaticDeletion
BRE2	LotReference2
BREF	LotReference1
BREM	Remark

Selected Outputs:

Filter Field on

Field	Alias
BANO	LotNumber
CAMU	Container
CONO	Company
ITNO	ItemNumber
REPN	ReceivingNumber
STAS	StatusBalanceID
STAT	Status
STQT	OnhandBalanceApproved
WHLO	Warehouse
WHSL	Location

Overview Test List

Arena US MAD...

```

// create a proxy
var proxy = assembly.CreateInstance("MMS060MI");

// SOAP Request Header (user and password)
var header = assembly.CreateInstance("headerType");
header.user = "secret";
header.password = "secret";
proxy.mws = header;

// SOAP Request Body (input parameters)
var body = assembly.CreateInstance("ListItem");
body.Company = new Decimal(701);
body.Warehouse = "110";

var collection = assembly.CreateInstance("ListCollection");
collection.maxRecords = 100;
var ListItem = body.GetType();
collection.ListItem = new ListItem[1];
collection.ListItem[0] = body;

// call the web service
var response = proxy.List(collection);

// SOAP Response
debug.WriteLine(response.length + " records returned");
for (var i in response) {
    debug.WriteLine(
        response[i].Location + "=" +
        response[i].StatusBalanceID);
}

```

Method name + Item suffix

LWS Studio - Web service MMS060MI - Eclipse SDK

File Edit Navigate Search Project LWS Studio Run Window Help

Web Service Rep

C:\Documents and Settings\12\My Documents\...

MMS001

MMS060MI

List

STS105

TestAPI

TestAPI2

TestMDP

TestSQL

Thibaud

TLS\_Customers

Web Service Method List wrapping an M3 API Program T

General Information

General information about this method.

Name: List Short Description: Detail

M3 API Program: MMS060MI Long Description: Detail

Wrapped Transaction: List

Alias: List

Input to List

Specify aliases for the input fields of this method.

Filter Field on

Field	Alias	Type	Length	Offset	Constraint
BANO	LotNumber	alphanumeric	20	46	Optional
CAMU	Container	alphanumeric	20	66	Optional
CONO	Company	numeric	3	15	Optional
ITNO	ItemNumber	alphanumeric	15	21	Optional
REPN	ReceivingNumber	numeric	10	86	Optional
WHLO	Warehouse	alphanumeric	3	18	Mandatory
WHSL	Location	alphanumeric	10	36	Optional

Output from List

Specify aliases for the output fields of this method.

Available Outputs:

Filter Field on

Field	Alias
ABCD	ABCClassVolume
ALOC	Allocatable
ALQT	AllocatedQuantityBasicUM
ATNB	AttributeNumberLot
ATNR	AttributeNumber
AUDD	AutomaticDeletionDelay
AUDE	AutomaticDeletion
BRE2	LotReference2
BREF	LotReference1
BREM	Remark

Selected Outputs:

Filter Field on

Field	Alias
BANO	LotNumber
CAMU	Container
CONO	Company
ITNO	ItemNumber
REPN	ReceivingNumber
STAS	StatusBalanceID
STAT	Status
STQT	OnhandBalanceApproved
WHLO	Warehouse
WHSL	Location

Overview Test List

Arena US MAD...

```

// create a proxy
var proxy = assembly.CreateInstance("MMS060MI");

// SOAP Request Header (user and password)
var header = assembly.CreateInstance("headerType");
header.user = "secret";
header.password = "secret";
proxy.mws = header;

// SOAP Request Body (input parameters)
var body = assembly.CreateInstance("ListItem");
body.Company = new Decimal(701);
body.Warehouse = "110";

var collection = assembly.CreateInstance("ListCollection");
collection.maxRecords = 100;
var ListItem = body.GetType();
collection.ListItem = new ListItem[1];
collection.ListItem[0] = body;

// call the web service
var response = proxy.List(collection);

// SOAP Response
debug.WriteLine(response.length + " records returned");
for (var i in response) {
    debug.WriteLine(
        response[i].Location + "=" +
        response[i].StatusBalanceID);
}

```

Method name  
+ Collection suffix

LWS Studio - Web service MMS060MI - Eclipse SDK

File Edit Navigate Search Project LWS Studio Run Window Help

Web Service Rep

C:\Documents and Settings\12\My Documents\...

MMS001

MMS060MI

List

STS105

TestAPI

TestAPI2

TestMDP

TestSQL

Thibaud

TLS\_Customers

MMS060MI

Web Service Method List wrapping an M3 API Program T

General Information

General information about this method.

Name: List Short Description: Detail

M3 API Program: MMS060MI Long Description: Detail

Wrapped Transaction: List

Alias: List

Input to List

Specify aliases for the input fields of this method.

Filter Field on

Field	Alias	Type	Length	Offset	Constraint
BANO	LotNumber	alphanumeric	20	46	Optional
CAMU	Container	alphanumeric	20	66	Optional
CONO	Company	numeric	3	15	Optional
ITNO	ItemNumber	alphanumeric	15	21	Optional
REPN	ReceivingNumber	numeric	10	86	Optional
WHLO	Warehouse	alphanumeric	3	18	Mandatory
WHSL	Location	alphanumeric	10	36	Optional

Output from List

Specify aliases for the output fields of this method.

Available Outputs:

Filter Field on

Field	Alias
ABCD	ABCClassVolume
ALOC	Allocatable
ALQT	AllocatedQuantityBasicUM
ATNB	AttributeNumberLot
ATNR	AttributeNumber
AUDD	AutomaticDeletionDelay
AUDE	AutomaticDeletion
BRE2	LotReference2
BREF	LotReference1
BREM	Remark

Selected Outputs:

Filter Field on

Field	Alias
BANO	LotNumber
CAMU	Container
CONO	Company
ITNO	ItemNumber
REPN	ReceivingNumber
STAS	StatusBalanceID
STAT	Status
STQT	OnhandBalanceApproved
WHLO	Warehouse
WHSL	Location

Overview Test List

Arena US MAD...

```

// create a proxy
var proxy = assembly.CreateInstance("MMS060MI");

// SOAP Request Header (user and password)
var header = assembly.CreateInstance("headerType");
header.user = "secret";
header.password = "secret";
proxy.mws = header;

// SOAP Request Body (input parameters)
var body = assembly.CreateInstance("ListItem");
body.Company = new Decimal(701);
body.Warehouse = "110";

var collection = assembly.CreateInstance("ListCollection");
collection.maxRecords = 100;
var ListItem = body.GetType();
collection.ListItem = new ListItem[1];
collection.ListItem[0] = body;

// call the web service
var response = proxy.List(collection);

// SOAP Response
debug.WriteLine(response.length + " records returned");
for (var i in response) {
    debug.WriteLine(
        response[i].Location + "=" +
        response[i].StatusBalanceID);
}

```

Input parameters

Output parameters

LWS Studio - Web service MMS060MI - Eclipse SDK

File Edit Navigate Search Project LWS Studio Run Window Help

Web Service Rep

C:\Documents and Settings\12\My Documents\... MMS001 MMS060MI List STS105 TestAPI TestAPI2 TestMDP TestSQL Thibaud TLS\_Customers

### Web Service Method List wrapping an M3 API Program T

#### General Information

General information about this method.

Name:	List	Short Description:	Detail
M3 API Program:	MMS060MI	Long Description:	Detail
Wrapped Transaction:	List		
Alias:	List		

#### Input to List

Specify aliases for the input fields of this method.

Filter Field on

Field	Alias	Type	Length	Offset	Constraint
BANO	LotNumber	alphanumeric	20	46	Optional
CAMU	Container	alphanumeric	20	66	Optional
CONO	Company	numeric	3	15	Optional
ITNO	ItemNumber	alphanumeric	15	21	Optional
REPN	ReceivingNumber	numeric	10	86	Optional
WHLO	Warehouse	alphanumeric	3	18	Mandatory
WHSL	Location	alphanumeric	10	36	Optional

#### Output from List

Specify aliases for the output fields of this method.

Available Outputs:

Filter Field on

Field	Alias
ABCD	ABCClassVolume
ALOC	Allocatable
ALQT	AllocatedQuantityBasicUM
ATNB	AttributeNumberLot
ATNR	AttributeNumber
AUDD	AutomaticDeletionDelay
AUDE	AutomaticDeletion
BRE2	LotReference2
BREF	LotReference1
BREM	Remark

Selected Outputs:

Filter Field on

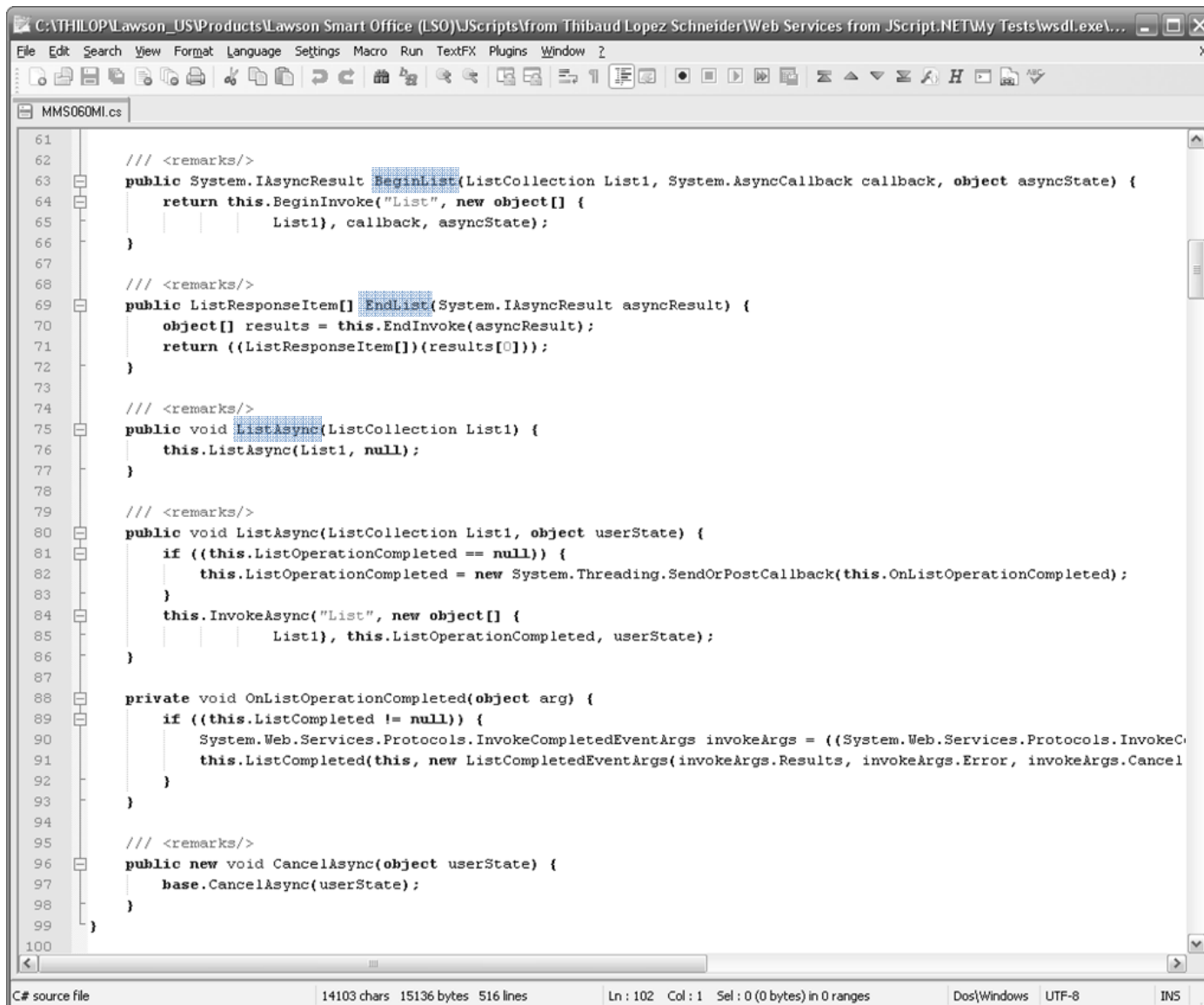
Field	Alias
BANO	LotNumber
CAMU	Container
CONO	Company
ITNO	ItemNumber
REPN	ReceivingNumber
STAS	StatusBalanceID
STAT	Status
STQT	OnhandBalanceApproved
WHLO	Warehouse
WHSL	Location

Overview Test List

Arena US MAD...



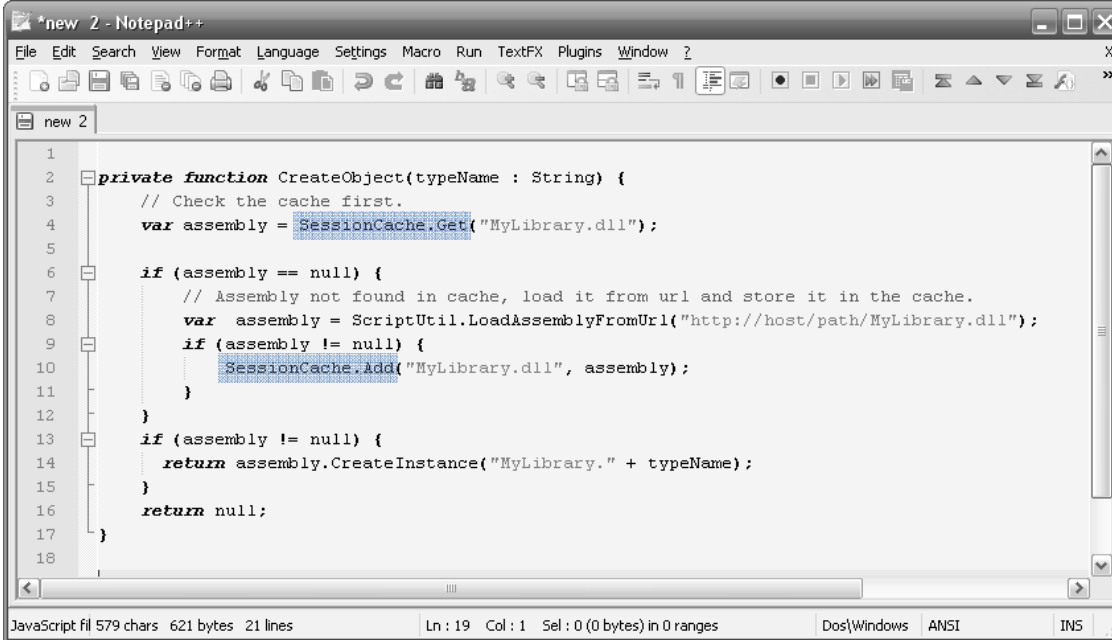
Note 1: the example I wrote makes a synchronous call, meaning Smart Office will freeze until it receives the response. That's bad. Instead, we should use asynchronous calls. The generated proxy class contains the methods ListAsync, BeginList, and EndList. I might provide an example some day.



```
61
62 /// <remarks/>
63 public System.IAsyncResult BeginList(ListCollection List1, System.AsyncCallback callback, object asyncState) {
64     return this.BeginInvoke("List", new object[] {
65         List1, callback, asyncState);
66 }
67
68 /// <remarks/>
69 public ListResponseItem[] EndList(System.IAsyncResult asyncResult) {
70     object[] results = this.EndInvoke(asyncResult);
71     return ((ListResponseItem[]) (results[0]));
72 }
73
74 /// <remarks/>
75 public void ListAsync(ListCollection List1) {
76     this.ListAsync(List1, null);
77 }
78
79 /// <remarks/>
80 public void ListAsync(ListCollection List1, object userState) {
81     if ((this.ListOperationCompleted == null)) {
82         this.ListOperationCompleted = new System.Threading.SendOrPostCallback(this.OnListOperationCompleted);
83     }
84     this.InvokeAsync("List", new object[] {
85         List1, this.ListOperationCompleted, userState);
86 }
87
88 private void OnListOperationCompleted(object arg) {
89     if ((this.ListCompleted != null)) {
90         System.Web.Services.Protocols.InvokeCompletedEventArgs invokeArgs = ((System.Web.Services.Protocols.InvokeCompletedEventArgs) arg);
91         this.ListCompleted(this, new ListCompletedEventArgs(invokeArgs.Results, invokeArgs.Error, invokeArgs.Cancelled));
92     }
93 }
94
95 /// <remarks/>
96 public new void CancelAsync(object userState) {
97     base.CancelAsync(userState);
98 }
99
100 }
```

C# source file 14103 chars 15136 bytes 516 lines Ln : 102 Col : 1 Sel : 0 (0 bytes) in 0 ranges Dos\Windows UTF-8 INS

Note 2: Also, Peter A Johansson recommends storing the DLL in the Session cache so as to minimize network traffic.

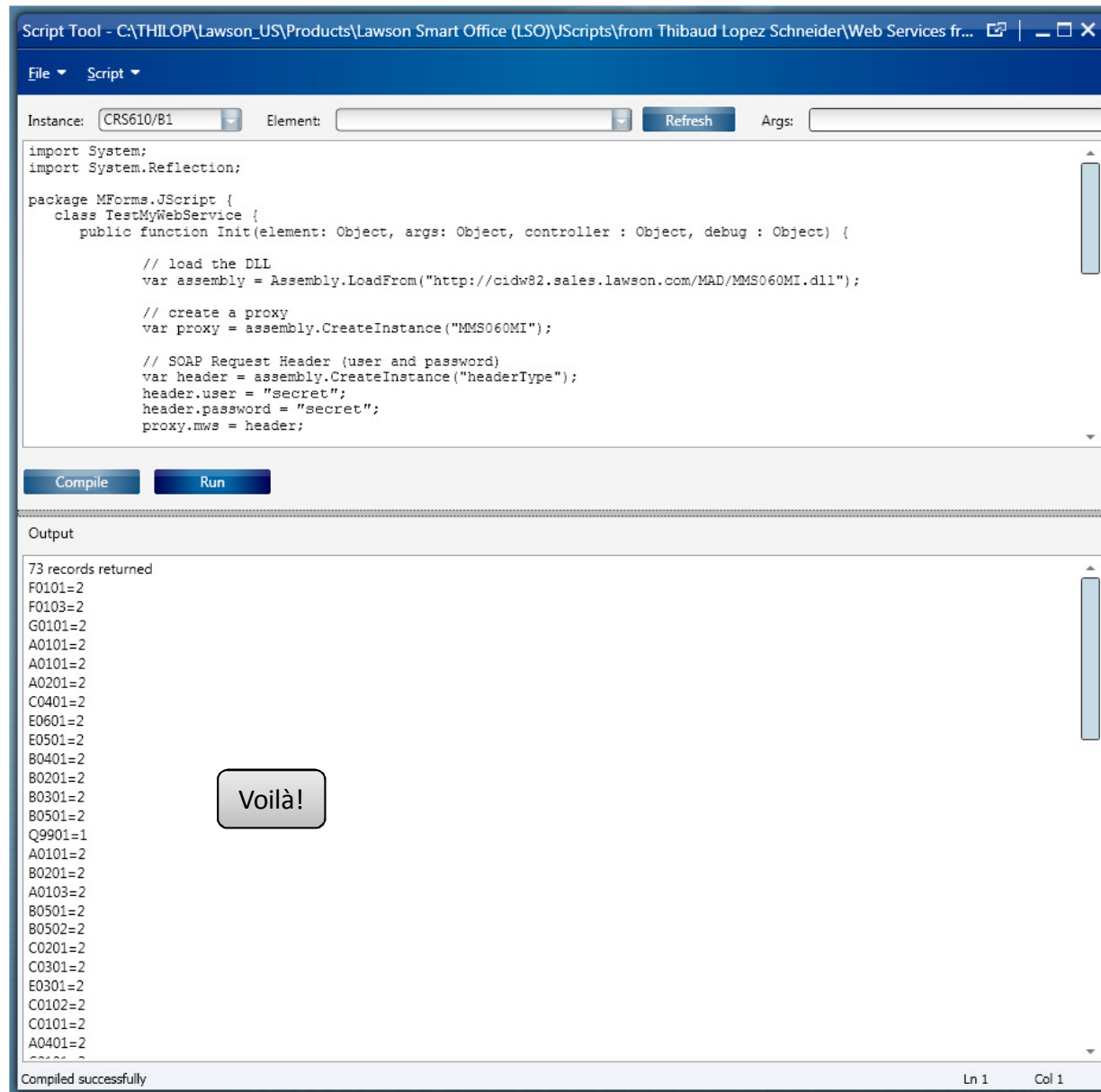


```
1
2 private function CreateObject(typeName : String) {
3     // Check the cache first.
4     var assembly = SessionCache.Get("MyLibrary.dll");
5
6     if (assembly == null) {
7         // Assembly not found in cache, load it from url and store it in the cache.
8         var assembly = ScriptUtil.LoadAssemblyFromUrl("http://host/path/MyLibrary.dll");
9         if (assembly != null) {
10             SessionCache.Add("MyLibrary.dll", assembly);
11         }
12     }
13     if (assembly != null) {
14         return assembly.CreateInstance("MyLibrary." + typeName);
15     }
16     return null;
17 }
18
```

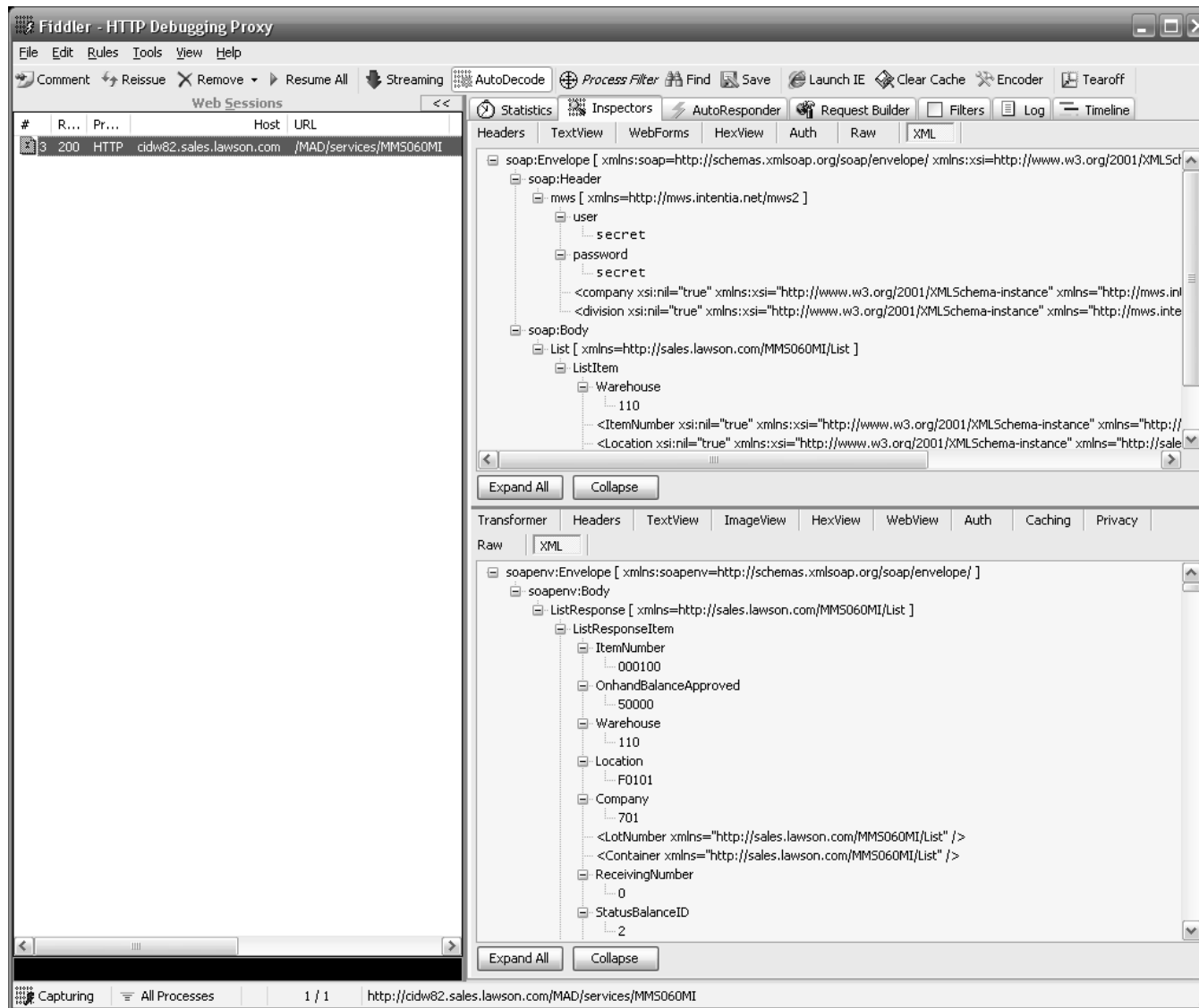
JavaScript file 579 chars 621 bytes 21 lines Ln : 19 Col : 1 Sel : 0 (0 bytes) in 0 ranges Dos\Windows ANSI INS

Note 3: Also, I recommend using exception handling in the code. It's not shown in my example.

## Step 7. Compile and Run the script in Lawson Smart Office with the Script Tool (this example does not require any particular M3 Program Instance).



Optionally, you can check the SOAP Request in Fiddler.



	SOAP validation?	DLL free?	Notepad only?	JScript programming only?
"Big String"	No	Yes	Yes	Yes
Xml Writer	No	Yes	Yes	Yes
Proxy with Visual Studio C#	Yes	No	No	No
wsdl.exe *	Yes	No	No	Yes

*Summary:* This paper presented a new solution to call a Lawson Web Service from a Personalized Script in JScript.NET in Lawson Smart Office. This solution complements the other three known solutions. Now the developer has even more flexibility to choose a solution that best suit its needs. This fourth solution provides SOAP validation, and doesn't require C# programming, nor Microsoft Visual Studio. But it requires to install the free Microsoft .NET SDK, it requires following the proxy syntax, and produces a DLL that must be published to the users. (\*) Peter A Johansson and myself recommend this fourth solution over the other three. Peter adds: "The first three are no longer 'best practice' [...] are 'old and obsolete' and should not be used."

# Discussion

From an architecture point of view, there are good and bad reasons to call a Lawson Web Service from a Personalized Script.

For example, one could call a Lawson Web Service from a standalone script that is executing in its own window in Smart Office, i.e not in an M3 Panel. In that case, using Lawson Web Services would be legitimate as there are no other alternatives (except direct SQL to M3 which is not recommended for writing to M3). Here, Lawson Web Services would have the same role as the IDSP connector had for IBrix.

However, when it comes to Personalized Scripts that live and run inside of an M3 Panel, it's better to access the data from within M3 itself, close to the source, i.e. from the M3 Business Engine with some M3 Java code in MAK, even though it means an M3 modification. There are several technical reasons why: performance, security, dependencies, maintenance, etc.

Special thanks to Peter A Johansson for the code of the DLL solution, and for the review of this paper.

**Thibaud Lopez Schneider**  
[thibaud.lopez.schneider@us.lawson.com](mailto:thibaud.lopez.schneider@us.lawson.com)