Ellipse C# Client

Power and productivity
for a better world™

Ellipse C# Client

Contents

Commercial In Confidence

Legal Disclaimer

The product described in this documentation may be connected to, and/or communicate information and data via, a network interface, which should be connected to a secure network. It is your sole responsibility to ensure a secure connection to the network and to establish and maintain appropriate measures (such as but not limited to the installation of firewalls, application of authentication measures, encryption of data, installation of antivirus programs, etc.) to protect the product, the network, your systems, and the interface against any kind of security breach, unauthorised access, interference, intrusion, leakage, damage, or corruption or theft of data. We are not liable for damages or losses related to any such security breach, unauthorised access, interference, intrusion, leakage, damage, or corruption or theft of data.

Preface

This document provides information on Ellipse Web Services.

Summary information

Confidentiality

The contents of this document are confidential between ABB and its customers. The parties must keep the information herein confidential at all times and not disclose it, or permit it to be disclosed, to any third party, apart from any of their officers, employees, agents or advisers who have a specific need to access the information herein and have agreed to be bound by the terms of confidentiality.

Document Control

Once the project is completed or terminated, this document will revert to an uncontrolled document status. No further advice will be provided, and each recipient may either destroy the document or mark it as obsolete and retain it for future personal reference.

All copies of this document will be issued electronically.

Who should use this guide?

This guide provides information on Ellipse Web Services for Ellipse Technical Consultants and Programmers.

.NET Ellipse WebService

This document serves as a reference for implementing and connecting to the new Ellipse WebServices in .NET.

This document will assume a fairly solid understanding of both .NET and WebServices and their associated terminology, such as WSDL and SOAP.

Screenshots are taken using Visual Studio 2008 and so some of the terminology and procedures may vary slightly in older versions.

Requirements

.NET 2.0 runtime or greater. The 1.0 .NET runtime is missing nullable types which were introduced in 2.0 to handle, among other things, xsi:nil.

Visual Studio 2008, 2005 or 2003. Note that if any Office integration is intended then the Visual Studio Tools for Office is required.

This is an add-on for 2003 and 2005, but is only included in 2008 Professional or Team System.

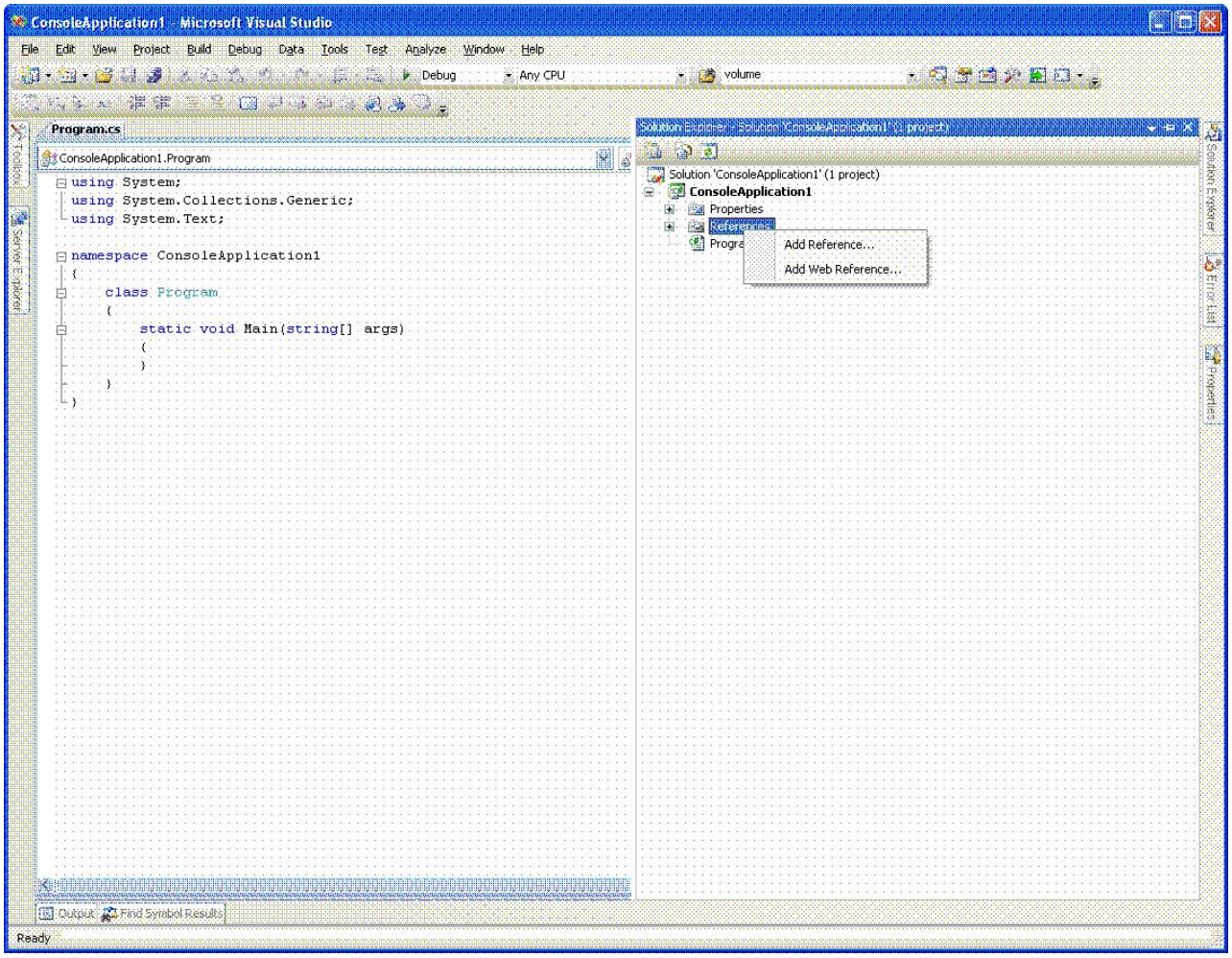
Visual Studio

New Project

Create a new project in VS as required.

Adding a Web Reference

Add a Web Reference to your project by selecting Add Web Reference from either right-clicking on the References folder or the from the Project menu.

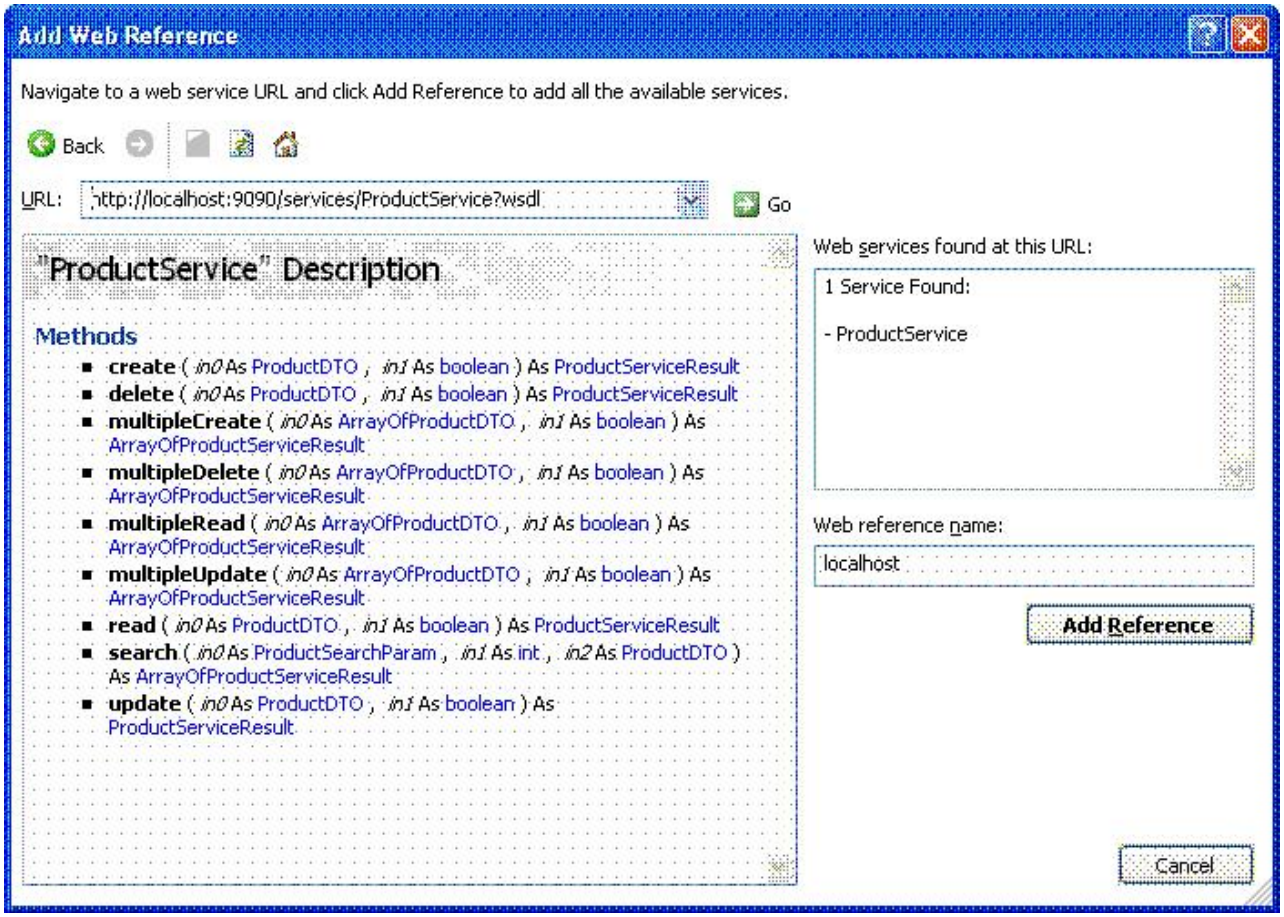


Visual Studio Web Reference

Enter the URL to the service WSDL. eg.

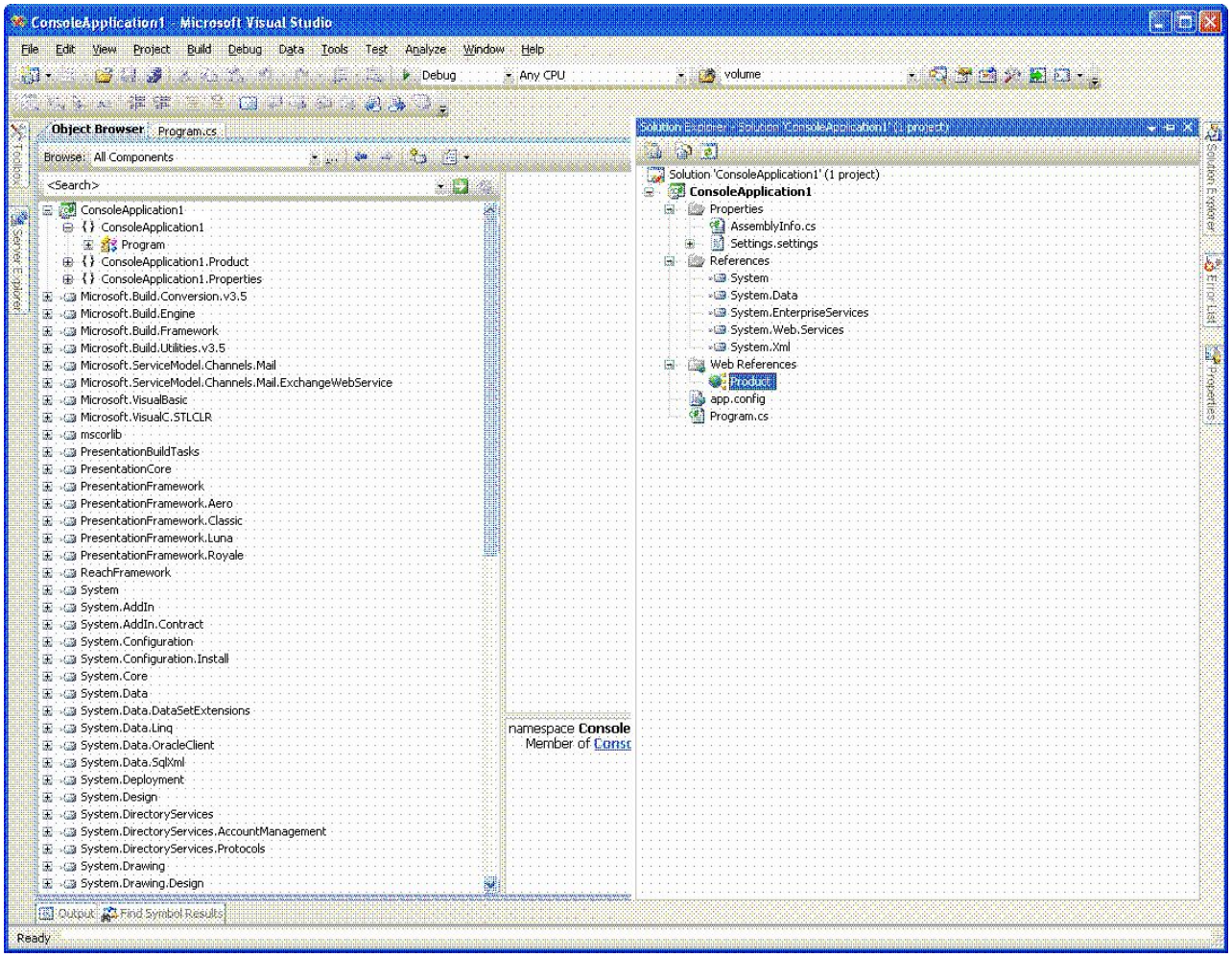
http://host:port/services/ProjectService?WSDL.

Once loaded press the Add Reference button.



Visual Studio Web Reference Dialog

There should now be a new node under the Web References folder in the project view. In addition a namespace of the same name will be present in the Object Browser.



Visual Studio Web Reference Object

Ellipse Web Services Authentication

To be able to authenticate successfully against Ellipse the SOAP request will require that a security header and context details are provided.

The following will need to be added to the project for the Ellipse client web service processing:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Xml;
using System.ServiceModel.Channels;
using System.ServiceModel.Dispatcher;
using System.ServiceModel;
using System.ServiceModel.Description;
using System.ServiceModel.Configuration;

/*
 * http://weblogs.asp.net/paolopia/archive/2008/02/25/handling-custom-soap-headers-via-wcf-behaviors.aspx
 */
namespace EllipseWebServicesClientV3
{
  public class ClientConversation
  {
    public static string username;
    public static string password;
    public static string district;
    public static string position;

    public static void authenticate(string username, string password, string district, string position)
    {
      ClientConversation.username = username;
      ClientConversation.password = password;
      ClientConversation.district = district;
      ClientConversation.position = position;
```

```
    }
}

public class SecurityHeader : MessageHeader
{
    public override string Name
    {
        get { return "Security"; }
    }

    public override string Namespace
    {
        get { return "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"; }
    }

    protected override void OnWriteHeaderContents(XmlDictionaryWriter writer, MessageVersion messageVersion)
    {
        writer.WriteStartElement("UsernameToken");
        writer.WriteElementString("Username", ClientConversation.username);
        writer.WriteElementString("Password", ClientConversation.password);
        writer.WriteEndElement();
    }
}

public class ValueHeader : MessageHeader
{
    private string name;
    private string value;

    public ValueHeader(string name, string value)
    {
        this.value = value;
        this.name = name;
    }

    public override string Name
    {
        get { return name; }
    }

    public override string Namespace
    {
        get { return "http://connectivity.ews.mincom.com/"; }
    }

    protected override void OnWriteHeaderContents(XmlDictionaryWriter writer, MessageVersion messageVersion)
    {
        writer.WriteAttributeString("value", value);
    }
}

public class EllipseMessageInspector : IDispatchMessageInspector, IClientMessageInspector
{
    #region Message Inspector of the Service

    public object AfterReceiveRequest(ref Message request, IClientChannel channel, InstanceContext instanceContext)
    {
        return null;
    }

    public void BeforeSendReply(ref Message reply, object correlationState)
    {
    }

    #endregion

    #region Message Inspector of the Consumer

    public void AfterReceiveReply(ref Message reply, object correlationState)
    {
    }

    public object BeforeSendRequest(ref Message request, IClientChannel channel)
    {
        // Prepare the request message copy to be modified
        MessageBuffer buffer = request.CreateBufferedCopy(Int32.MaxValue);
        request = buffer.CreateMessage();
```

```
      // Simulate to have a random Key generation process
      request.Headers.Add(new SecurityHeader());
      request.Headers.Add(new ValueHeader("District", ClientConversation.district));
      request.Headers.Add(new ValueHeader("Position", ClientConversation.position));

      return null;
    }

    #endregion
  }

  [AttributeUsage(AttributeTargets.Class)]
  public class EllipseHeaderBehavior : Attribute, IEndpointBehavior
  {
    #region IEndpointBehavior Members

    public void AddBindingParameters(ServiceEndpoint endpoint, System.ServiceModel.Channels.BindingParameterCollection bindingParameters)
    {
    }

    public void ApplyClientBehavior(ServiceEndpoint endpoint, System.ServiceModel.Dispatcher.ClientRuntime clientRuntime)
    {
      clientRuntime.MessageInspectors.Add(new EllipseMessageInspector());
    }

    public void ApplyDispatchBehavior(ServiceEndpoint endpoint, System.ServiceModel.Dispatcher.EndpointDispatcher endpointDispatcher)
    {
      ChannelDispatcher channelDispatcher = endpointDispatcher.ChannelDispatcher;
      if (channelDispatcher != null)
      {
        foreach (EndpointDispatcher ed in channelDispatcher.Endpoints)
        {
          ed.DispatchRuntime.MessageInspectors.Add(new EllipseMessageInspector());
        }
      }
    }

    public void Validate(ServiceEndpoint endpoint)
    {
    }

    #endregion
  }

  public class EllipseBehaviorExtensionElement : BehaviorExtensionElement
  {
    protected override object CreateBehavior()
    {
      return new EllipseHeaderBehavior();
    }

    public override Type BehaviorType
    {
      get
      {
        return typeof(EllipseHeaderBehavior);
      }
    }
  }
}
```

App.config

The projects app.config file will need to be setup for the Ellipse Web Services Client. If one does not exist then create one in the root directory of the project. This will enable the custom authentication headers to be added to every SOAP message being passed to Ellipse.

An example app.config file is as below:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding name="LocationServiceHttpBinding" closeTimeout="00:01:00"
          openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
          allowCookies="false" bypassProxyOnLocal="false" hostNameComparisonMode="StrongWildcard"
          maxBufferSize="65536" maxBufferPoolSize="524288" maxReceivedMessageSize="65536"
          messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
          useDefaultWebProxy="true">
          <readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"
            maxBytesPerRead="4096" maxNameTableCharCount="16384" />
          <security mode="None">
            <transport clientCredentialType="None" proxyCredentialType="None"
              realm="" />
```

```
            <message clientCredentialType="UserName" algorithmSuite="Default" />
          </security>
        </binding>
      </basicHttpBinding>
    </bindings>
    <client>
      <endpoint address="http://localhost:9080/ews/services/LocationService"
        binding="basicHttpBinding" bindingConfiguration="LocationServiceHttpBinding"
        contract="Location.Location" name="LocationServiceHttpPort" behaviorConfiguration="serviceOneBehavior" />
    </client>

   <extensions>
    <behaviorExtensions>
      <add name="customBehavior" type="EllipseWebServicesClientV3.EllipseBehaviorExtensionElement, EllipseWebServicesClientV3, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null" />
    </behaviorExtensions>
   </extensions>

   <behaviors>
    <endpointBehaviors>
      <behavior name="serviceOneBehavior">
        <customBehavior />
      </behavior>
    </endpointBehaviors>
   </behaviors>
  </system.serviceModel>
</configuration>
```

App.config file with system.net

For the customer who has built their own Web Service Client from Ellipse 8.7 onwards, to avoid timeouts which cause errors such as "502 Bad Gateway" in EWS, the following change must be made to the Apache Reverse Proxy configuration.

```
<system.net>
   <settings>
      <servicePointManager expect100Continue="false" />
   </settings>
</system.net>
```

This is because the reverse proxy doesn't handle the "Expect 100 Continue" header properly.

An example app.config file with this code is as below:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding name="LocationServiceHttpBinding" closeTimeout="00:01:00"
          openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
          allowCookies="false" bypassProxyOnLocal="false" hostNameComparisonMode="StrongWildcard"
          maxBufferSize="65536" maxBufferPoolSize="524288" maxReceivedMessageSize="65536"
          messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
          useDefaultWebProxy="true">
          <readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"
            maxBytesPerRead="4096" maxNameTableCharCount="16384" />
          <security mode="None">
            <transport clientCredentialType="None" proxyCredentialType="None"
              realm="" />
            <message clientCredentialType="UserName" algorithmSuite="Default" />
          </security>
        </binding>
      </basicHttpBinding>
    </bindings>
    <client>
      <endpoint address="http://localhost:9080/ews/services/LocationService"
        binding="basicHttpBinding" bindingConfiguration="LocationServiceHttpBinding"
        contract="Location.Location" name="LocationServiceHttpPort" behaviorConfiguration="serviceOneBehavior" />
    </client>

   <extensions>
    <behaviorExtensions>
      <add name="customBehavior" type="EllipseWebServicesClientV3.EllipseBehaviorExtensionElement, EllipseWebServicesClientV3, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null" />
    </behaviorExtensions>
   </extensions>

   <behaviors>
    <endpointBehaviors>
      <behavior name="serviceOneBehavior">
        <customBehavior />
      </behavior>
    </endpointBehaviors>
   </behaviors>
  </system.serviceModel>

  <system.net>
    <settings>
      <servicePointManager expect100Continue="false" />
    </settings>
  </system.net>
</configuration>
```

Authenticate

In the appropriate source file you will need to import/use the ClientConversation class from the EllipseWebServiceClient namespace. From this class you need to call the authenticate method with the appropriate username, password.

```
01. ClientConversation.authenticate("username", "password");
```

Example

Here is a very trivial example of calling the read method on the ProductService. This is very similar to most of the services calls, with the relevant values being set on the DTO before the operation is called. See Web_Services_examples for more examples.

```
01. ProductService service = new ProductService();
02. ProductDTO dto = new ProductDTO();
03. ProductServiceResult result = service.read(context, dto);
04. Console.WriteLine(result.productDTO.partNumber);
```