

# Westcon Web Services Developer's Guide

<b>Revision History</b>			
<b>Document Version</b>	<b>Publication Date</b>	<b>Author</b>	<b>Record Description</b>
1.0	2/2/2008	Sedat Behar	Draft
1.1	10/22/2008	Elisa Blackman	Release Version

## Your Specific Implementation Details:

<b>Customer</b>	This is your Company Name
<b>Account: (username)</b>	This is your Account number as it appears in our backend systems
<b>Authentication Key: (password)</b>	This is a 36-character key with which you will be authenticated
<b>Your Westcon Company</b>	This value is data that is specific to your account which should be included with your web service call
<b>RBU / HRBU</b>	This value is data that is specific to your account which should be included with your web service call
<b>Main WWS-URL</b>	<a href="https://webservices.westcon.com">https://webservices.westcon.com</a>
<b>Availability Web Service URL</b>	<a href="https://webservices.westcon.com/availability.asmx">https://webservices.westcon.com/availability.asmx</a>

## Contents

Your Specific Implementation Details: .....	2
Introduction .....	4
1.1 Anatomy of the Web Service .....	6
1.2 Security .....	7
1.2.1 Configure the Client for Security .....	8
1.3 Sample Code for the client (Availability Call).....	10
1.4 Configure non .net Clients.....	12
Inbound Requests .....	13
SOAP failures and Exceptions .....	15

## Introduction

At Westcon Group, we help ease the flow of your own business process by providing you with the information you need on your transactions—when you need it. The use of Web Services is an efficient method to exchange information directly between systems. These efficiencies are realized through the amount and frequency of distributor information which can be automatically entered directly into your system. The direct entry eliminates the manual keying of information as well as the time and cost of mailing and/or tracking documents.

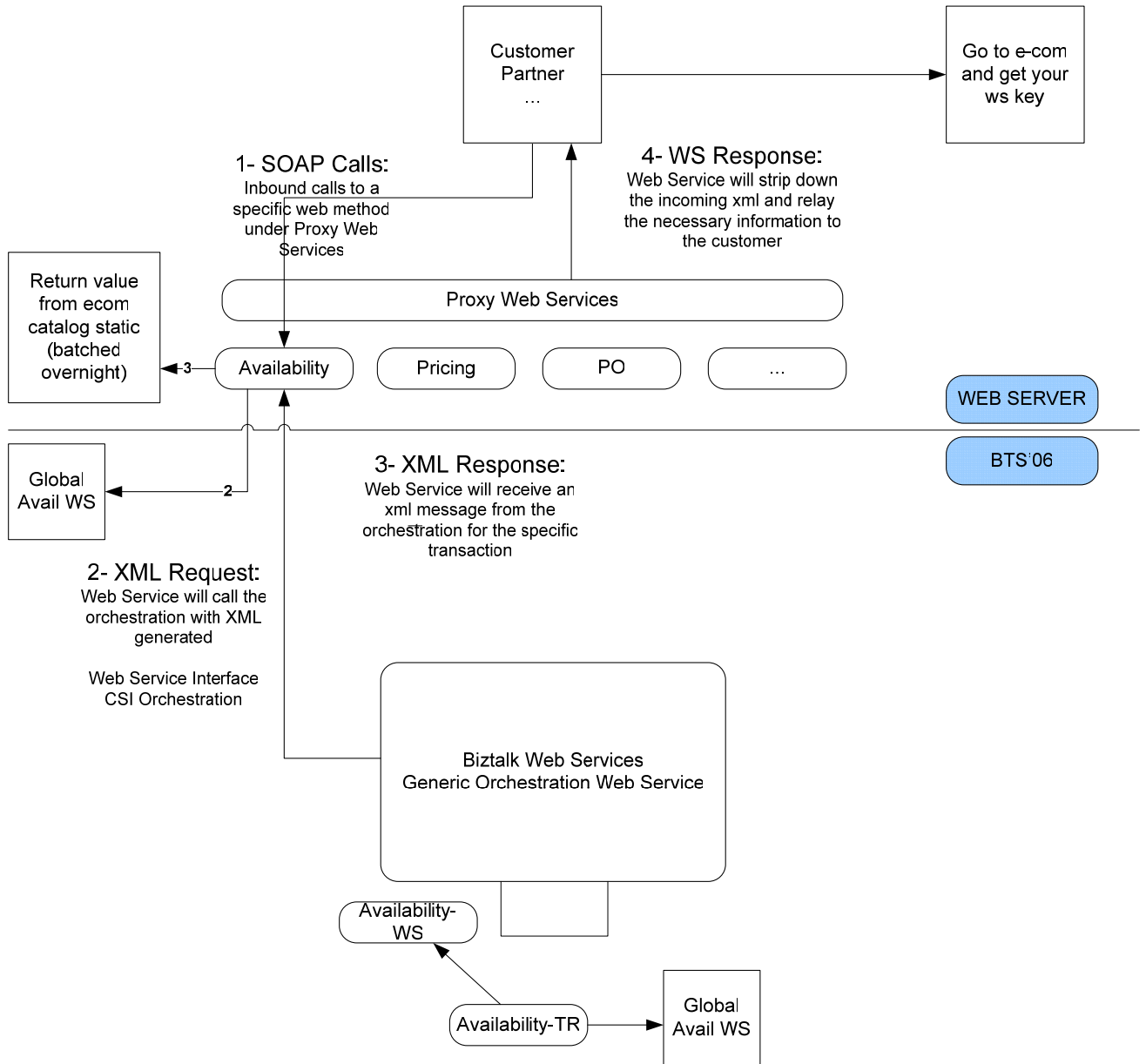
This document provides detailed instructions for using Westcon Web Services. These services will help you obtain pricing/availability/catalog related details as well as submitting a Purchase Order. At the end of the day, we would like you to run the Catalog Web Service and receive your custom catalog, search products and run Availability and Pricing transactions (Web Services) to get latest data. Depending on the response, form a Purchase Order document and submit it using the Purchase Order Web Service. Later, using the Order Tracking Web Service, check the status of your order; once shipped, track the status of the shipment using the Shipment Tracking Web Service. Finally, if there is a return of merchandise, use the RMA Web Service to execute it. Our end goal is to provide all of our sales-business-functions seamlessly and in a service-oriented way to you.

Here are some topics covered in this guide:

- Guidelines
- Security (authentication and authorization)
- Web Service Methods and implementations
- Case Scenarios

In our implementation, we chose to expose singular web services from which you would create a proxy class instance and call available web methods. At this point, these methods are expecting valid schemas that will be made available in this guide. We are trying to make sure that we provide detailed and clear exceptions and errors to ease your handling of calls. We are compliant to SOAP 1.1 and 1.2 and hosting our web services offerings via IIS 6.0. We are hoping that you will be able to implement these services with no hassle.

Below is a detailed look at our architecture:



## 1.1 Anatomy of the Web Service

- 1- Customer makes a SOAP Call: before performing this call, the customer must obtain the WS-key (which with the account number will become the authentication username and password) and other material related to deployment and specifics for different transactions. (HRBU for availability etc)
- 2- Call is received by the Proxy Web Services, which authenticates the user. In case of an error, the connection is terminated with a SOAP exception (login failed)
- 3- After a successful login, user's web service method call executes another web service call to the BizTalk equivalent of the transaction. Biztalk Server executes this transaction; which is one of the following:
  - a. Web Service Call to an Oracle Store Procedure (Availability)
  - b. Store Proc call to a Datamart (Pricing)
  - c. Call to JDE using the JDE Adapter
- 4- Response is forwarded back to the pending proxy web service method which forwards the request back to the customer. At this point, the transaction is terminated, however certain values are cached.

## 1.2 Security

An important aspect for the consumption of the web services is Security. Although transactions such as pricing or availability are not very content – sensitive; they are, fact, caller-sensitive; that is, we need make sure that the caller group is a limited, authenticated set of users that can use the site.

One way of making sure that this is the case is to give them an authentication key; a randomly generated key that the user needs to include in the Web Services WSE proxy request. One of the ideal places that this key can be obtained is the e-commerce website!! The customer, when they login, would see a link that says “please obtain your web services key here” which would take them to a web page where the key would appear as well as instructions on how to integrate certain web services to their e-commerce architecture. This key would be an assignment to the “account”. This value resides in the web services database. This key can be changed on an email request only at this point. In the future, this functionality will be embedded to our e-commerce sites.

Currently, our implementation is using WSE 3.0 Web Services Extensions offered by Microsoft. The security mechanism used underneath is WS-Security 1.1 compliant. You can find more about WSE 3.0 at the following site:  
<http://blogs.msdn.com/mfussell/archive/2006/05/25/607820.aspx>

## 1.3 Configure the Client for Security

After enabling the client application to support WSE 3.0 during General Setup, you must enable policy support for it. If your application does not currently have a policy cache file, you can add one for this purpose, and enable policy support by performing the following steps.

### To add policy support to a WSE 3.0-enabled Visual Studio 2005 project

1. In Visual Studio 2005, right-click the application project and select WSE Settings 3.0.
2. On the **Policy** tab, select the **Enable Policy** checkbox. Selecting this setting adds a policy cache file with the default name *wse3policyCache.config*.
3. Under **Edit Application Policy**, click **Add**, and then type a policy friendly name for the new application policy, such as "usernameTokenSecurity."
4. Click **OK** to start the WSE Security Settings Wizard, and then click **Next**.
5. On the **Authentication Settings** page, the wizard provides a choice to secure a service or a client. Select **secure a client application** to configure the client.
6. The wizard also provides a choice of authentication methods in the same step. Select **Username**, and then click **Next**.
7. On the **Optionally Provide Username and Password** page, the wizard provides you with options to define a user name and password. Ensure that the **Specify Username Token in code** checkbox is selected and click **Next**.

On the **Message Protection** page, you configure options for message protection. For transport layer security, select **None (rely on transport protection)** for the **Protection Order** to use the **usernameOverTransportSecurity** assertion.

8. Click **Next**.
9. If you selected **None (rely on transport protection)** to use transport security in Step 8, skip this step.
10. On the **Create Security Settings** page, review your settings, and then click **Finish**.

After you complete these tasks, your client security policy should look similar to the following code example. Examples for **usernameOverTransportSecurity**:

```
File: wse3PolicyCache.config
<pol icies xmlns="http://schemas.microsoft.com/wse/2005/06/po li cy">
  <extensi ons>
    <extensi on name="usernameOverTransportSecuri ty"
type="Mi crosoft. Web. Servi ces3. Desi gn. UsernameOverTransportAsserti on,
Mi crosoft. Web. Servi ces3, Versi on=3. 0. 0. 0, Cul ture=neutral ,
Publ icKeyToken=31bf3856ad364e35" />
    <extensi on name="requi reActi onHeader"
type="Mi crosoft. Web. Servi ces3. Desi gn. Requi reActi onHeaderAsserti on,
Mi crosoft. Web. Servi ces3, Versi on=3. 0. 0. 0, Cul ture=neutral ,
Publ icKeyToken=31bf3856ad364e35" />
  </extensi ons>
</pol icy name="usernameTokenSecuri ty">
```

```
<usernameOverTransportSecurity />
<requireActionHeader />
</policy>

</policies>
```

When you add a Web reference to the service from the client application, two proxies are generated for the Web service—one is a non-WSE 3.0 proxy and the other is WSE 3.0-enabled. In this guidance, Microsoft uses the WSE 3.0-enabled proxy class, which is defined as name + "Wse". For example, if your Web service is named "westconAvailability," your WSE 3.0-enabled Web service proxy class name would be "westconAvailabilityWse."

The following code example provides an example of how to initialize an instance of a **UsernameToken** and to bind the appropriate policy defined in the preceding policy file to the Web service proxy. You can copy or insert this code into a new code module.

The following are **the Client side changes/requirements**:

- 1- If coding with .net, make sure that you are using System.Web.Services.Protocols.
- 2- Create the WSE proxy class (westconWebServicesWse)

```
ex: westcon.westconWebServicesWse proxyWse = new
westcon.westconWebServicesWse();
```

- 3- Create a security token and load it to the proxy

```
UsernameToken token = new UsernameToken(userName, password,
PasswordOption.SendPlainText);

proxyWse.SetClientCredential(token);
proxyWse.SetPolicy("usernameTokenSecurity");
```

- 4- Prepare your request object and a blank response object (more information on this later)
- 5- Call the web service with the request object as an argument and assign the results to the response object. You can call the method asynchronously, which would allow the page/application to load or do something else without having to wait for the web service call to return. Calls are expected to run under 2-3 seconds max, with average internal times around 0.5 seconds. (Please estimate for the internet traffic between our servers)

## 1.4 Sample Code for the client (Availability Call)

More information about the nature of the request and response objects is mentioned in the following sections.

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

using Microsoft.Web.Services3;
using Microsoft.Web.Services3.Security;
using Microsoft.Web.Services3.Security.Tokens;

namespace WSTester
{
    public partial class wsTesterQTYT : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            // Create also a WSE class security-authn object
            westcon.westconWebServicesWse proxyWSE = new
            westcon.westconWebServicesWse();

            // Sent in plainText, the transport channel will be secure SSL
            String userName = " USE ACCOUNT (USERNAME) MENTIONED ON PAGE 3";
            String passWord = "USE TOKEN MENTIONED ON PAGE 3";
            UsernameToken token = new UsernameToken(userName, passWord,
            PasswordOption.SendPlainText);

            proxyWSE.SetClientCredential(token);
            proxyWSE.SetPolicy("usernameTokenSecurity");

            // Empty Response Object
            westcon.AvailabilityResponse res = new westcon.AvailabilityResponse();

            // Empty Request Object
            westcon.AvailabilityRequest reqWS = new westcon.AvailabilityRequest();

            // Form Request Object
            reqWS.Company = " USE COMPANY MENTIONED ON PAGE 3";
            reqWS.CustomerAddress = " USE ACCOUNT (USERNAME) MENTIONED ON PAGE 3";
            reqWS.EndVert = "COM";
            reqWS.HeaderBusinessUnit = " USE RBU MENTIONED ON PAGE 3";

            // Item Details (enter up to 10 per request)
            westcon.ItemDetail item = new westcon.ItemDetail();
            item.BranchPIant = "";
            item.SecondItemNumber = tbItem2ndNumber.Text;
            item.LineType = "QS";
            item.OrderQty = tbOrderQty.Text;

            westcon.ItemDetail[] items = new westcon.ItemDetail[1];
            items[0] = item;
            reqWS.ItemDetail = items;

            reqWS.OrderType = "S";
            reqWS.PODetailFlag = "1";
            reqWS.TransCurCd = "USD";
        }
    }
}
```

```
try
{
    res = proxyWSE.itemAvailability(reqWS);
}
catch (Exception ex)
{
    Response.Write (ex.ToString() + "</b>");
}

if (res.ItemAvailability != null)
{
    lbRes.Text += th.printAvailRes(res, true);
}
else
{
    Response.Write ("Error: Availability Response is Null<p>");
}
}
}
```

## 1.5 Configure non .net Clients

Microsoft WSE 3.0 is used as the Security Authentication mechanism in our implementation which is WS-Security 1.1 compliant. Our security mechanism, usernameOverTransport is WS-Security 1.0 compatible. This allows all WS-Security 1.0 conforming clients to make secure connections and implement our web services.

## Inbound Requests

For each transaction that will be exposed as a web service, a request and response object is built, which will be sent to the respective BizTalk Orchestration Web Service, referred to in the BizTalk 2006 Exposed Web Services section, which will respond with the response object. These are specified in the WSDLs inherently using the external schemas that later map to internal schemas. Essentially, there is nothing that the customer should worry about except for the schema in which the data should be submitted and the schema in which it will be responded.

In case there is a failure at BizTalk end, we will be forwarding the error message (SOAP exception) to the client. You can find more info about the transaction errors and translations below.

You will see that when the appropriate web service method is referenced, the available request and response objects will be shown (using Intellisense in Visual Studio or any modern IDE). At this point, all the customer needs to do is to initiate these variables, fill them in with appropriate data and send in the request. On most modern IDEs, provided the WSDL schema, a developer can find out what the requested objects are. The WSDL for the web service would also show the full implementation details for the specific transaction.

Although there are many variables available in the request response objects, the customer does not need to fill every single one of them. We are providing a set of required elements in the definition of each one of the transactions. You can find more details when the transactions are explained below.

For each inbound request, the transaction will operate in the same manner: WSDL implies the name of the operation that needs to be called (E.g.: **itemPricing** or **itemAvailability**). Request object (E.g.: **Westcon.AvailabilityRequest** where **Westcon** is the name of the proxy object) will be populated and be provided to the web service method. The response (E.g.: **Westcon.AvailabilityResponse res**) needs to be captured using the response object associated to the web method.

e.g.: `public btsAvailability.AvailabilityResponse itemAvailability  
(btsAvailability.AvailabilityRequest request)`

Common to all web method calls is the Security. Client needs to initiate a client token which should include the username (account number of the customer) and a long key provided by Westcon Group. In .net, initiation of the token and its assignment to the call is as follows.

```
availabilityDirect.availabilityWse proxyWSE = new availabilityDirect.availabilityWse();  
  
    // UsernameOverTransport security mechanism of WSE 3.0  
    // initiate security token  
    String userName = "<company name on page 3>";  
    String password = "<admin key on page 3>";  
    UsernameToken token = new UsernameToken(userName, password, PasswordOption.SendPlainText);  
  
    // attach token to proxy and set policy type  
    lbDebug.Text += "Load token to proxyWSE<br>";  
    proxyWSE.SetClientCredential(token);  
    proxyWSE.SetPolicy("usernameTokenSecurity");
```

All web service operations are enclosed in a try/catch block and will send out detailed SOAP exceptions if necessary. Please see the SOAP Exceptions and Failures section for more detail.

## SOAP failures and Exceptions

We need to respond back to the caller of the Web Services with appropriate messages so that we can provide an increased quality of service. These messages are a good way to allow the customers to act accordingly when they encounter these errors. Most of the time, these exceptions will be thrown in the BizTalk layer and be forwarded to the Customer by the Proxy. Customer should react based on the exception caught.

Location	Error	Error Text
Compass (captured on Biztalk)	Compass Down	Error text
Biztalk (captured on Proxy WS)	Biztalk Down	We are sorry, our Biztalk server is not responding to your request at the moment.
Biztalk (captured on Proxy WS)	Validation Error	The document you have provided is not valid. Please correct and resubmit document.
Proxy WS	Login Error	Your account number or your account key is not authenticating. Please check.
	10 error codes...	Approp biztext

Error Code	Error Description	Fault Location	Reason
To be confirmed	Biztalk service is down	Proxy Service	The BizTalk service is currently down for maintenance or another issue.
To be confirmed	Validation error	BizTalk Transaction	The client hasn't supplied mandatory fields or a business validation exception has occurred.
To be confirmed	Security key account number invalid	Proxy Service	The security key account number that has been supplied is invalid.
To be confirmed	Timeout	Proxy Service	If the BizTalk Service takes too long to respond then return a timeout fault exception.
To be confirmed	GAWS down	GAWS	GAWS web service might be down.

More error messages are in the MOM Rules.xls document, where the error details are included in the application. These errors will be relayed to the customer in a different manner. These will be received from Biztalk.