

Westcon Web Services Developer's Guide

Revision History			
Document Version	Publication Date	Author	Record Description
1.0	2/2/2008	Sedat Behar	Start
1.1	7/10/2008	Sedat Behar	Update with Production Deployment
1.2	10/22/2008	Elisa Blackman	Revised

Contents

- Westcon Web Services1
- Developer’s Guide1
- Westcon Web Services INFO TABLE: Your Specific Implementation Details:3
- Introduction.....4
- Anatomy of the Web Service.....6
- Security7
 - 1.1 Configure the Client for Security.....8
 - 1.2 Sample Code for the client (Availability Call)10
 - 1.3 Configure non .net Clients12
- Inbound Requests.....14
- Availability:16
 - 1.1 Availability Request.....18
 - 1.2 Availability Response19
- SOAP failures and Exceptions.....22

Westcon Web Services INFO TABLE: Your Specific Implementation Details:

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

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.

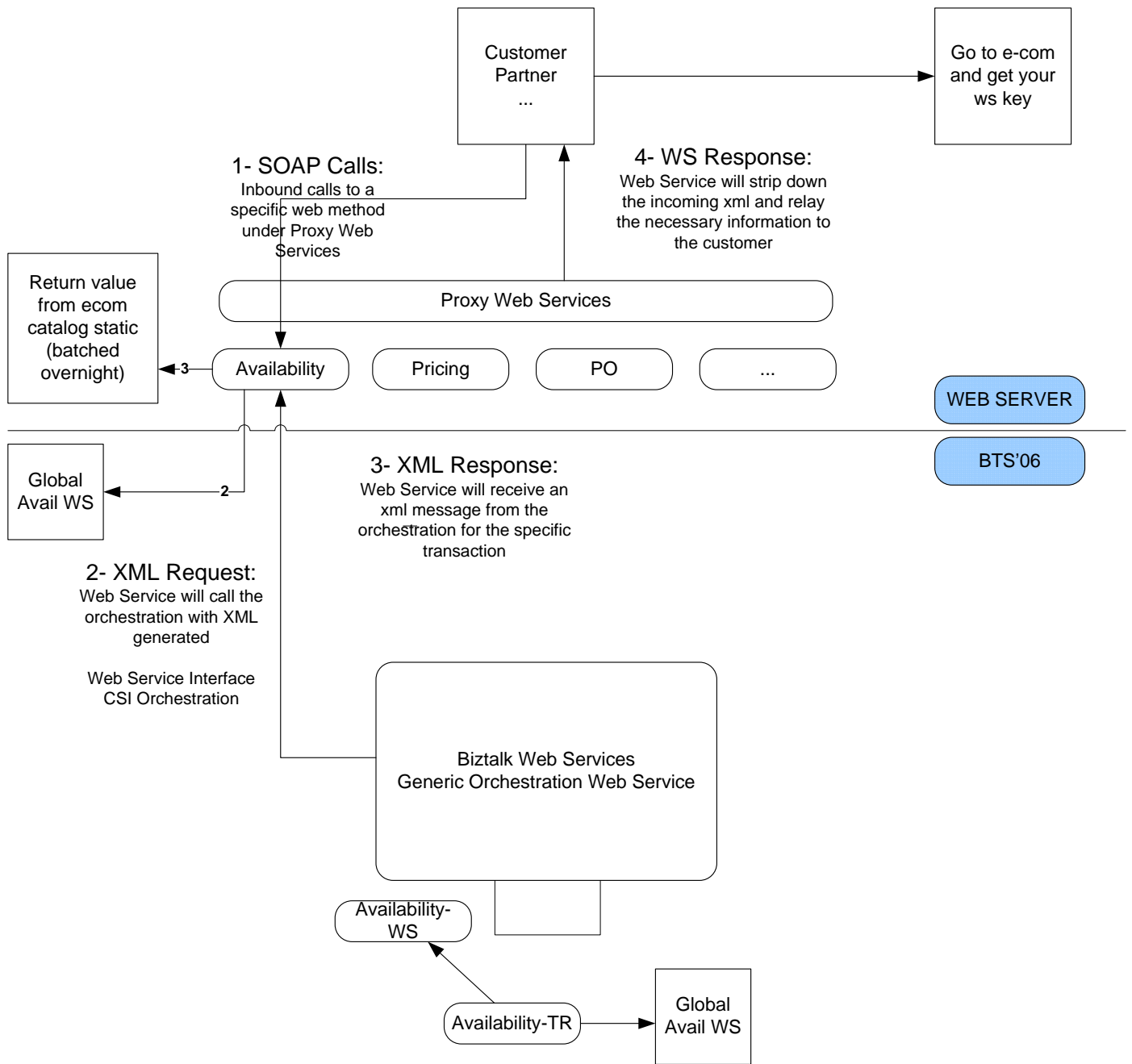
This document provides detailed instructions for using Westcon Web Services. These services will help you obtain pricing/availability/catalog related details as well as submitting a Purchase Order. At the end of the day, we would like you to run the Catalog Web Service and receive your custom catalog, 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:



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.

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>

1.1 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
<policy xmlns="http://schemas.microsoft.com/wse/2005/06/policy">
  <extensions>
    <extension name="usernameOverTransportSecurity"
      type="Microsoft.Web.Services3.Design.UsernameOverTransportAssertion,
      Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral,
      PublicKeyToken=31bf3856ad364e35" />
    <extension name="requireActionHeader"
      type="Microsoft.Web.Services3.Design.RequireActionHeaderAssertion,
      Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral,
      PublicKeyToken=31bf3856ad364e35" />
  </extensions>
  <policy name="usernameTokenSecurity">
    <usernameOverTransportSecurity />
    <requireActionHeader />
  </policy>
</policy>
```

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)

1.2 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.BranchPlant = "";
            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>");
    }
}
}
```

1.3 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.

Unfortunately, the Security details specified in the WS-Security layer is not visible in the WSDL we expose. Even if you do not have a “programmatic” way of making the call compliant with the WS-Security 1.0, you can use the following post request and response to communicate with our web services:

Availability Request

Make sure to set up the content-type to text/xml;charset=UTF-8 in the post request. Also, you need to make sure that the expiry dates for messages are up to date. Make sure to change your username and password as well.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
<soap:Header>
<wsa:Action>http://webservices.westcon.com/itemAvailability</wsa:Action>
<wsa:MessageID>urn:uuid:4a66732e-c9ce-4e6d-95e6-cf598399a2c7</wsa:MessageID>
<wsa:ReplyTo><wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa
:Address>
</wsa:ReplyTo><wsa:To>https://webservices.westcon.com/availability.asmx</wsa:To>
<wsse:Security soap:mustUnderstand="1">
<wsu:Timestamp wsu:Id="Timestamp-bce68ceb-656d-4403-8fbf-fc386a51b787">
<wsu:Created>2008-11-11T21:08:33Z</wsu:Created>
<wsu:Expires>2009-11-11T21:13:33Z</wsu:Expires>
</wsu:Timestamp>
<wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd" wsu:Id="SecurityToken-0cfed848-f6e6-4549-80d0-2dfe22761bdc">
<wsse:Username>88141652</wsse:Username>
<wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-
profile-1.0#PasswordText">9WPE7dhgufg6cKoHa16sOaIUFiSnGvUSrSz8</wsse:Password>
<wsse:Nonce>q/HNe+V6lZ77At5nuLsllg==</wsse:Nonce>
<wsu:Created>2010-01-18T21:08:33Z</wsu:Created>
</wsse:UsernameToken>
</wsse:Security>
</soap:Header>
```

```
<soap:Body><itemAvailability xmlns="http://webservices.westcon.com"><request><Company
xmlns="">00011</Company><OrderType xmlns="">S</OrderType>
<HeaderBusinessUnit xmlns="">119992</HeaderBusinessUnit><CustomerAddress
xmlns="">1107922</CustomerAddress><TransCurCd xmlns="">USD</TransCurCd>
<EndVert xmlns="">COM</EndVert><PODetailFlag xmlns="">0</PODetailFlag><ItemDetail
xmlns=""><BranchPlant /><SecondItemNumber>WIC-1T=</SecondItemNumber>
<LineType>QS</LineType><OrderQty>1</OrderQty></ItemDetail></request></itemAvailability>
</soap:Body></soap:Envelope>
```

Please make sure that the following comply:

- Set the correct username and password
- Set the content-type to text/xml;charset=UTF-8 in the post request
- Update the timestamps and GUIDs
- wsa:MessageID and other “GUIDs” can be reused.

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 = "<use Company ID on page 3>";  
String passWord = "<use Authentication Key on page 3>";  
UsernameToken token = new UsernameToken(userName, passWord, PasswordOption.SendPlainText);  
  
// attach token to proxy and set policy type  
IbDebug.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.

Availability:

Availability web service returns the real time availability information for a given product. It can retrieve the availability of up to 10 products at a time. Most likely scenario would be to retrieve product information from Westcon Catalog, provide details as mentioned in the table below and perform the call. The response will contain the item availability data in an array.

If a part is not found or obsolete, an error will be forwarded to the customer. Availability search takes item restriction into account.

Currently, availability web service is hosted at <https://webservices.westcon.com/availability.asmx>. You will find the following web methods under this service:

Web Method	Description
itemAvailabilityViaBiztalk2006	This web service goes via Biztalk 2006 to get the availability information. It uses the same proxy, request, response schemas as below. This method will be renamed to itemAvailability once Biztalk 2006 Layer is optimized
itemAvailability	This method by-passes the Biztalk 2006 environment and gets availability from GAWS. One we switch back to the Biztalk 2006 implementation, itemAvailabilityViaBiztalk2006 method will be renamed to itemAvailability and this method will be renamed to itemAvailabilityFromGAWS
itemAvailabilityFromGAWSAsynch	This method is identical to the itemAvailability method except for the part where call is executed to GAWS. While the call in itemAvailability to GAWS is synchronous, this version is asynchronous.
itemAvailabilityFromGAWSReturnString	This method returns the XML response from GAWS in a string variable.
helloAvailability	Echoes incoming string; meant to be used for connectivity purposes

Customer would be creating the following objects for an Availability call: (where Westcon is the name of the web reference) (Customer \leftrightarrow Proxy)

Proxy	<code>westcon.westconWebServicesWse proxyWSE = new westcon.westconWebServicesWse();</code>
Request	<code>westcon.AvailabilityRequest req = new westcon.AvailabilityRequest();</code>
Response	<code>westcon.AvailabilityResponse res = new westcon.AvailabilityResponse();</code>
Call Method	<code>avRes = avProxy.AvailabilityRequest(req);</code>

1.1 Availability Request

Availability Request: Contains variables on the customer call as well as an array (ItemDetail) of type ItemDetail				REQUIRED
Element	SubElement	GAWS Equivalent	Example	Customer Information
Company		szCompany	“00011”	Please use the company variable in your Info Table
CustomerAddress		mnCustomerAddress	“1234567”	Please use the Account (username) variable in your Info Table
EndVert		szEndVert	“COM”	Use three letter vertical codes (e.g.: COM, GOV, EDU,...)
HeaderBusinessUnit		szHeaderBusinessUnit	“119920”	Please use the RBU / HRBU variable in your Info Table
ItemDetail[] (of type ItemDetail)				You will be creating an ItemDetail object for each of the items you want to query up-to 10 items. Please initiate the values below for each item
	BranchPlant	szBranchPlant	“”	Do not initialize. This is for potential future use. You can refer to the BranchPlant column in the catalog for this variable.
	LineType	szLineType	“QS”	Optional; Use the value provided in the catalog for this item. You can also default to “QS”
	OrderQty	mnOrderQty	“12”	Quantity Required
	SecondItemNumber	sz2ndItemNumber	“Wic-1t=”	Item name/serial needs to be entered here. You can find this information in the catalog. There is no search functionality embedded to the call.
OrderType		szOrderType	“QS”	Optional; you can default to QS or leave blank/null
PODetailFlag		mnPODetailFlag		Optional; please ignore; at this point of the implementation, we will not be exposing PO Details on items.
TransCurCd		szTransCurCd	“USD”	Please enter the 3-character currency code for your transaction (USD, EUR, ...)

Availability Request Notes:

- Most of the sub-elements are in type String.
- You can put in up to 10 items in the ItemDetail[] array.

1.2 Availability Response

Availability Response					
Contains an ItemAvailability[] array (of type AvailabilityResponseItemAvailability)					
Element	SubElement1	SubElement2	GAWS Equivalent	Example	Customer Information
Revenue Business Unit			szRevenueBusinessUnit	922143	You will use this information for other web services
ItemAvailability[] (of type AvailabilityResponseItemAvailability)					
	BranchPlantDetails[] (of type BranchPlantDetails)				
		AvailableQty	mnAvailableQty	“4518”	This is the available quantity in the default or supplied branch plant. Available quantity is defined as: Quantity On Hand MINUS Quantity Hard-Committed
		BackOrderQty	mnBackOrderQty	Not mapped	This value will be null
		BPType	szBPType	Not mapped	This value will be null
		BranchPlant	szBranchPlant	“110210”	This is the Virtual (Westcon) Location in which the product exists. If no Branch Plant is supplied then the part codes Default Branch Plant is used. This is calculated within the webservice. This is used to return Item Availability
		EffectiveThruDate	jdEffectiveThruDate	“2020/01/01”	This is the date the SBA is effective until.
		Error	mnError	“0”	This is the error code returned by the webservice should an error occur. If this value is not 0, you should expect an error message in the ErrorMessage variable
		ErrorMessage	szErrorMessage	“Product: 23sdads3 returned an error from WS: Invalid Item or Branch Plant not Set Up in BPM”	This is the textual message associated with Error variable
		ETADate	jdETADate	“2008/07/09”	ETA Date on the item

Availability Response Contains an ItemAvailability[] array (of type AvailabilityResponseItemAvailability)					
Element	SubElement1	SubElement2	GAWS Equivalent	Example	Customer Information
		ItemRestriction	mnItemRestriction	“0”	This indicates whether the item is restricted or not. 0=Item NOT Restricted, 1 =Item restricted
		OnHandQty	mnOnHandQty	Not mapped	This variable will appear as null
		OpenPOQty	mnOpenPOQty	“132”	This is the amount of stock on all inbound purchase orders that has not been allocated to sales orders in the COMPASS ERP system
		OrderQty	mnOrderQty	“12”	This is the quantity ordered (pass through from request – see OrderQty)
		outListPrice	mnoutListPrice	“400”	This is the local list price of the item.
		outLLP1	mnoutLLP1	Not mapped	This variable will appear as null
		outLLP2	mnoutLLP2	Not mapped	This variable will appear as null
		outLPDesc1	szoutLPDesc1	Not mapped	This variable will appear as null
		outLPDesc2	szoutLPDesc2	Not mapped	This variable will appear as null
		outLPDspCurcd1	outLPDspCurcd1	“USD”	This is the Display currency code. This is the currency code in which the customer wants the product displayed.
		outLPDspCurcd2		Not mapped	This variable will appear as null
		outLPDspExRate1		Not mapped	This variable will appear as null
		outLPDspExRate2		Not mapped	This variable will appear as null
		outLPExRate		Not mapped	This variable will appear as null
		outLPLedg1		Not mapped	This variable will appear as null
		outLPLedg2		Not mapped	This variable will appear as null
		outLPMultiplier1		Not mapped	This variable will appear as null
		outLPMultiplier2		Not mapped	This variable will appear as null
		PODetail[] (of type BranchPlantDetailsPODetail)		Not mapped	-
		RevenueBusinessUnit	szRevenueBusinessUnit	“922143”	Revenue Business unit on the line item for this part. This will be/might be used for other transactions.
		SalesPerson1	mnSalesPerson1	Not mapped	
		SalesPerson2	mnSalesPerson2	Not mapped	
		SBACat1	szSBACat1	Not mapped	

Availability Response Contains an ItemAvailability[] array (of type AvailabilityResponseItemAvailability)					
Element	SubElement1	SubElement2	GAWS Equivalent	Example	Customer Information
		SBADescription	szSBADescription	Not mapped	
		SBANumber	szSBANumber	Not mapped	
		StockingType	szStockingType	“S”	This indicates whether an item is stocked or not. This is at the branch plant level. Examples of Stocking type are as follows: S=Stock Item (Physical Item) N=Non-Stock Item (Software) D=Drop ship (Physical Item not held in inventory) NOTE: Although this looks similar to szLineType (see Table 1), an items line type for the purposes of the availability call indicates whether the item is a physical item that can be held/stocked in inventory. Examples of line types are as follows: S=Stockable Item N=Non-Stockable Item F=Freight If a branch plant is supplied this will be the stocking type for that item in that branch plant. If no branch plant is supplied then this will be the stocking type for that item in the default branch plant for that item.
	ItemNumber		szItemNumber	“Wic-1t=”	This is the product (the Stock Keeping Unit name – SKU) – as held in Westcon’s Compass ERP system as well as in the catalog you have received.

Availability Response Notes:

- PODetails are not mapped and not requested in the request by default. (even if the user requests this information by updating PODetailFlag to “1”)

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.