

Westcon Web Services Developer's Guide - Pricing

Revision History			
Document Version	Publication Date	Author	Record Description
1.0		Sedat Behar	Start
1.1		Sedat Behar / Elisa Blackman	Pricing Request and Response tables and overall modifications for Pricing
1.2	10/22/2008	Elisa Blackman	Production Re vision

Contents

Westcon Web Services	1
Developer's Guide - Pricing	1
Westcon Web Services INFO TABLE: Your Specific Implementation Details:.....	3
Introduction.....	4
Anatomy of the Web Service.....	6
Security	7
Configure the Client for Security.....	8
Pricing.....	10
1.1 Pricing Sample Code.....	10
1.2 Pricing Request	14
1.3 Pricing Response.....	15
SOAP failures and Exceptions.....	17

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.

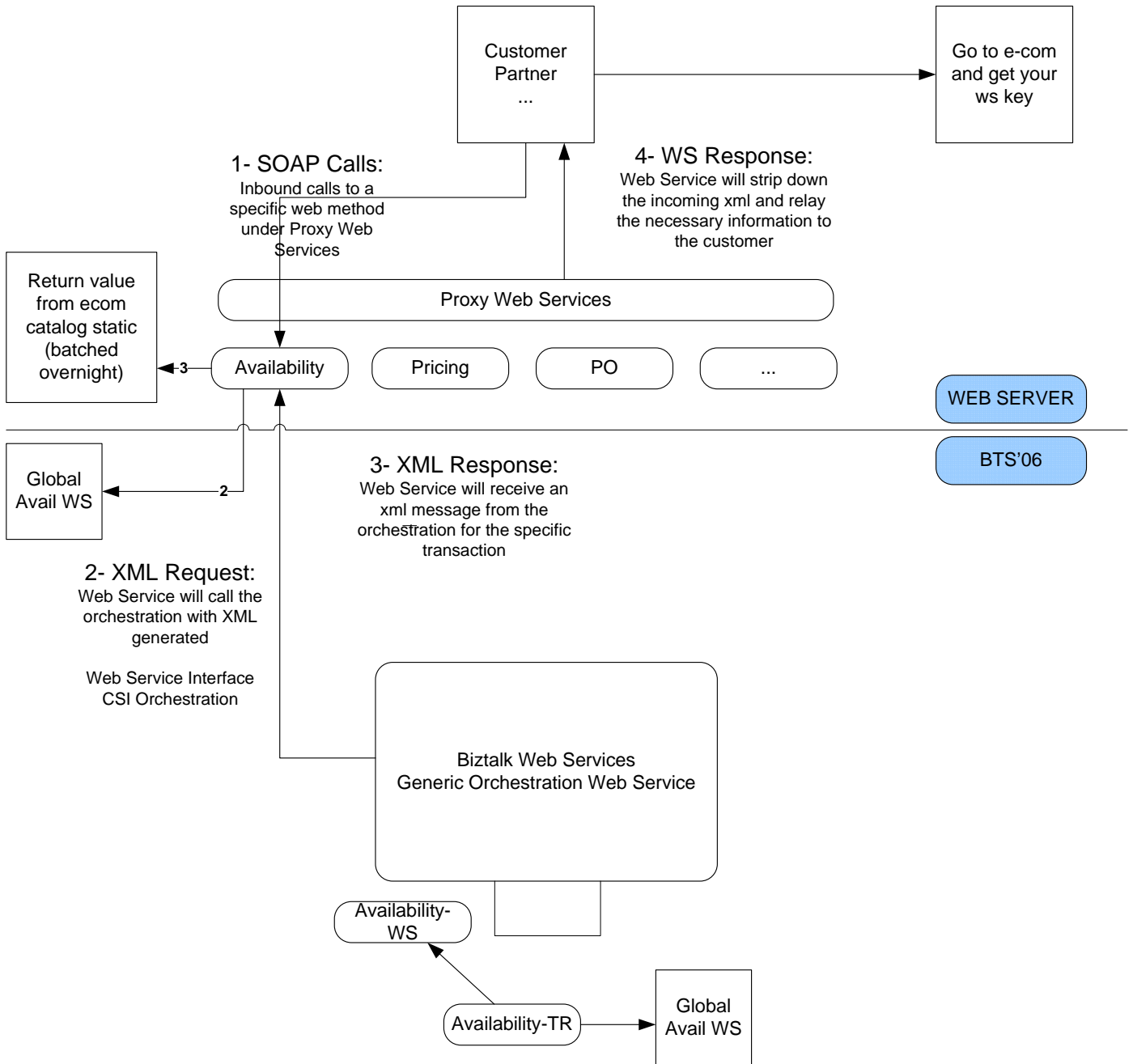
This document provides detailed instructions for using Westcon Web Services. These services will help you obtain pricing/availability/catalog related details as well as submitting a Purchase Order. At the end of the day, we would like you to run the Catalog Web Service and receive your custom catalog,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

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/pol icy">
  <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">
    <usernameOverTransportSecuri ty />
    <requi reActi onHeader />
  </pol icy>
</pol icies>
```

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 "westconPricing," your WSE 3.0–enabled Web service proxy class name would be "westconPricingWSE."

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)

Pricing

For items found in the catalog, you can call the pricing function to get the advanced pricing specific to the customer.

Below are the objects required for the pricing call: (if you called your web reference pricingDirect)

Proxy	pricingDirect.PricingWse proxyWSE = new pricingDirect.PricingWse();
Request	pricingDirect.PricingInboundRequest req = new pricingDirect.PricingInboundRequest();
Response	pricingDirect.PricingOutboundResponsePricingOBHeader[] res;

Please find the Pricing Request and Response fields below with examples, matching Biztalk Internal XML Elements and detailed info. Required fields must be filled in, in order to get customized pricing.

For multiple item queries, make sure that you include an entry in the **ItemNumbers (array of type PricingInboundRequestPricingIBDetailItemNumbers)** array for each Item you are querying for. (Maximum 10 items)

1.1 Pricing Sample Code

Here is sample call for calling the pricing web service: (this is provided in C#, .net 2.0)

```
// Calls the pricing transaction via new proxy to BizTalk
public void testPricingViaBizTalk()
{
    // 1- Create proxy
    pricingDirect.PricingWse proxyWSE = new pricingDirect.PricingWse();

    // Authentication Mechanism:
    // Sent in plainText since the transport channel will be secure SSL
    String userName = "use your values here";
    String password = " use your values here ";
    UsernameToken token = new UsernameToken(userName, password,
PasswordOption.SendPlainText);

    // Attach security token to the proxy
    proxyWSE.SetClientCredential(token);
    proxyWSE.SetPolicy("usernameTokenSecurity");

    // Create Empty Response Object
    pricingDirect.PricingOutboundResponsePricingOBHeader[] res;

    // Build the request
    pricingDirect.PricingInboundRequest req = new pricingDirect.PricingInboundRequest();
    pricingDirect.PricingInboundRequestPricingIBDetail i bdet = new
pricingDirect.PricingInboundRequestPricingIBDetail ();

    req.PricingIBDetail = i bdet;
```

```

req.PricingIBDetail.Company = " use your values here ";
req.PricingIBDetail.CostCenterHeader = use your values here; // "";
req.PricingIBDetail.CustomerAddressNumber = use your values here; // "116325";
req.PricingIBDetail.CustomerCurrencyCode = use your values here; // "USD";
req.PricingIBDetail.EndUserVertical = use your values here; // "COM";

// create new sub-element for pricing and
// include it in the ItemNumbers array as new entry
pricingDirect.PricingInboundRequestPricingIBDetailItemNumbers[] itemNum = new
pricingDirect.PricingInboundRequestPricingIBDetailItemNumbers[1];
pricingDirect.PricingInboundRequestPricingIBDetailItemNumbers itemNumDet = new
pricingDirect.PricingInboundRequestPricingIBDetailItemNumbers();

itemNumDet.BranchPlant = use your values here;
itemNumDet.QtyOrdered = use your values here; // 1;
itemNumDet.SecondItemNumber = use your values here; // "wic-1t"; // string
itemNum[0] = itemNumDet;

req.PricingIBDetail.ItemNumbers = itemNum;

// Make the web service call
try
{
    res = proxyWSE.itemPricing(req);

    if (res != null)
    {
        foreach (pricingDirect.PricingOutboundResponsePricingOBHeader pobh in res)
        {
            lbRes.Text += "<b>Return base price: </b>" + pobh.BasePrice.ToString() +
"<br>";

            // TODO
            lbRes.Text += "<b>BasePrice: </b>" + pobh.BasePrice + "<br>";
// double
            lbRes.Text += "<b>BasePriceSpecified: </b>" + pobh.BasePriceSpecified +
"<br>";
// bool
            lbRes.Text += "<b>BranchPlant: </b>" + pobh.BranchPlant + "<br>";
// string
            lbRes.Text += "<b>CustomerAddressNumber: </b>" +
pobh.CustomerAddressNumber + "<br>"; // double
            lbRes.Text += "<b>CustomerAddressNumberSpecified: </b>" +
pobh.CustomerAddressNumberSpecified + "<br>"; // bool
            lbRes.Text += "<b>CustomerCurrencyCode: </b>" + pobh.CustomerCurrencyCode
+ "<br>"; // string
            lbRes.Text += "<b>EndUserVertical: </b>" + pobh.EndUserVertical + "<br>";
// string
            lbRes.Text += "<b>ErrorMessage: </b>" + pobh.ErrorMessage + "<br>";
// string
            lbRes.Text += "<b>ListPrice: </b>" + pobh.ListPrice + "<br>";
// double
            lbRes.Text += "<b>ListPriceSpecified: </b>" + pobh.ListPriceSpecified +
"<br>"; // bool

            // complex element
            foreach
(pricingDirect.ArrayOfPricingOutboundResponsePricingOBHeaderSBInfoSBInfo sbainfo in
pobh.PricingOBDetail)
            {
                // lbRes.Text += "<b> : </b>" + res[0].PricingOBDetail + "<br>";
// pricingDirect.ArrayOfPricingOutboundResponsePricingOBHeaderSBInfoSBInfo
                lbRes.Text += "<b> sbainfo.EffectiveThruDate: </b>" +
sbainfo.EffectiveThruDate + "<br>"; //string
                lbRes.Text += "<b>sbainfo.ListPriceType: </b>" +
sbainfo.ListPriceType + "<br>"; //string
                lbRes.Text += "<b>sbainfo.SBACat3: </b>" + sbainfo.SBACat3 + "<br>";
//string
                lbRes.Text += "<b>sbainfo.SBADescription: </b>" +
sbainfo.SBADescription + "<br>"; //string
            }
        }
    }
}

```

```

        lbRes.Text += "<b>sbainfo.SBALongDescription: </b>" +
sbainfo.SBALongDescription + "<br>"; //string
        lbRes.Text += "<b>sbainfo.SBANumber: </b>" + sbainfo.SBANumber +
"<br>"; //string
        lbRes.Text += "<b>sbainfo.SBAPriceAmount: </b>" +
sbainfo.SBAPriceAmount + "<br>"; //double
        lbRes.Text += "<b>sbainfo.SBAPriceAmountSpecified: </b>" +
sbainfo.SBAPriceAmountSpecified + "<br>"; //bool
        lbRes.Text += "<b>sbainfo.SBAPriceDiscountPercentage: </b>" +
sbainfo.SBAPriceDiscountPercentage + "<br>"; //double
        lbRes.Text += "<b>sbainfo.SBAPriceDiscountPercentageSpecified: </b>"
+ sbainfo.SBAPriceDiscountPercentageSpecified + "<br>"; //bool
        lbRes.Text += "<b>sbainfo.SBATYPE: </b>" + sbainfo.SBATYPE + "<br>";
//string
    }

    lbRes.Text += "<b>SecondItemNumber: </b>" + pobh.SecondItemNumber +
"<br>"; // string
    lbRes.Text += "<b>ShortItemNumber: </b>" + pobh.ShortItemNumber + "<br>";
// string
    }
    lbRes.Text += "<p>";
}
else
{
    lbRes.Text += "Problem with the pricing Call (SOAP object returned, but no
pricing available) <p>";
}

}
catch (Exception ex)
{
    lbRes.Text += "Error occured calling webservice <b>" + ex.ToString() + "</b>";
}

finally
{
    // TODO: Cleanup and logging here...
}
}

```

1.2 Code for 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 as well. This allows all WS-Security 1.0 conforming clients to make secure connections and implement our web services.

Pricing Request

```

<?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:wss="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/itemPricing</wsa:Action>
<wsa:MessageID>urn:uuid:9f75952c-2575-4974-a60a-d8fd00a91b0d</wsa:MessageID>
<wsa:ReplyTo><wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:Address></wsa:ReplyTo>
<wsa:To>http://webservices.westcon.com/pricing.asmx</wsa:To>
<wsse:Security soap:mustUnderstand="1"><wsu:Timestamp wsu:Id="Timestamp-6826727b-0540-4fa8-831d-6a7d36412e92"><wsu:Created>2008-11-11T21:02:29Z</wsu:Created>
<wsu:Expires>2008-11-11T21:07:29Z</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-42936715-debc-4c37-97e3-6d2480754795">
<wsse:Username>123456789</wsse:Username>
<wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText">xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx</wsse:Password>
<wsse:Nonce>v0zMMhaWz/AUxU9RGeEFVg==</wsse:Nonce><wsu:Created>2008-11-11T21:02:29Z</wsu:Created></wsse:UsernameToken>
</wsse:Security></soap:Header>

<soap:Body><itemPricing xmlns="http://webservices.westcon.com"><request><PricingIBDetail xmlns=""><Company>00011</Company>
<CustomerAddressNumber>1107922</CustomerAddressNumber><CustomerCurrencyCode>USD</CustomerCurrencyCode>
<EndUserVertical>COM</EndUserVertical><CostCenterHeader>119992</CostCenterHeader>
<ItemNumbers><SecondItemNumber>wic-1t</SecondItemNumber><BranchPlant>110210</BranchPlant>
<QtyOrdered>1</QtyOrdered></ItemNumbers></PricingIBDetail></request></itemPricing></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.

1.3 Pricing Request

Pricing Request:				Array	Required Field
L1	L2	L3	Example	Info for Customer	Data Type
PricingIBDetail (array of type PricingInboundReq uestPricingIBDetail ItemNumbers)	Company		00011	Please use the company variable in your Info Table	string
	CustomerAddressNumber		116325	Please use the Account (username) variable in your Info Table	string
	CustomerCurrencyCode		USD	Please enter the 3-character currency code for your transaction (USD, EUR, ...)	string
	CostCenterHeader		119992	Please use the RBU / HRBU variable in your Info Table	string
	EndUserVertical		COM (Commercial)	Use three letter vertical codes (e.g.: COM, GOV, EDU,...)	string
	ItemNumbers (array of type PricingInboundRequestPricing IBDetailItemNumbers)	BranchPlant	110210	Do not initialize. This is for potential future use. You can refer to the BranchPlant column in the catalog for this variable.	string
		QtyOrdered	100	Quantity Required	int
		SecondItemNumber	“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.	string
		ShortItemNumber		DO NOT SUPPLY or INITIATE THIS FIELD	string

1.4 Pricing Response

Pricing Response					Array
L1	L2	L3	Example	Info for Customer	Data Type
PricingOutboundResponsePricingOBHeader (this is an array)	BasePrice		125	This is the calculated price at which the customer is offered the item. This will be in the currency in which the customer requested the transaction	double
	BasePriceSpecified		TRUE	Returns whether the base price is specified	bool
	BranchPlant		110110	Returns the branch plant (as supplied)	string
	CustomerAddressNumber		116325	Returns your account number (as supplied)	double
	CustomerAddressNumberSpecified		TRUE	Returns whether the account number is specified	bool
	CustomerCurrencyCode		USD	3-letter currency code used	string
	EndUserVertical		COM	3-letter vertical code used	string
	ErrorMessage			Contains the error message, if any, from the system.	string
	ListPrice		150.00	This is the List Price of the item in the transaction currency. The transaction currency is the currency that the sale will be made should the item be included in a sales request.	double
	ListPriceSpecified		TRUE	Returns whether the list price is specified	bool
	PricingOBDetail(array of type ArrayOfPricingOutboundResponsePricingOBHeader)	EffectiveThruDate	1/1/2010	This is the date the SBA is effective until.	string

Pricing Response					Array
L1	L2	L3	Example	Info for Customer	Data Type
	erSBAInfoSBAInfo)				
		ListPriceType	LP	List Price Type	string
		SBACat3	D17	SBA Category	string
		SBADescription	This SBA applied to all Small Office/Home Office (SOHO) Routers	This is a short description of the SBA	string
		SBALongDescription		This is a long description of the SBA	string
		SBANumber	SBA00001	This is the Code of the SBA. This is stipulated by the vendor of the item	string
		SBAPriceAmount	116	This is the price of the item once the SBA has been applied. The calculation is mnBasePrice minus the SBA discount	double
		SBAPriceAmountSpecified		Returns whether the SBA Price Amount is specified	bool
		SBAPriceDiscountPercentage	17	SBA Discount percentage	double
		SBAPriceDiscountPercentageSpecified		Returns whether the SBA Discount Percentage is specified	bool
		SBAType	4	SBA Type	string
	SecondItemNumber		WIC-1T	Item Serial Number	string
	ShortItemNumber			you will see this field as blank	string

SOAP failures and Exceptions

Below is what the exceptions will look like.

SOAP Failures – Errors Returned:					
Business Default Description	Business Fault	Exception Type	Possible Causes	Possible Fixes	Knowledge Base
Invalid response received from IVAL web service	20116	System.Exception			
Validation Exception has Occurred	10001	System.Exception	Message didn't match message schema	Analyze issue in unprocessed BAM view, manually repair source file and resubmit	

In many cases, the error that you might expect to see will be an internal error. We are in the process of improving our Customer-facing error handling procedures.

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.