



**RESELLONE.NET**

# **ResellOne.net Provisioning System (RPS) XML Over HTTPS Post**

An Addendum to the ResellOne.net API Specifications

February, 2006

# Table of Contents

Using HTTPS Post to Access The ResellOne.net Provisioning System .....	3
Connection Information.....	3
Live Production Environment.....	3
OT&E Test Environment .....	<b>Error! Bookmark not defined.</b>
MD5 Authentication.....	4
PERL.....	4
PHP .....	4
VB6.....	4
VB .NET .....	5
JAVA .....	5
HTTPS Post Examples .....	6
PERL.....	6
PHP .....	9
VB6.....	11
VB .NET .....	15
JAVA .....	16
Troubleshooting .....	21
If you're getting a "401 Authentication Error" .....	21
If you're getting an "Invalid XML" response.....	21
If requests in ASP pages timeout and fail.....	22

# Using HTTPS Post to Access the ResellOne.net Provisioning System

The ResellOne.net Provisioning System (RPS) server now supports SSL encryption and will handle XML posts from ResellOne.net authorized resellers. The RPS server:

1. Applies IP address authentication to all HTTPS POST requests.
2. Authenticates the user by verifying the username and MD5 signature of the XML, signed using the resellers private key (generated in the RWI) in the request header.

ResellOne.net selected to post XML, as it is a more structured and manageable approach than using "Name Value Pairs".

## Connection Information Live Production Environment

**Server:** <https://resellers.resellone.net>

**Port:** 52443

# MD5 Authentication

The MD5 Signature provides the authentication required by ResellOne.net. The process involves two steps:

1. Obtain an MD5 signature of the XML Content and the Private Key.  
Note: The XML and the Private Key are concatenated.
2. Perform another MD5 of the signature from STEP 1 with the Private Key. Note: The MD5 Signature from Step 1 and the Private key are concatenated.

## PERL

```
use Digest::MD5 qw/md5_hex/;
md5_hex(md5_hex($xml, $private_key), $private_key)
```

## PHP

```
md5(md5($xml.$private_key).$private_key);
```

## VB6

Using the "di\_MD5DLL.dll" from DI Management

This free cryptographic software code was written or adapted in Visual Basic and ANSI C by David Ireland.

<http://www.di-mgt.com.au/crypto.html#MD5>

```
Private Declare Function MakeMD5Digest Lib "di_MD5DLL.dll" _
    (ByVal sData As String, ByVal sDigest As String) As Long
```

```
Public Function MD5Encode(ByVal sData As String) As String
    Dim iRet As Long
    Dim sDigest As String
    ' Set sDigest to be 32 chars
    sDigest = String(32, " ")
    iRet = MakeMD5Digest(sData, sDigest)
    MD5Encode = Trim(sDigest)
End Function
```

```
MD5Encode(MD5Encode(pXMLDoc.xml & strPrivateKey) & strPrivateKey)
```

## VB .NET

```
Public Function cMD5(ByVal str As String) As String
    'Must have Imports System.Web.Security in General Declarations
    Dim Hash As String =
FormsAuthentication.HashPasswordForStoringInConfigFile(str, "MD5")
    Return Hash.ToLower
End Function

cMD5(cMD5(str & PRIVATE_KEY) & PRIVATE_KEY))
```

## JAVA

```
public String getSignature(String xml) {
    String signature = new String();
    try {
        MessageDigest md5 = MessageDigest.getInstance("MD5");
        String concat = x\ml + privateKey;
        concat = bytesToHex(md5.digest(concat.getBytes()));
        signature =
bytesToHex(md5.digest(concat.concat(privateKey).getBytes()));
    }
    catch (Exception ex) {
    }
    return signature;
}

protected String bytesToHex(byte[] bytes) {
    String s = new String();
    BigInteger bi = new BigInteger(1,bytes);
    // Format to hexadecimal
    s = bi.toString(16); // 120ff0
    if (s.length() % 2 != 0) {
        // Pad with 0
        s = "0" + s;
    }
    return s;
}
```

# HTTPS Post Examples

## PERL

```
#!/usr/bin/perl
#####
#Configuration Variables
my $rspusername = 'rspusername';
my $private_key='privatekey';
my $REMOTE_HOST = " resellers.resellone.net:52443";
#Path to ResellOne.net Client Packages for XML Parsing
my $PATH_TO_FILES = "Enter Path";
#####

## global defines
use vars qw(%in $cgi);
( %in, $cgi) = ();

use CGI ':cgi-lib';
use HTTP::Request::Common qw(POST);
use HTTP::Request::Common;
use Digest::MD5 qw/md5_hex/;
use LWP::UserAgent;
use Data::Dumper;
#Using the following two lines ResellOne.net Client Packages for XML
Parsing
require "$PATH_TO_FILE]/OPS.pm";
#You might have to replace the 'use XML_Codec' in OPS.pm with the
following line as well.
require "$PATH_TO_FILES/XML_Codec.pm";

$ua = LWP::UserAgent->new;
my $ops = new OPS;

$cgi = $ENV{SCRIPT_NAME};
%in = ();
print "Content-type:  text/html\n\n";

# read in the form data
ReadParse(\%in);

my $domain = $in{domain};
my $username = $in{username};
my $password = $in{password};
my $type = $in{type};

my $cookie = create_cookie($domain,$username,$password);
#Create XML
```

```

my $xml = "<?xml version='1.0' encoding='UTF-8' standalone='no' ?>
<!DOCTYPE OPS_envelope SYSTEM 'ops.dtd'>
<OPS_envelope>
  <header>
    <version>0.9</version>
  </header>
  <body>
    <data_block>
      <dt_assoc>
        <item key='object'>DOMAIN</item>
        <item key='attributes'>
          <dt_assoc>
            <item key='type'>$type</item>
          </dt_assoc>
        </item>
        <item key='cookie'>$cookie</item>
        <item key='protocol'>XCP</item>
        <item key='action'>get</item>
      </dt_assoc>
    </data_block>
  </body>
</OPS_envelope>";

```

```

my $request = POST (
  "https://$REMOTE_HOST",
  'Content-Type' => 'text/xml',
  'X-Username'   => $rspusername,
  'X-Signature' => md5_hex(md5_hex($xml,
$private_key), $private_key),
  'Content'      => $xml);
my $response = $ua -> request($request);

```

```

print Dumper($response->{'_content'});

```

```

sub create_cookie {
my ($domain, $username, $password) = @_;
my $xml = "<?xml version='1.0' encoding='UTF-8' standalone='no' ?>
<!DOCTYPE OPS_envelope SYSTEM 'ops.dtd'>
<OPS_envelope>
  <header>
    <version>0.9</version>
  </header>
  <body>
    <data_block>
      <dt_assoc>
        <item key='attributes'>
          <dt_assoc>
            <item key='reg_password'>$password</item>
            <item key='domain'>$domain</item>
            <item key='reg_username'>$username</item>
          </dt_assoc>
        </item>
      </dt_assoc>
    </data_block>
  </body>
</OPS_envelope>";

```

```

        </item>
        <item key='object'>cookie</item>
        <item key='action'>set</item>
        <item key='protocol'>XCP</item>
    </dt_assoc>
</data_block>
</body>

</OPS_envelope>";

my $request = POST (
    "https://$REMOTE_HOST",
    'Content-Type' => 'text/xml',
    'X-Username'   => $rspusername,
    'X-Signature' => md5_hex(md5_hex($xml,
$private_key), $private_key),
    'Content'      => $xml);
my $response = $ua -> request($request);
my $xcp = $ops->decode($response->{'_content'});
return ($xcp->{attributes}->{cookie});
}

```

# PHP

```
<html>
<body>
<?php
$username = "RSPUsername";
$private_key = "PrivateKey";

$xml = '<?xml version=\'1.0\' encoding="UTF-8" standalone="no" ?>
<!DOCTYPE OPS_envelope SYSTEM "ops.dtd">
<OPS_envelope>
  <header>
    <version>0.9</version>
  </header>
  <body>
    <data_block>
      <dt_assoc>
        <item key="object">DOMAIN</item>
        <item key="action">LOOKUP</item>
        <item key="protocol">XCP</item>
        <item key="attributes">
          <dt_assoc>
            <item key="domain">example.com</item>
          </dt_assoc>
        </item>
      </dt_assoc>
    </data_block>
  </body>
</OPS_envelope>';

$signature = md5(md5($xml.$private_key).$private_key);
$host = " resellers.resellone.net";
$port = 52443;
$url = "/";
$header .= "POST $url HTTP/1.0\r\n";
$header .= "Content-Type: text/xml\r\n";
$header .= "X-Username: " . $username . "\r\n";
$header .= "X-Signature: " . $signature . "\r\n";
$header .= "Content-Length: " . strlen($xml) . "\r\n\r\n";
# ssl:// requires OpenSSL to be installed
$fp = fsockopen ("ssl://$host", $port, $errno, $errstr, 30);

echo "<pre>";

if (!$fp) {
  print "HTTP ERROR!";
} else {
  # post the data to the server
  fputs ($fp, $header . $xml);
  while (!feof($fp)) {
```

```
        $res = fgets ($fp, 1024);
        echo htmlentities($res);
    }
    fclose ($fp);
}

echo "</pre>";
?>
</body>
```

## VB6

Special Thanks to Serguei Seleznev from Softcom Technology for developing this script.

This Script is using di\_MD5DLL.dll from DI Management. This free cryptographic software code that David Ireland has written or adapted in Visual Basic and ANSI C. <http://www.di-mgt.com.au/crypto.html#MD5>

```
Dim ErrNumber As Long
Dim ErrDescription As String
Const strUserName = "RSP_USERNAME"
Const strPrivateKey = "Your Private Key Go's here"
Const strURL = "https://resellers.resellone.net:52443"

Private Declare Function MakeMD5Digest Lib "di_MD5DLL.dll" _
    (ByVal sData As String, ByVal sDigest As String) As Long

Sub Main()
' POST XML document using VB6 and MSXML4(has to be installed)

    Dim DocToSend As MSXML2.DOMDocument
    Dim pFileRequest As String
    Dim strXML As String

On Error GoTo err_Main
    pFileRequest = App.Path & "\Sample.XML"
    strXML = ""

    Set DocToSend = New MSXML2.DOMDocument
    If Not ReadXMLDocument(pFileRequest, DocToSend) Then
        MsgBox "Cannot read " & pFileRequest & vbCrLf & _
            ErrNumber & ", " & ErrDescription, vbCritical
        Exit Sub
    End If

    If SendRequestXML(DocToSend, strXML, strURL) Then
        MsgBox "Response has come. " & strXML
        ' here you may save to file or reload in DOMdocument to parse
    Else
        MsgBox "Cannot send " & DocToSend.xml & vbCrLf & _
            ErrNumber & ", " & ErrDescription, vbCritical
        Exit Sub
    End If
    Set DocToSend = Nothing
    Exit Sub
err_Main:
    ErrNumber = Err.Number
    ErrDescription = "Run-time ERROR in Main. " & Err.Description
```

```

        MsgBox "Error " & ErrNumber & ", " & ErrDescription, vbCritical

End Sub

Private Function SendRequestXML( _
    ByRef pXMLDoc As MSXML2.DOMDocument, _
    ByRef pcTmp As String, _
    ByVal pstrURL As String _
) As Boolean

    Dim xmlHttp As MSXML2.ServerXMLHTTP40

On Error GoTo err_SendRequestXML
    SendRequestXML = False

    Set xmlHttp = New MSXML2.ServerXMLHTTP40
    xmlHttp.Open "POST", pstrURL, False ' False - synchronous mode
    xmlHttp.setRequestHeader "Content-Type", "text/xml"
    xmlHttp.setRequestHeader "X-Username", strUserName
    xmlHttp.setRequestHeader "X-Signature",
MD5Encode(MD5Encode(pXMLDoc.xml & strPrivateKey) & strPrivateKey)
    xmlHttp.send pXMLDoc.xml

    pcTmp = xmlHttp.responseXML.xml
    Set xmlHttp = Nothing
    SendRequestXML = True

''     'You may try to use asynchronous post
''     xmlHttp.Open "POST", pstrURL, True
''     xmlHttp.setRequestHeader "Content-Type", "text/xml"
''     xmlHttp.setRequestHeader "X-Username", strUserName
''     xmlHttp.setRequestHeader "X-Signature",
MD5Encode(MD5Encode(pXMLDoc.xml & strPrivateKey) & strPrivateKey)
''     xmlHttp.send pXMLDoc.xml
''     PauseSeconds 1                               'wait
''     pcTmp = ""
''     If xmlHttp.readyState = 4 Then               'we got it
''         If xmlHttp.Status = 200 Then
''             pcTmp = xmlHttp.responseXML.xml
''             SendRequestXML = True
''         End If
''     Else                                         ' let's wait for a while
''         PauseSeconds 3
''         If xmlHttp.readyState = 4 Then          ' check again
''             If xmlHttp.Status = 200 Then
''                 cTmp = xmlHttp.responseXML.xml
''                 SendRequestXML = True
''             End If
''         End If
''     End If

Exit Function

```

```

err_SendRequestXML:
    If IsObject(xmlHttp) Then Set xmlHttp = Nothing

    ErrNumber = Err.Number
    ErrDescription = "Run-time ERROR in SendRequestXML. " &
Err.Description

    Err.Clear
End Function

Private Function ReadXMLDocument(ByVal pDocName As String, ByRef
pXMLDoc As MSXML2.DOMDocument) As Boolean

    On Error GoTo err_ReadXMLDocument

    ReadXMLDocument = False

    pXMLDoc.async = False
    pXMLDoc.resolveExternals = False      ' otherwise you must have a
referred DTD
    pXMLDoc.validateOnParse = False      ' in a same directory
    pXMLDoc.Load pDocName

    If pXMLDoc.parseError.errorCode = 0 Then
        ReadXMLDocument = True
    Else
        ErrNumber = pXMLDoc.parseError.errorCode
        ErrDescription = "Errors in " & pXMLDoc.parseError.url & ",
line " & pXMLDoc.parseError.Line & ", pos " &
pXMLDoc.parseError.linepos
        ErrDescription = ErrDescription & ". Error #" & ErrNumber & ".
" & pXMLDoc.parseError.reason
    End If

Exit Function
err_ReadXMLDocument:
    ErrNumber = Err.Number
    ErrDescription = "Run-time ERROR in ReadXMLDocument " &
Err.Description & " for " & pDocName
    Err.Clear
End Function

' contents of file Sample.XML

'<?xml version="1.0" encoding="UTF-8" standalone="no"?>
'<!DOCTYPE OPS_envelope SYSTEM "ops.dtd">
'<OPS_envelope>
'|   <header>
'|       <version>0.9</version>
'|   </header>

```

```
'
  <body>
  <data_block>
    <dt_assoc>
      <item key="object">DOMAIN</item>
      <item key="attributes">
        <dt_assoc>
          <item key="domain">MyDomainToLookup.com</item>
        </dt_assoc>
      </item>
      <item key="protocol">XCP</item>
      <item key="action">LOOKUP</item>
    </dt_assoc>
  </data_block>
</body>
'</OPS_envelope>
```

```
Public Function MD5Encode(ByVal sData As String) As String
  Dim iRet As Long
  Dim sDigest As String
  ' Set sDigest to be 32 chars
  sDigest = String(32, " ")
  iRet = MakeMD5Digest(sData, sDigest)
  MD5Encode = Trim(sDigest)
End Function
```

## VB .NET

```
Dim mypost As New ResellOne.net_XMLPOST
Dim RESPONSE_TEXT As String = "<?xml version='1.0' encoding='UTF-8' standalone='no' ?><!DOCTYPE OPS_envelope SYSTEM 'ops.dtd'><OPS_envelope><header><version>0.9</version></header><body><data_block><dt_assoc><item key='protocol'>XCP</item><item key='action'>LOOKUP</item><item key='object'>DOMAIN</item><item key='attributes'><dt_assoc><item key='domain'>" & TextBox2.Text & "</item></dt_assoc></item></dt_assoc></data_block></body></OPS_envelope>"

    TextBox1.Text = mypost.sendPost(RESPONSE_TEXT)

-----
Imports System.Web.Security
Public Class ResellOne.net_XMLPOST

    Public Const URL_BASE = "https://resellers.resellone.net:52443"
    Public Const RSP_USERNAME As String = "RSP USERNAME"
    Public Const PRIVATE_KEY = "Enter Private Key"

    Public Function sendPost(ByVal str As String)
        Dim myHttpRequest As New System.Net.WebClient
        myHttpRequest.Headers.Add("Content-Type", "text/xml")
        myHttpRequest.Headers.Add("X-Username", RSP_USERNAME)
        myHttpRequest.Headers.Add("X-Signature", cMD5(cMD5(str & PRIVATE_KEY) & PRIVATE_KEY))

        Dim sendData As Byte() =
System.Text.Encoding.ASCII.GetBytes(str)
        Dim myHttpResponse As Byte() =
myHttpRequest.UploadData(URL_BASE, "POST", sendData)

        Return System.Text.Encoding.ASCII.GetString(myHttpResponse)
    End Function

    'Used to convert to MD5
    Public Function cMD5(ByVal str As String) As String
        'Must have Imports System.Web.Security in General Declarations
        Dim Hash As String =
FormsAuthentication.HashPasswordForStoringInConfigFile(str, "MD5")
        Return Hash.ToLower
    End Function
End Class
```

# JAVA

```
package net.client;

import javax.net.ssl.*;
import javax.net.SocketFactory;
import java.net.*;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.IOException;
import java.security.MessageDigest;
import java.util.Hashtable;
import java.math.BigInteger;

import org.apache.commons.httpclient.*;
import org.apache.commons.httpclient.methods.PostMethod;
import org.apache.commons.httpclient.protocol.*;
import com.pureload.task.api.TaskExecuteException;

public class SslClient {
    private String privateKey;
    private String host;
    private int port;
    private String userName;
    private Header [] headers = null;

    public class MySSLSocketFactory implements SecureProtocolSocketFactory {
        private TrustManager[] getTrustManager() {
            TrustManager[] trustAllCerts = new TrustManager[] {
                new X509TrustManager() {
                    public java.security.cert.X509Certificate[] getAcceptedIssuers()
                    {
                        return null;
                    }

                    public void checkClientTrusted(
                        java.security.cert.X509Certificate[] certs, String
authType) {
                    }

                    public void checkServerTrusted(
                        java.security.cert.X509Certificate[] certs, String
authType) {
                    }
                }
            };
            return trustAllCerts;
        }

        public Socket createSocket(String host, int port) throws IOException,
UnknownHostException {
            TrustManager[] trustAllCerts = getTrustManager();
            try {
                SSLContext sc = SSLContext.getInstance("SSL");
                sc.init(null, trustAllCerts, new java.security.SecureRandom());
            }
        }
    }
}
```

```

        HttpURLConnection.setDefaultSSLConnectionFactory(sc.getSocketFactory());
        SocketFactory socketFactory =
HttpURLConnection.getDefaultSSLConnectionFactory();
        return socketFactory.createSocket(host, port);
    }
    catch (Exception ex) {
        throw new UnknownHostException("Problems to connect " + host +
ex.toString());
    }
}

    public Socket createSocket(Socket socket, String host, int port, boolean
flag) throws IOException, UnknownHostException {
    TrustManager[] trustAllCerts = getTrustManager();
    try {
        SSLContext sc = SSLContext.getInstance("SSL");
        sc.init(null, trustAllCerts, new java.security.SecureRandom());

        HttpURLConnection.setDefaultSSLConnectionFactory(sc.getSocketFactory());
        SocketFactory socketFactory =
HttpURLConnection.getDefaultSSLConnectionFactory();
        return socketFactory.createSocket(host, port);
    }
    catch (Exception ex) {
        throw new UnknownHostException("Problems to connect " + host +
ex.toString());
    }
}

    public Socket createSocket(String host, int port, InetAddress clientHost,
int clientPort) throws IOException, UnknownHostException {
    TrustManager[] trustAllCerts = getTrustManager();
    try {
        SSLContext sc = SSLContext.getInstance("SSL");
        sc.init(null, trustAllCerts, new java.security.SecureRandom());

        HttpURLConnection.setDefaultSSLConnectionFactory(sc.getSocketFactory());
        SocketFactory socketFactory =
HttpURLConnection.getDefaultSSLConnectionFactory();
        return socketFactory.createSocket(host, port, clientHost,
clientPort);
    }
    catch (Exception ex) {
        throw new UnknownHostException("Problems to connect " + host +
ex.toString());
    }
}

}

    public SslClient(String host, int port, String userName, String
privateKey) {
    this.host=host;
    this.port = port;
    this.userName = userName;
}

```

```

        this.privateKey = privateKey;
    }
    protected String bytesToHex(byte[] bytes) {
        String s = new String();
        BigInteger bi = new BigInteger(1,bytes);
        // Format to hexadecimal
        s = bi.toString(16);           // 120ff0
        if (s.length() % 2 != 0) {
            // Pad with 0
            s = "0" + s;
        }
        return s;
    }
    public String getSignature(String xml) {

        String signature = new String();
        try {
            MessageDigest md5 = MessageDigest.getInstance("MD5");
            String concat = xml + privateKey;
            concat = bytesToHex(md5.digest(concat.getBytes()));
            signature =
bytesToHex(md5.digest(concat.concat(privateKey).getBytes()));
        }
        catch (Exception ex) {
        }
        return signature;
    }

    public String sendRequest(String xml) throws TaskExecuteException {
        HttpClient client = new HttpClient();
        client.setConnectionTimeout(60000);
        client.setTimeout(60000);
        String response = new String();
        String portStr = String.valueOf(port);
        Protocol.registerProtocol("https", new Protocol("https", new
MySSLSocketFactory(), port));
        String signature = getSignature(xml);
        String uri = "https://" + host + ":" + portStr + "/";
        PostMethod postRequest = new PostMethod(uri);
        postRequest.setRequestHeader("Content-Length",
String.valueOf(xml.length()));
        postRequest.setRequestHeader("Content-Type", "text/xml");
        postRequest.setRequestHeader("X-Signature", signature);
        postRequest.setRequestHeader("X-Username", userName);
        postRequest.setRequestBody(xml);

                System.out.println("Sending https request....."+postRequest.
toString());
        try {
            client.executeMethod(postRequest);
        }
        catch (Exception ex) {
            throw new TaskExecuteException("Sending post got exception ", ex);
        }

        response = postRequest.getResponseBodyAsString();
        headers = postRequest.getRequestHeaders();
    }

```

```

    return response;
}

public String getPrivateKey() {
    return privateKey;
}

public void setPrivateKey(String privateKey) {
    this.privateKey = privateKey;
}

public String getHost() {
    return host;
}

public void setHost(String host) {
    this.host = host;
}

public int getPort() {
    return port;
}

public void setPort(int port) {
    this.port = port;
}

public String getUsername() {
    return userName;
}

public void setUsername(String userName) {
    this.userName = userName;
}

public Header[] getHeaders() {
    return headers;
}

public void setHeaders(Header[] headers) {
    this.headers = headers;
}

public static void main(String[] args) {
    String privateKey = "your_private_key";
    String userName = "your_user_name";
    String host="resellers.resellone.net";
    int port = 52443;

    String xml=
"<?xml version='1.0' encoding='UTF-8' standalone='no' ?>"+
"<!DOCTYPE OPS_envelope SYSTEM 'ops.dtd'>"+
"<OPS_envelope>"+
"  <header>"+
"    <version>0.9</version>"+

```

```

    "<msg_id>2.21765911726198</msg_id>" +
    "<msg_type>standard</msg_type>" +
    "</header>" +
    "<body>" +
    "  <data_block>" +
    "    <dt_assoc>" +
    "      <item key='attributes'>" +
    "        <dt_assoc>" +
    "          <item key='domain'>test-1061911771844.com</item>" +
    "          <item key='pre-reg'>0</item>" +
    "        </dt_assoc>" +
    "      </item>" +
    "      <item key='object'>DOMAIN</item>" +
    "      <item key='action'>LOOKUP</item>" +
    "      <item key='protocol'>XCP</item>" +
    "    </dt_assoc>" +
    "  </data_block>" +
    "</body>" +
    "</OPS_envelope>";

    SslClient sslclient = new SslClient(host,port,userName,privateKey);

    try {
        String response = sslclient.sendRequest(xml);
        System.out.println("\nResponse is:\n"+response);
    }
    catch (TaskExecuteException e) {
        e.printStackTrace();
    }
}
}

```

# Troubleshooting

## If you're getting a "401 Authentication Error"

- Check that you are using the correct Private Key for the right system – Horizon or Production.
- Check that you have the correct RSP Username.

### If the above two are correct, the problem is with the MD5:

- Ensure that you have concatenated the XML Content and the Private Key.
- Ensure that you have performed an MD5 twice. See the MD5 section for more information.
- Ensure that your HTTP Post implementation is not adding any extra information. Some implementations of HTTP Post will add a NULL to the end of the HTTP Request. This will reflect in the MD5 and thus cause an authentication error.

### If you are still not connecting properly, check the result of the MD5 Hash:

- Some MD5 algorithms will uppercase the MD5 hash. Make sure the result is in lowercase before sending to ResellOne.net.
- Some MD5 Algorithms need to convert the string to bytes before generating the hash. Make sure this is done properly. You can test your script by performing an MD5 on the following text:

Text:                   ConnecttoResellOne.netviaSSL  
MD5 Result:         e787cc1d1951dfec4827cede7b1a0933

## If you're getting an "Invalid XML" response

Make sure you are sending the CORRECT XML. The XML used in the MD5 is just for authentication purposes. You must also send the XML as part of the request.

Please see the API Specification document for complete XML examples.

## If requests in ASP pages timeout and fail

Microsoft introduced an undocumented security feature in SP2 for MSXML4 which causes any XML over HTTP requests embedded in ASP pages to timeout and fail. This feature breaks code that was working in versions previous to SP2. If a user connects to your servers with an ASP page using MSXML4 and later upgrades to MSXML4 SP2, the page will no longer work.

This can be fixed by adding two registry entries:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\InternetSettings\Zones\3]
```

```
"1601"=dword:00000000
```

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\CurrentVersion\InternetSettings]
```

```
"Security_HKLM_only"=dword:00000001
```