Alliance Payment Technologies, Inc.
XML Requester Service
Sample VBScript

Version: 1.0
Monday, October 06, 2003

For support, please contact:
support@allianceach.com
(909) 974-0100

Table of Contents

Accessing Alliances XML Requester services from VBScript on the Windows platform is best done using the objects provided by MSXML. Whether from within a browser window, an ASP page, or a stand-alone VBScript program running within Windows Scripting Host, MSXML takes care of all the hard work, freeing the developer to concentrate on the business problem that needs to be solved.

Supporting Code

The supporting code for this document can be downloaded at:
http://www.allianceach.com/downloads/

Creating a Request

A service request is built using an instance of XMLDOMDocument. The newly created DOMDocument is then populated with the elements that make up an Alliance XML Requester service request.

```
'
'Create an empty XMLDOMDocument
'
dim doc
set doc = CreateObject(strMSXMLProgID)

'
'Create a new <AACHRequest> element and add it to the DOM (it's the document
element).
'
dim root
set root = doc.appendChild(doc.createElement("AACHRequest"))
```

Once we've created an empty AACHRequest element, we need to add authentication information to it.

```
'
'Create an <Authentication> element and append it to the document element
'
dim auth
set auth = root.appendChild(doc.createElement("Authentication"))

'
'Create <Username> and <Password> elements and append them to the <Authentication>
'
auth.appendChild(doc.createElement("Username")).text = strUser
auth.appendChild(doc.createElement("Password")).text = strPassword
```

Adding a <Request> Element

Now that we've got an empty AACHRequest document, with the needed authentication elements, we're ready to start adding requests. Note that any number of requests can be included in a single AACHRequest document. The server will process all of the requests, in the order they appear in the document, before returning any results to the caller.

```
'
'Create a <Request> element and append it to the document element
'
dim aachrequest
set aachrequest = doc.documentElement.appendChild(doc.createElement("Request"))
aachrequest.setAttribute "ID",strID

'
'Create <RequestType> element and append it to the <Request> element
'
aachrequest.appendChild(doc.createElement("RequestType")).text = strType
```

Here we've created a <Request> element, set it's ID attribute to "0", and created the <RequestType> element as a child of the <Request> element. Note that doc is always used to create new elements regardless of where they're to be inserted into the tree. Also note that the newly inserted node is returned by appendChild, so we can directly access it's text property, to set the text content of the element.

Now, if the request we wanted to invoke required additional parameters, we'd need to add them to the request document at this time. Use a structure similar to what we just used to append the <RequestType> element.

Since we're invoking the ABASearch request, there is at least one more element we need to add, the ABA Number we are searching for:

```
aachrequest.appendChild(doc.createElement("ABANumber")).text = "322282603"
```

There are no further elements to add, so we're ready to invoke the request and see what we get back.

Invoking the Request

We'll use another object provided by MSXML to send the request to the server and get the response back: XMLHTTP. Once we have a response, we'll load it into another instance of XMLDOMDocument so that we can easily extract whatever elements we desire from the response.

Note: This example uses the XMLHTTP object. For production use in a server-based application, you should use the ServerXMLHTTP object, which exposes the same functions, but is designed and tuned for use in a server environment.

```
'
'Create an XMLHTTP instance
'
dim req
set req = CreateObject(strXMLHTTPProgID)

'
'Send the request & get the response
'
req.open "POST",strAACHRequesterUrl, false
req.send doc
```

The call to XMLHTTP.send sends the <AACHRequest> document to the server, using the HTTP POST method. The server's response is received and stored within the httpreq object until we're ready for it.

For the moment, we'll skip error checking, and jump straight into getting the response document.

```
'
'Load the response into an XMLDOMDocument and check for parse errors
'
if not objLastResponse.load(req.responseStream) then

        ' ... handle the error as appropriate

        exit sub
end if
```

That's it!

Extracting Results

Assuming no errors occurred the response object now contains the parsed response from the server. In order to examine the response, MSXML provides the powerful XPath query language. Using XPath it's easy to extract any particular node (or nodes) from the response document.

```
'
'Check for successful request invocation (this only tests that the entire
'request was received and processed - individual <Request> elements may have
'had errors)
'
dim wrapperStatus
wrapperStatus = GetResponseValue("/AACHRequest/ResponseSummary/Error")
if wrapperStatus = "True" then

        ' ... handle the error as appropriate

        exit sub
end if
```

If our request was processed successfully, the response document will contain one or more <Request> elements, corresponding to the <Request> elements which we included in the request document. Each of these elements serves as a container for all of the results produced by a single request invocation. Before we go looking for results, however, we should check the status of the actual request, to make sure that the request itself was processed successfully.

```
'
'Check for successful Request
'
dim statusCode
statusCode = session.GetResponseValue("/AACHRequest/Request/Status")
if statusCode <> "Success" then

        ' ... handle the error as appropriate

end if
```

Note that if we'd included more than a single <Request> element in the request document, we'd have to iterate through each of the <Request> elements in the response document, checking the <Status> of each one before extracting response data. Multiple nodes can be selected using the selectNodes method instead of selectSingleNode. As it happens, the request we invoked (ABASearch) returns multiple results, so we'll have to learn how to iterate through a set of nodes before getting much farther!

```
'
'Get the resultsItems & print them out
'
dim results
set results = session.GetResponseElementList("/AACHRequest/Request/Results/Result")
if results.length <> 0 then
        dim i
        for i = 0 to results.length-1
                stdout.WriteLine "Result " & i
                dim result
                set result = results.item(i)
                dim node
                set node = result.selectSingleNode("ABANumber")
                if not (node is Nothing) then
                  stdout.WriteLine "    ABA Number: " & node.text
```

```
                end if
                set node = result.selectSingleNode("Name")
                if not (node is Nothing) then
                  stdout.WriteLine "     Bank Name: " & node.text
                end if
                set node = result.selectSingleNode("Phone")
                if not (node is Nothing) then
                  stdout.WriteLine "  Phone Number: " + node.text
                end if          stdout.WriteLine ""
        next
else
        stdout.WriteLine "No accounts were found"
end if
```

The Results

This program was designed to be invoked using the console version of Windows Scripting Host, cscript.exe. Here's the results of a typical invocation of this program:

```
>cscript CallAACH.vbs username password
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

Result 0
    ABA Number: 322282603
     Bank Name: ARROWHEAD CREDIT UNION
  Phone Number:

Result 1
    ABA Number: 322282603
     Bank Name: ARROWHEAD CREDIT UNION
  Phone Number: 9098813355
```