Upload/Download Automation – Description and Example

Application Program Access

It is also possible to write a client application to access the upload/download interface instead of using a web browser. To do so, the application will need to issue a 'GET' or 'POST' HTTP request to the upload/download program at the following URI:

https://bidpost.nyiso.com/cgi-bin/main.exe

Use of the 'POST' request method is recommended to avoid the limitations on content length that occur with the 'GET' request. Use of uppercase in the request verb is also recommended.

As part of the HTTP request, the 'Content-length' header must be sent. The length value in the header should be set to the number of characters in the upload/download template header and body, as defined in Section 8.1.2. Although it is typical to send other HTTP request headers, such as 'Content-type', they will be ignored by the upload/download program and may be omitted.

The response from the upload/download program may be parsed as a stream or stored for later processing. The format of the response follows the conventions defined in Section 8.1.2. Depending on your application program, you may need to parse the HTTP response headers as well.

Since access to the upload/download program requires authentication via the use of digital certificates, an application program may need to handle the exchange of web server and client certificates that occurs as part of the handshake before an SSL connection is established. Use of a particular certificate store or client certificate may also require special processing.

For a Windows client, a possible implementation of a client application can be done using Microsoft Windows HTTP Services (WinHTTP). An example below is provided in Visual Basic Script, using the WinHTTPRequest COM object and has been tested on Windows 2000:

```
Option Explicit

Dim oFSO, oWinHttpRequest, fStream
Dim sURI, sUDTemplate, sUDResponse, sHttpResponse
Dim nContentLength

' Instantiate a File System Object for storage of the response
Set oFSO = CreateObject("Scripting.FileSystemObject")

' Instantiate a WinHttpRequest object to handle the request/response
Set oWinHttpRequest = CreateObject("WinHttp.WinHttpRequest.5")

' Define the URI of the upload/download program
sURI = "https://bidpost.nyiso.com/cgi-bin/main.exe"

' Define the upload/download template header and body (you will need to
use an actual userid/password)
```

```
sUDTemplate =
"USERID=***&PASSWORD=***&QUERY_TYPE=GEN_SCH&MARKET_TYPE=DAM&DATE=06/01/200
3&GENERATOR=23571"

' Determine the length of the request
nContentLength = Len(sUDTemplate)

' Open the connection
oWinHttpRequest.Open "POST", sURI, False

' Set the HTTP request headers (only content-length is required)
oWinHttpRequest.SetRequestHeader "Content-type", "multipart/form-data"
oWinHttpRequest.SetRequestHeader "Content-length", nContentLength

' You would need to select a client certificate using the
SetClientCertificate method if you don't use the default

' Send the request (this also handles the client-server certificate
exchange)
oWinHttpRequest.Send sUDTemplate

' Upload/download program processes the request and sends a response

' Retrieve the upload/download response
sUDResponse = oWinHttpRequest.ResponseText

' Retrieve the HTTP reponse if you're interested
sHttpResponse = oWinHttpRequest.GetAllResponseHeaders

' Write out the HTTP and u/d responses
Set fStream = oFSO.CreateTextFile("C:\Temp\response.txt")
fStream.WriteLine(sHttpResponse)
fStream.WriteLine(sUDResponse)
fStream.Close
```

For more information on the HTTP 1.1 request/response protocol, please refer to RFC 2616. For more information on WinHTTP or Visual Basic Script, please refer to the Microsoft MSDN site.