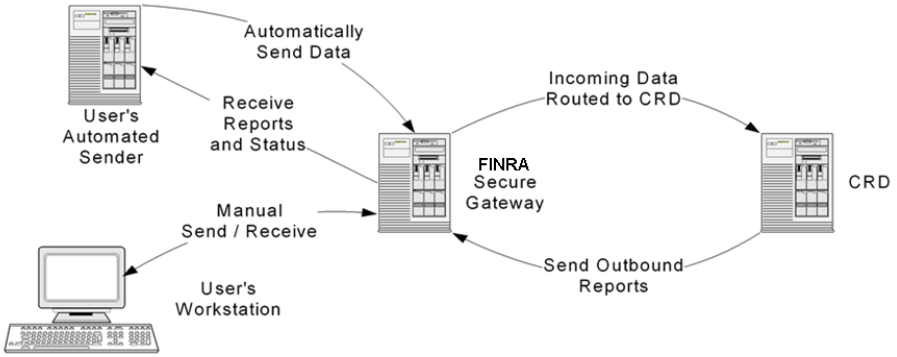
Web EFT ™
User's Guide

Release 2018.12

Table of Contents

Introduction

Web EFT (Electronic File Transfer) allows subscribing firms to interface with Web CRD® and IARD™ in an automated manner (Web EFT also offers a manual interface). Web EFT offers firms the ability to upload batches of form filings to Web CRD and IARD and provides its subscribers with reports containing the results of batch form file processing, as well as downloads of registration and accounting activity.



The Web EFT System currently accepts the following form filing types:

U4 Initial	U5 Partial
U4 Relicense All	U5 Full
U4 Relicense CRD	U5 Amendment
U4 Relicense IA	Non Registered Fingerprint Initial
U4 Dual	Non Registered Fingerprint Amendment
U4 Amendment	Non Registered Fingerprint Termination
U4 Page 2 Initial for Schedule A/B	BR Initial
U4 Page 2 Amendment for Schedule A/B	BR Amendment
	BR Closing/Withdrawal

NOTE: Web EFT does not currently support programmatic uploads of Form BD/BDW, Form ADV/ADV-W, Form U6, or Form U4 Concurrence filings. These must be filed online through Web CRD/IARD.

Web EFT also provides the following CRD/IARD downloads:

Downloads	**Frequency**
Branch Information Report	daily
Individual Information Report	weekly
Individual Information Report Delta	daily
Post Accounting Report	daily
Post Appointments Report	daily
Post Approvals Report	daily
Post Branch Deficiencies Report	daily
Post Branch Individual Report	daily
Post Branch Status Report	daily
Post Deficiencies Report	daily
Post Enrollments Report	daily
Post Exams Report	daily
Post Fingerprint Report	daily
Post Pending Report	daily
Post Terminations Report	daily

Web EFT provides two interfaces for uploading batch form filings and downloading reports:

(1) A web interface (see Section 1)
(2) A direct machine-to-machine interface (see Section 2)

Details on using these methods are described in the following sections:

1 Web Interface

The web interface is provided to support access to Web EFT through a standard HTTPS browser interface. The FINRA Secure Gateway (FSG) site uses a secure web server that supports Internet-standard Secure Sockets Layer (SSL) to protect all data sent to and from the server. Using a standard Web browser, users can securely access the site to upload batch form files, download reports and check the status of batch form files that were previously submitted.

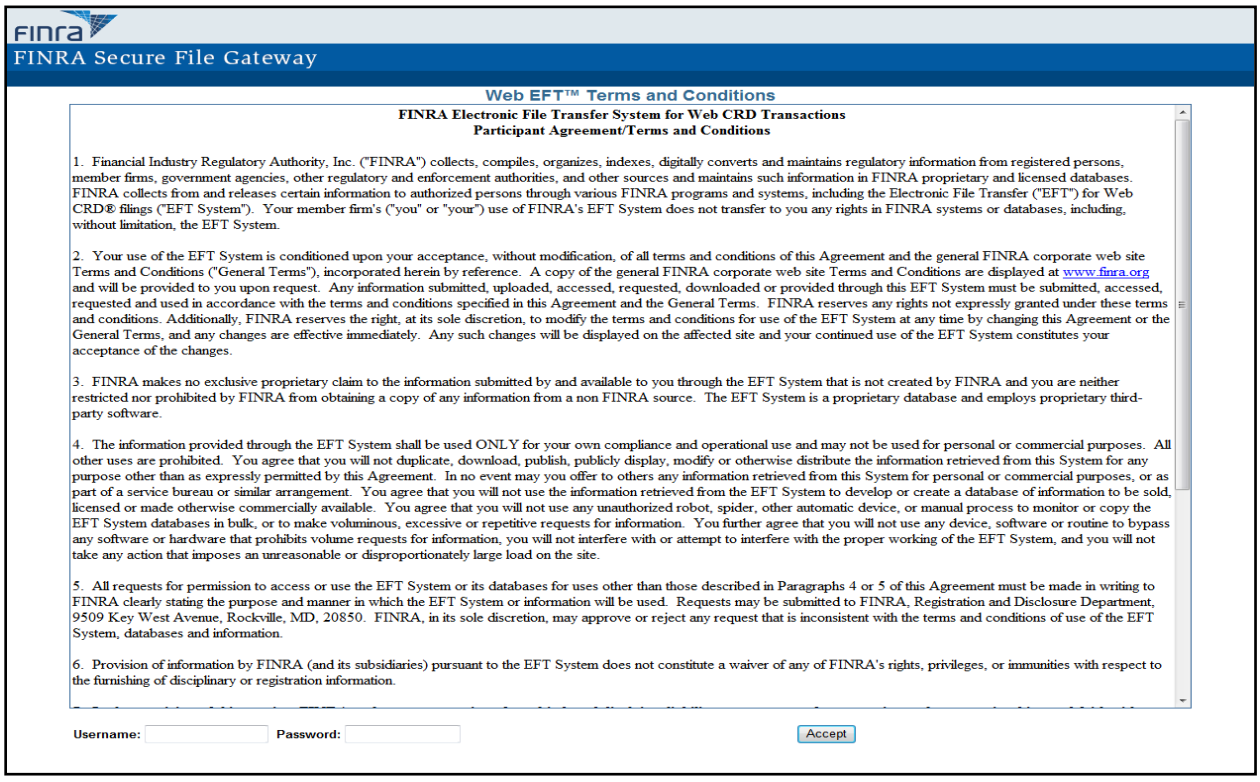
1.1 Logging In

Users must access the web interface through a login screen, which displays below. FINRA provides a username and password to each subscribing firm. The user must enter the firm's username and password and accept the Web EFT Terms and Conditions.

When your Web EFT account is initially established, you will be issued a "one-time-only" password. On your initial login, you will be forced to change the issued password to one of your choosing. The password you choose must meet the following criteria:

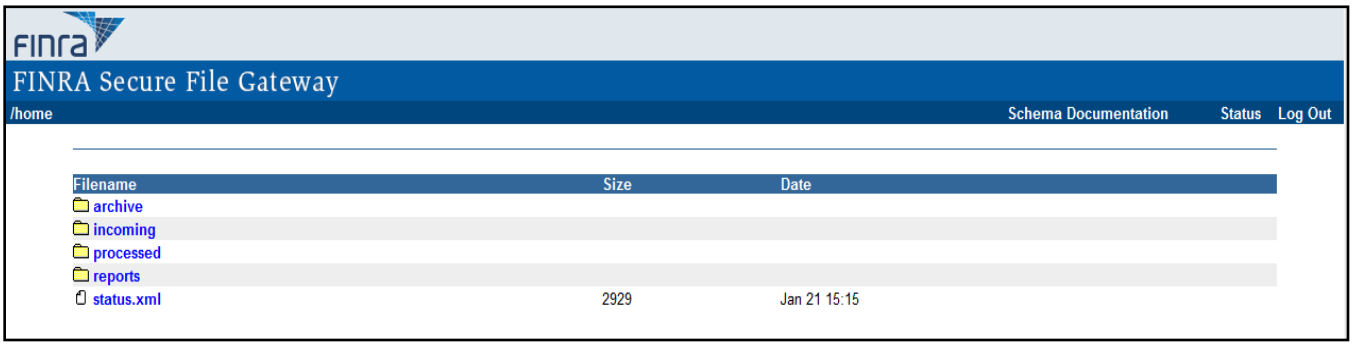
- Password must be between 8 and 10 characters.
- Password must contain at least 1 alpha character.
- Password must contain at least 1 numeric character.
- Password must contain at least 1 "special" (non-alpha, non-numeric) character. "#" or "$", for example.

Web EFT passwords do not automatically expire, however we recommend that firms change their password every 90 days (similar to the Web CRD password expiration timeframe). To open a case to change your password, please contact the FINRA Gateway Call Center at (301) 869-6699. Your old password will be set to expired and you will be directed to choose a new password on your next login.



1.2 Home Page

After a successful login, the system will display the home directory of the firm user. This screen is used to access the functions available to the firm.



There are four sub-directories in the firm's home directory:

- **"archive"** – Contains a copy of all Web EFT files submitted and reports generated for each firm in the last 31 days (see Section 1.5).

- **"incoming"** – All batch form files sent to FSG must be uploaded to the "incoming" directory (see Section 1.6). If the user does not subscribe to the Web EFT batch form filing upload service, the incoming directory will not appear.

- **"processed"** – Stores all batch form filing results files corresponding to batch form filing submissions. One batch results file is created for each batch form file submission (see Section 1.7).

- **"reports"** – All daily and weekly Web CRD reports are posted to the reports directory (see Section 1.8). If the user does not subscribe to the Web EFT download service, the reports directory will be empty.

Available links:

- **"/home"** – Indicates the user's present directory. When the user navigates to one of the subdirectories, this display will include the name of the subdirectory (for example, if a user selects the reports directory, this link will change to **"/home/reports"**).

  To access a subdirectory, select the name of the subdirectory. To move from the subdirectory back to the home directory, either select the "/home" link in the upper left-hand side of the tool bar, or use your browser's **"Back"** button.

- **"Schema Documentation"** – Links to the current form filing and reports schema files (.xsd) located on FINRA's Web site (www.finra.org/webeft/schema) (see Section 1.9).

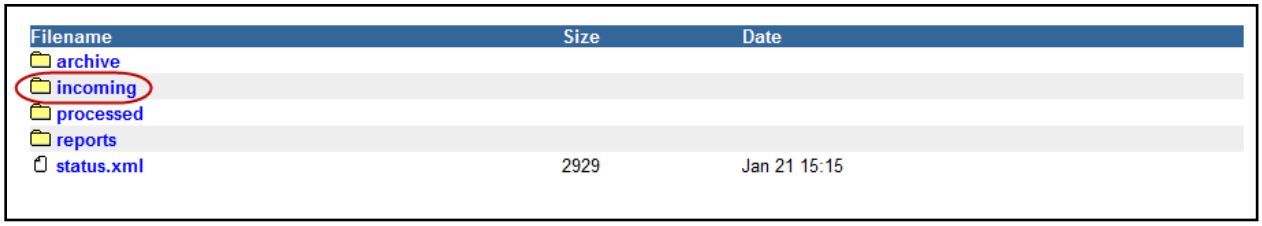
- **"Status**" and **"status.xml"** – Links to the status file which lists the processing status of all batch form files sent to Web EFT in the past 31 days (see Section 1.10).

- **"Log Out"** – Logs the user out of Web EFT (see Section 1.11).

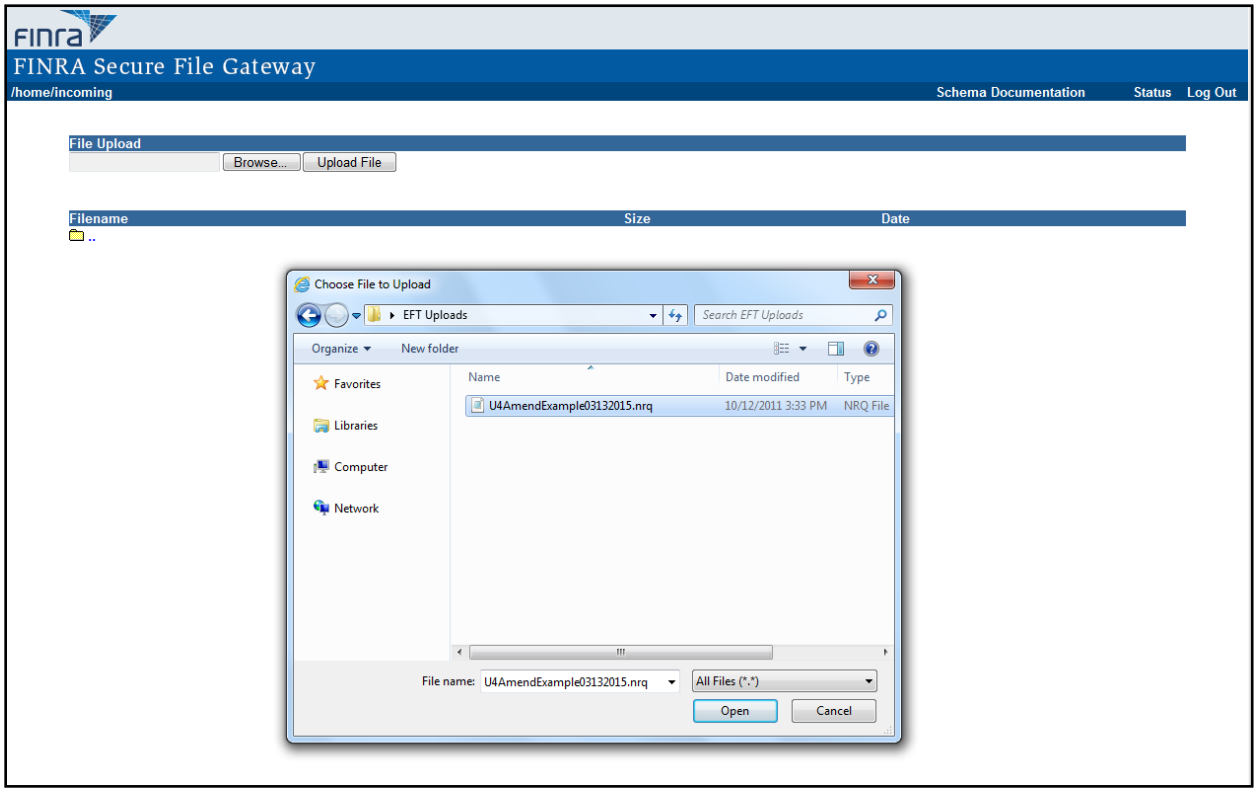
1.3 Submitting (Uploading) Batch Form Files to Web CRD

Users can upload batch form files to FSG (which submits the forms within the file to Web CRD) using the "File Upload" function at the top of the directory page.

To upload a file, select the "incoming" directory.



On the "File Upload" form, select the "Browse" button. This opens a standard directory browser on your local computer.
Select the file you wish to upload.
Select the "Open" button in the file window.
Select the "Upload File" button to send your file to Web EFT. This submits your form filings to Web CRD.



File Submission (Upload) Rules:

- Any files uploaded to Web EFT must be loaded to the **"incoming"** directory.
- All uploaded batch form files must be named with an '.nrq' (lower case) extension to be processed by Web EFT. (If a file does not have an .nrq extension, it will not be processed.)
- All uploaded batch form files must not contain spaces in the filename. (If a filename contains spaces, the file will not be processed.)
- Each uploaded batch form file must be uniquely named within the last 31 days. An attempt by a firm to upload a file with a duplicate name will result in an error message.
- All uploaded batch form files must be in XML format and must comply with either, the CRD U4, U5, NRF or BR Form Filing Schema.

- In order to process successfully, any form within the file must also comply with Web CRD's business rules and completeness checks. [**Tip:** If a form fails a Web CRD completeness check, the unprocessed form may also be located and corrected "online" in the firm's "Pending Filings" queues in Web CRD.]
- Web EFT's XML parser does not recognize the standard "<!-- -->" XML comment syntax. If a comment is included in the file, Web EFT will attempt to process the commented text.

When Web EFT and Web CRD have completed processing a batch form file, a batch results file with the same name as the uploaded file and the extension '.nrs' will be placed in the processed directory. Also, the EFT primary technology contact listed on your Web EFT UAAF will receive an e-mail indicating that Web EFT has finished processing the file.

For example:

1) A user (or automated script) uploads a file named *U4addressupdate.nrq* into the user's **incoming** directory.
2) Web EFT sends the form data within the file to Web CRD for form processing.
3) Web EFT also sends an e-mail to the subscriber's primary technology contact, indicating that the file was received.
4) When the file is completely processed, Web EFT moves *U4addressupdate.nrq* to the firm's **archive** directory and places a results file named *U4addressupdate.nrs* in the user's **processed** directory.
5) Web EFT also sends an e-mail to the subscriber's primary technology contact, indicating that the file has completed processing.
6) A copy of *U4addressupdate.nrs* is also placed in the firm's **archive** directory.

NOTE: After a successful file upload, primary technology contact listed on your Web EFT User Account Acknowledgement Form (UAAF) will receive an e-mail indicating that Web EFT has received the file. Should the primary technology contact chooses not to receive any processing e-mails, two steps must be taken:

(1) Contact the FINRA Gateway Call Center at (301) 869-6699 or email and request that e-mail address be excluded from receiving any processing e-mails.
(2) When submitting a batch form file, omit the Email attribute in the NSGBatch element (please refer to one of the form filing schema for the Email attribute's precise location).

To update your EFT contacts, please use the UAAF form found at: www.finra.org/webeft > Forms for Existing Web EFT Firms.

1.4 Retrieving Batch Form Filing Results Files and Reports from Web EFT

All Web EFT submitted batch form files, batch results files, and scheduled reports in any of the firm's directories are produced in XML format and can be viewed in a browser or downloaded to the user's computer. To view a file, simply **"single-click"** on the name of the file. To download a file, right-click on the name and select **"Save Target As"** from the menu (Firefox users would select **"Save Link As"** in the menu). **Do not "double-click"** on the hyperlink. Double-clicking may cause the file to be become corrupt.

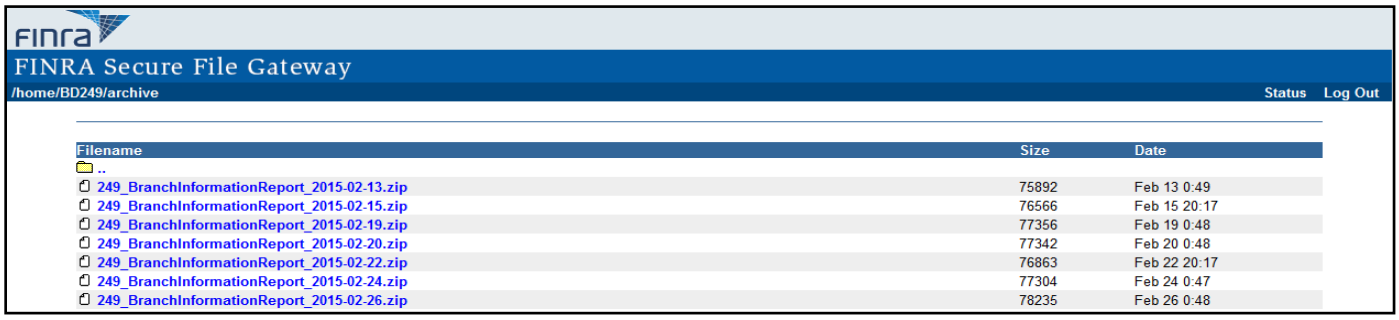
Any files viewed or downloaded in the firm's **"processed"** and **"reports"** directories are removed immediately after they are viewed or downloaded. Copies of all files are maintained in the "archive" directory for 31 days.

When a user selects the **"processed"** or **"reports"** directory, the user knows that any file in the processed or reports directory is new and has not yet been retrieved. If a subsequent retrieval of a file is required, the file will still exist in the archive directory for 31 days.

For example:

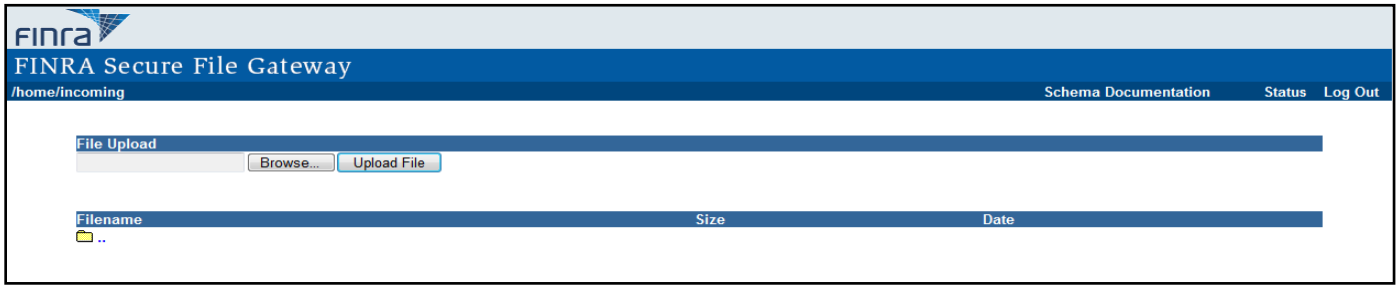
1) Web EFT places a daily report *BDNumber_PostApprovalsReport_2017-05-01.app* in the firm's **reports** directory.
2) A copy of the daily report *BDNumber_PostApprovalsReport_2017-05-01.app* is also placed in the firm's **archive** directory.
3) A user (or automated script) retrieves the daily report *BDNumber_PostApprovalsReport_2017-05-01.app* from the firm's **reports** directory (either manually by opening the file in a browser or by right-clicking and saving the file locally, or automatically by performing a **"get"**).
4) Daily report *BDNumber_PostApprovalsReport_2017-05-01.app* is deleted from the firm's **reports** directory.
5) The copy of daily report *BDNumber_PostApprovalsReport_2017-05-01.app* will remain in the firm's **archive** directory for 31 days.

1.5 The archive folder



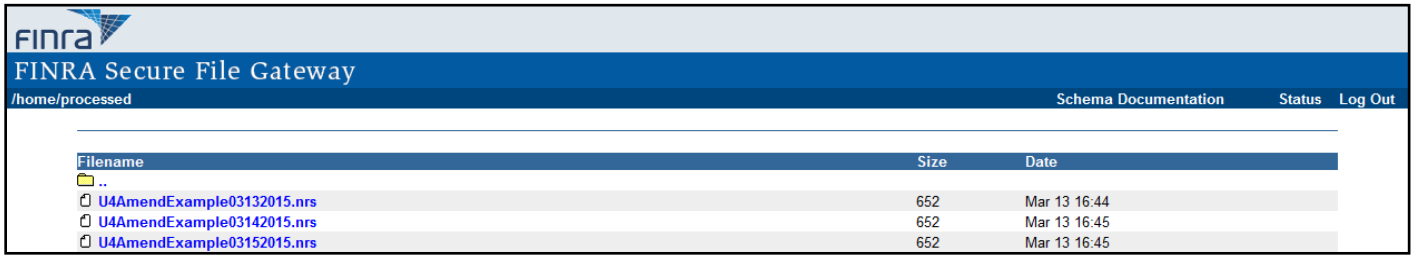
The **"archive"** directory retains a copy of all daily and weekly reports that were delivered to the firm within the previous 31 days. Also retained are batch form files submitted to Web EFT as well as copies of all batch form results files. The files are retained in the archive directory for 31 days. After 31 days, all files and reports are deleted automatically. The Full Individual Information Report is deleted every Sunday due to size.

1.6 The incoming folder



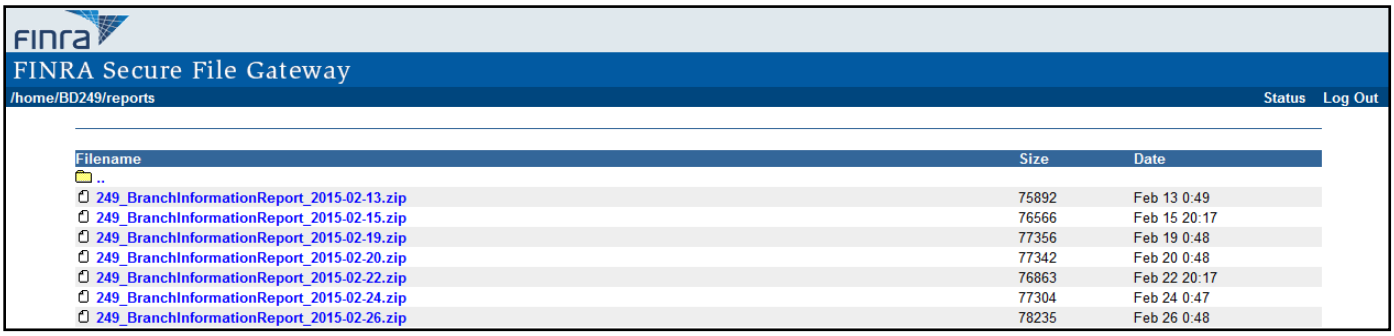
All batch form files sent to Web EFT must be uploaded to the **"incoming"** directory. The files are held in the incoming directory until the file has been completely processed. Once the file is completely processed, the uploaded file is automatically moved to the archive directory and no longer appears in the incoming directory.

1.7 The processed folder



The **"processed"** directory stores all batch form filing result files corresponding to batch form filing submissions. One batch form result file is created for each batch form filing file submission. Each batch form result file contains an entry for each form submitted, indicating whether the form submission was successful and, if unsuccessful, the reason for failure.

1.8 The reports folder



The daily and weekly CRD Reports are placed in the **"reports"** directory. These reports are produced in XML format and can be viewed in a browser or downloaded to the user's computer. Reports are removed from this directory once they are accessed.

NOTE: The Individual Information Report and the Individual Information Report Delta are produced as a zipped XML document, due to the potentially large size of the report. It can be unzipped with any standard **"unzip"** utility (for example, WinZip® or pkunzip®). If you have any questions regarding unzipping files, please contact your Technology Department.

1.9 Schema Files

There are five form filing schema files (.xsd): U4, U5, DRP, NRF, and BR. These form filing schema files may be downloaded to a user's computer and loaded locally into a user's XML editing software. This permits the user to validate files against the CRD Form Filing schema immediately, without the need to repeatedly upload batch form filings and wait for the results. Web EFT requires all data to be in XML format. Any form filings submitted via Web EFT must also comply with Web CRD's completeness checks and business rules. Report schemas provide the basic data elements to help users build their tables. These schema files are located on FINRA's Web site (www.finra.org/webeft/schema) and are updated periodically to reflect any changes to the CRD Form Filing and Web EFT Reports schemas.

1.10 View Batch Status

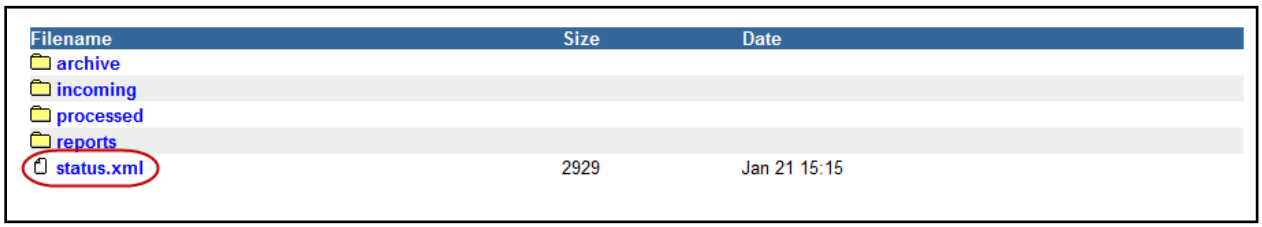
Users may access the Web EFT status page in two ways. Users may select on the **"Status"** link that appears in the upper right corner of each page. When selected, the link displays a list of the processing status of all batch form files sent to Web EFT in the past 31 days.

There are four possible statuses for any submitted file:

- **Submitted** – The file has been submitted to Web EFT and is queued for CRD processing.
- **Processing** – CRD has received the file and begun processing it.
- **Completed** – CRD has finished processing the file. There were no errors on any forms within the file.
- **Completed with Errors** – CRD has finished processing the file. At least one of the forms within the file was not successfully processed.

In addition, selecting the **"status.xml"** link in the home directory will also provide the status file.

Sample status.xml file:



Filename	Size	Date
archive		
incoming		
processed		
reports		
status.xml	2929	Jan 21 15:15

```
<?xml version="1.0" encoding="utf-8" ?>
<?xml:stylesheet type="text/xsl" ?>
- <status lastUpdate="2008-05-19T10:00:32" xmlns="http://nsg.nasd.com/2002/03/01/NSGStatus.xsd">
    <file UploadTime="2008-05-19T10:00:21" Status="Completed with Errors" StatusTime="2008-05-19T10:00:32" FileSize="4265"
      FileName="archive/U4_Initial_5.nrq" UserBatchRef="BatchReg'Test" TrackingNumber="BDFIRM.1212121212.00001"
      StatusReport="archive/U4_Initial_5.nrs" NumberRequests="1" NumberCompleted="1" NumberErrors="1" />
    <file UploadTime="2008-05-19T09:57:34" Status="Completed with Errors" StatusTime="2008-05-19T09:57:43" FileSize="4262"
      FileName="archive/U4_Initial_4.nrq" UserBatchRef="BatchReg'Test" TrackingNumber="BDFIRM.1212121212.00002"
      StatusReport="archive/U4_Initial_4.nrs" NumberRequests="1" NumberCompleted="1" NumberErrors="1" />
    <file UploadTime="2008-05-19T09:54:53" Status="Completed with Errors" StatusTime="2008-05-19T09:54:58" FileSize="4264"
      FileName="archive/U4_Initial_3.nrq" UserBatchRef="BatchReg'Test" TrackingNumber="BDFIRM.1212121212.00003"
      StatusReport="archive/U4_Initial_3.nrs" NumberRequests="1" NumberCompleted="1" NumberErrors="1" />
    <file UploadTime="2008-05-19T09:53:59" Status="Completed with Errors" StatusTime="2008-05-19T09:54:00" FileSize="4264"
      FileName="archive/U4_Initial_2.nrq" UserBatchRef="BatchReg'Test" TrackingNumber="BDFIRM.1212121212.00004"
      StatusReport="archive/U4_Initial_2.nrs" NumberRequests="1" NumberCompleted="1" NumberErrors="1" />
    <file UploadTime="2008-05-19T09:50:24" Status="Completed with Errors" StatusTime="2008-05-19T09:50:28" FileSize="4254"
      FileName="archive/U4_Initial_1.nrq" UserBatchRef="BatchReg'Test" TrackingNumber="BDFIRM.1212121212.00005"
      StatusReport="archive/U4_Initial_1.nrs" NumberRequests="1" NumberCompleted="1" NumberErrors="1" />
    <file UploadTime="2008-05-19T08:42:37" Status="Completed with Errors" StatusTime="2008-05-19T08:42:44" FileSize="1595"
      FileName="archive/NRF-Inital-122908_1.nrq" UserBatchRef="BatchRegTest" TrackingNumber="BDFIRM.1212121212.00006"
      StatusReport="archive/NRF-Inital-122908_1.nrs" NumberRequests="1" NumberCompleted="1" NumberErrors="1" />
    <file UploadTime="2008-05-19T07:24:56" Status="Completed with Errors" StatusTime="2008-05-19T07:25:04" FileSize="1598"
      FileName="archive/NRF-Inital-122908.nrq" UserBatchRef="BatchRegTest" TrackingNumber="BDFIRM.1212121212.00007"
      StatusReport="archive/NRF-Inital-122908.nrs" NumberRequests="1" NumberCompleted="1" NumberErrors="1" />
  </status>
```

1.11 Logout

Selecting the Logout link will log the user out of the Web EFT system. The user will need to login again to access the system.

1.12 Hours of Operation

Web EFT Availability (File Upload/Download)

Web EFT is generally available to upload and download files 24 hours a day, except for occasional maintenance periods. Please see the Web EFT System Availability & Upcoming Releases page for more information.

Please note that uploaded files are processed during Web CRD's hours of operations.

Web CRD Availability (File Processing)

Web CRD is typically available:

- Weekdays from 5 a.m. to 11 p.m., Eastern Time (ET)
- Saturday from 8 a.m. to 6 p.m., ET
- Sunday from 10 a.m. to 6 p.m., ET

Scheduled outages are reported on the Web CRD Availability Schedule page.

Batch form files that are uploaded outside of Web CRD's production hours will remain in the firm's "incoming" directory until Web CRD is available again to complete file processing. Similarly, if a batch does not finish processing before Web CRD shuts down, the system will suspend processing until Web CRD is back online.

Test Environment Availability

The Web EFT and Web CRD Test Environments are available Monday – Friday from 9:00 a.m. to 8:00 p.m., ET (except for occasional outages due to software deployments and other internal testing activities).

2 Machine-to-Machine Interface

Since all communication between the user's computer and the Web EFT server is achieved through a standard HTTPS transmission, any tools that support the HTTPS protocol for sending and receiving files can be used to communicate with Web EFT. (Due to the nature of custom scripting and third-party tools and the multiple ways of implementing a machine-to-machine interface, FINRA cannot support or trouble-shoot automated user interfaces.)

Machine-to-machine interaction still requires the user's machine to provide a valid user ID and password. Because all Web EFT accounts are initially issued a one-time-only password, there must be an initial login through the Web interface to set the permanent password (see Section 1.1). Upon completion of each session, please logout to clear all connections.

2.1 Script Samples

Sample Java and Perl scripts are provided (see Appendix A) to assist you with identifying and systematically retrieving files from the FSG Web site.

2.2 Certificate Errors

Should you receive a certificate error, go to the Entrust website for support (https://www.entrust.com/get-support/ssl-certificate-support/). When a script includes a request to list the contents of a directory and html content is returned instead of a text directory listing, try setting the USER-AGENT to "SecureTransport" (case sensitive).

3 Pre-Production Testing

Prior to being given access to the Web EFT *production environment*, firms are required to successfully use the Web EFT *test environment*. New subscribers will be provided with a login to the test environment. Please note that there are separate required test items for Upload access and Download access.

In addition to the Web EFT test environment username and password, if deemed necessary, each firm may request a username and password to the Web CRD test environment that ultimately processes Web EFT test files. Firms can use this Web CRD test environment logon to verify that their Web EFT test batch form filings have been processed correctly on test Web CRD, and to sample the report data posted by Web EFT.

If your firm intends to use Web EFT for batch form filing, your testing steps should include:

- Logging onto the Web EFT test environment.
- Building an XML batch form file that complies with the CRD schema and business rules.
- Uploading a batch file successfully.
- Retrieving a results file successfully.
- Verifying that your forms appear in the Web CRD test environment Form Filing History (Optional).

If your firm intends to use Web EFT for reports, your testing steps should include:

- Logging onto the Web EFT test environment.
- Retrieving a report successfully.
- Verifying that data within your report compares accurately with data in the Web CRD test environment (Optional).

Passed: Yes or No	Testing Check List for Upload access (Batch Form Filing) to Web EFT Production:
	1. User logs onto Web EFT test environment and successfully changes the initial password.
	2. User successfully uploads a batch form file containing at least **one** of the specific Web CRD Form Filing types below, and views the .nrq file in the firm's directories. Available filing types: a. U4 Initial U4 b. U4 Relicense c. U4 Dual U4 d. U4 Amendment of Page 1 info e. U4 Amendment of Page 2 info f. U4 Page 2 Amendment for Schedule A/B g. U5 Partial U5 h. U5 Full i. U5 Amendment j. NRF (Non-Registered Fingerprint) k. BR Initial l. BR Amendment m. BR Closure
	3. User retrieves an .nrs results file from the firm's directories. The uploaded batch form file's *CRDFormResults* indicates "success".
	4. User has determined readiness to move to Web EFT Production, and contacts the FINRA Gateway Call Center at 301-869-6699 or email.
Passed: Yes or No	**Testing Check List for Download access (Reports) to Web EFT Production**
	1. User logs onto Web EFT test environment and successfully changes the initial password.
	2. User has determined readiness to move to Web EFT Production, and contacts the FINRA Gateway Call Center at 301-869-6699.

4 Support

Users are encouraged to review the Web EFT Frequently Asked Questions for answers to common questions. For other questions or to report issues, please contact the FINRA Gateway Call Center at (301) 869-6699 or email.

5 Appendix A: Script Samples

Please note that the following Download and Upload scripts in JAVA, Perl, VBS were developed for an internal proof-of-concept test and did work at that time, running on internal FINRA machines within the FINRA network. These scripts are being provided to Web EFT users for example purposes only and FINRA does not guarantee that these scripts will work on any given platform or machine. As such, FINRA does not provide any support-related services in connection to these scripts.

5.1 Java basic download sample:

```
/*********************************************************************
* Copyright 2008, FINRA
*
* File BasicDownloadExample.java
*
* Purpose:
*     Example program on how to interact with WebEFT to
*     view files in a directory or to download files.
*
* No warranties are expressed or implied.
* This example is UNSUPPORTED and is only intended as an example
* on how to interact with WebEFT.
*
* This requires knowledge of Java and the jar libraries below, in
* particular  HttpClient. Jar libraries needed:
* commons-httpclient-3.1-rc1.jar, commons-codec-1.3.jar,
* commons-logging-1.1.jar.
*
* See www.apache.org or http://hc.apache.org/httpclient-3.x for more information
* about the libraries.
*
* Compile by giving classpath to jar files:
*     javac.exe -cp .\commons-httpclient-3.1-rc1.jar;.\commons-codec-1.3.jar;.\commons-
logging-1.1.jar .\BasicDownloadExample.java
*     Will create BasicDownloadExample.class in current directory
*   or
*   Make sure to have classpath set to run the BasicDownloadExample.class
*   Windows:
*   set CLASSPATH=.\commons-httpclient-3.1-rc1.jar;.\commons-codec-1.3.jar;.\commons-
logging-1.1.jar;%CLASSPATH%
*   UNIX:
*   CLASSPATH=<your_path_to>/commons-httpclient-3.1-rc1.jar:<your_path_to>/commons-codec-
1.3.jar:<your_path_to>/commons-logging-1.1.jar:$CLASSPATH; export CLASSPATH
*
* Usage:
*     java BasicDownloadExample [url] [BD#] [Password]
* - If the [url] is a directory, this will get the listing of reports
*   in the path.
*
* java BasicDownloadExample [url] [BD#] [Password] [local_file_name]
* - If the [url] is a filename, the filename will be downloaded as the
*   [local_file_name]
*
* [url] for production is fsg.finra.org or some path using that.
*     fsg.finra.org
*     fsg.finra.org/reports
*     fsg.finra.org/reports/filename.zip
*
*
* Examples:
* java BasicDownloadExample https://fsg.finra.org BD12345 secret_password
*  - Get the listing of files in the main directory
```

```
*  - The downloaded directory listings [and files] will be save to C:\temp\downloadfile-
YYYYMMDD-HHMMSS.log directory. Check that C:\temp is available.
*
* java BasicDownloadExample https://fsg.finra.org/reports/example.zip BD12345
secret_password file.zip
*  - Download the example.zip file from the reports directory and save it as file.zip
locally.

*/
import java.io.*;
import java.util.*;
import java.net.*;
import java.lang.*;
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.UsernamePasswordCredentials;
import org.apache.commons.httpclient.auth.AuthScope;
import org.apache.commons.httpclient.methods.GetMethod;
import org.apache.commons.httpclient.HttpStatus;
import org.apache.commons.httpclient.methods.PostMethod;
import org.apache.commons.httpclient.methods.MultipartPostMethod;

public class BasicDownloadExample {

    /**
     * Constructor for BasicAuthenticatonExample.
     */
    public BasicDownloadExample() {
        super();
    }

    public static void main(String[] args) throws Exception {

          // Check for Arguments - Less than 3 arguments?
      if ( args.length < 3 ) {
            System.out.println( "Usage: java BasicDownloadExample URL username password"
);
            System.out.println( "Eg:    java BasicDownloadExample
https://fsgtest6.finra.org/ BD# BD's-password");
            System.out.println( "Eg:    java BasicDownloadExample
https://fsgtest6.finra.org/archive BD# BD's-password");
            System.out.println( "Eg:    java BasicDownloadExample
https://fsgtest6.finra.org/archive/filename BD# BD's-password");
            System.out.println( "Eg:    java BasicDownloadExample
https://fsgtest6.finra.org/processed BD# BD's-password");
            System.out.println( "Eg:    java BasicDownloadExample
https://fsgtest6.finra.org/reports BD# BD's-password");
            System.out.println( "Eg:    java BasicDownloadExample
https://fsgtest6.finra.org/reports/filename.zip BD# BD's-password filename.zip");
            System.exit( 0 );
      }

        URL aURL = new URL(args[0]);
      String username = args[1];
      String password = args[2];
      String fileName;
      // If user specified 4 argument for given filename
      if (args.length == 4) {
            fileName = args[3];
      } else
            fileName = "";
      String file;
```

```
      int STInterface = 1; // To turn of STInterface, set to 0 which will be slower
because of html tags being downloaded also.


        System.out.println("protocol = " + aURL.getProtocol());
      System.out.println("authority = " + aURL.getAuthority());
        System.out.println("host = " + aURL.getHost());
        System.out.println("path = " + aURL.getPath());
        System.out.println("filename = " + aURL.getFile());

      // Get basename of the URL
      String bn = aURL.getFile(); int index = bn.lastIndexOf("/"); bn =
bn.substring(index + 1);
      System.out.println("basename = " + bn);

      // Convert URL to String
      String url = (String)aURL.toString();

      HttpClient client = new HttpClient();

        // pass our credentials to HttpClient, they will only be used for
        // authenticating to servers with realm "FileDriveWWW" on the host
        // "aURL.getHost()", to authenticate against
        // an arbitrary realm or host change the appropriate argument to null.
        client.getState().setCredentials(
            new AuthScope(aURL.getHost(), 443, "FileDriveWWW"),
            new UsernamePasswordCredentials(username, password)
        );

        // create a GET method that reads a file over HTTPS, we're assuming
        // that this file requires basic authentication using the realm above.
         GetMethod get = new GetMethod(url);

        try {
            // execute the GET
            int status = client.executeMethod( get );

            // print the status and response
//            System.out.println(status + "\n" + get.getStatusCode() + "\n"+ url);

          // Get the name of file in URL or set the name for the directory listing of URL

          if (fileName.length() != 0) {
            file = fileName; // Set to fileName which user specified
          } else {

          // Get daytime stamp to tag to file for directory listing to save to disk
          Calendar cal = Calendar.getInstance(TimeZone.getDefault());
          String DATE_FORMAT = "yyyyMMdd-HHmmss";
          java.text.SimpleDateFormat sdf = new java.text.SimpleDateFormat(DATE_FORMAT);

          // Set filename for saving the directory listing of root directory of URL to
disk
          if (bn == "" || bn.length() == 0) {
            file = "c:\\temp\\downloadfile-" +sdf.format(cal.getTime())+".log";
          } else { // Set filename for saving the directory listing of subdirectories of
URL
               if (getExtension(bn) == "") {
                  file =  "c:\\temp\\" + bn + sdf.format(cal.getTime())+ ".log";
               } else { // Set filename for saving the file specified in URL to disk
                    file = "c:\\temp\\" + bn;
               }
```

```
          }
          }
          // Initialize file to be save to disk
          OutputStream out = new FileOutputStream(file);

          if (get.getStatusCode() == HttpStatus.SC_OK) { // if StatusCode == 200
            // Specify the USER-AGENT to make the download more efficient
            if (STInterface == 1) {
                  get.setRequestHeader("USER-AGENT", "SecureTransport");
            }

            // Getting the URL
            client.executeMethod( get );

            // Saving the URL response to Stream Object
                  InputStream in = get.getResponseBodyAsStream();

            byte[] b = new byte[1024];
            int len;
            while ((len = in.read(b)) != -1) {
            //write bytes to file on disk
                  out.write(b, 0, len);
            }

            System.out.println(url + " downloaded to " + file);
            in.close();
            } else { // if StatusCode is not 200
                  System.out.println("Return Code: " + get.getStatusLine().toString());
            }


        } finally {
            // release any connection resources used by the method
          GetMethod logout = new
GetMethod(aURL.getProtocol()+"://"+aURL.getHost()+"/?logout");
          client.executeMethod( logout );
            get.releaseConnection();
        }
    }

    // Utility to determine if the basename of the URL is a directory or a filename with
extension
    public static String getExtension(final String filename) {
        String suffix = "";
        String shortFilename = filename;

        int lastDirSeparator = filename.lastIndexOf(File.separatorChar);
        if(lastDirSeparator > 0){
            shortFilename = filename.substring(lastDirSeparator + 1);
        }

        int index = shortFilename.lastIndexOf('.');

        if (index > 0 && index < shortFilename.length() - 1) {
            suffix = shortFilename.substring(index + 1);
        }

        return suffix;
    }
}
```

5.2 Java upload sample:

```
import java.net.*;
import java.io.File;
import java.io.FileInputStream;
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.UsernamePasswordCredentials;
import org.apache.commons.httpclient.auth.AuthScope;
import org.apache.commons.httpclient.methods.GetMethod;
import org.apache.commons.httpclient.HttpStatus;
import org.apache.commons.httpclient.methods.InputStreamRequestEntity;
import org.apache.commons.httpclient.methods.PostMethod;
import org.apache.commons.httpclient.methods.MultipartPostMethod;
import org.apache.commons.httpclient.HttpState;

/**
 * A simple example that uses HttpClient to perform a GET using Basic
 * Authentication. 4 parameters needed: URL username password nrq-file.
 *
 * You need to have JSSE on your classpath for JDK prior to 1.4
 * Disclaimer: This is an example of java script that was developed for an internal
proof-of-concept test.
 * The script did work at that time, running on internal FINRA machines within the FINRA
network.
 * We cannot guarantee that the script as written will work on any given platform or
machine.
 *
 */
public class BasicAuthenticationUploadExample {

    /**
     * Constructor for BasicAuthenticatonExample.
     */
    public BasicAuthenticationUploadExample() {
        super();
    }

    public static void main(String[] args) throws Exception {

      if ( args.length < 4 ) {
            System.out.println( "Usage: java BasicDownloadExample URL username password
c:\\temp\\sample.nrq" );
            System.out.println( "Eg: java BasicDownloadExample
https://fsgtest6.finra.org/ BD# BD's-password c:\\temp\\sample.nrq");
            System.exit( 0 );
      }

        URL aURL = new URL(args[0]);
      String username = args[1];
      String password = args[2];
      File file = new File(args[3]);

        System.out.println("protocol = " + aURL.getProtocol());
      System.out.println("authority = " + aURL.getAuthority());
        System.out.println("host = " + aURL.getHost());
        System.out.println("path = " + aURL.getPath());
        System.out.println("filename = " + aURL.getFile());

      // Convert URL to String
      String url = (String)aURL.toString();

      // Create new HttpClient Object
```

```
        HttpClient client = new HttpClient();

        // pass our credentials to HttpClient, they will only be used for
        // authenticating to servers with realm "FileDriveWWW" on the host
        // aURL.getHost(), to authenticate against
        // an arbitrary realm or host change the appropriate argument to null.
        client.getState().setCredentials(
            new AuthScope(aURL.getHost(), 443, "FileDriveWWW"),
            new UsernamePasswordCredentials(username, password)
        );

        // create a GET method over HTTPS, we're assuming
        // that this file requires basic authentication using the realm above.
        GetMethod get = new GetMethod("https://" + aURL.getHost() + "/incoming/?T");

      // create a MultipartPost method for preparing for the nrq file upload
        MultipartPostMethod httppost = new MultipartPostMethod("https://" +
aURL.getHost() + "/incoming/");

        try {
            // execute the GET
            int status = client.executeMethod( get );

            // print the status and response
      //  System.out.println(status + "\n" + get.getResponseBodyAsString());

          // Add parameters for the HTTP header
          httppost.addParameter("USER-AGENT", "Mozilla/4.0");

         //Add parameters for the file data header and file content
         httppost.addParameter("Content-type", "multipart/form-data; boundary=----------
----------7d338d12c0614");
         httppost.addParameter("File", "");
         httppost.addParameter(file.getName(), file);

           // Post data to URL
           int status1 = client.executeMethod( httppost );
           if (httppost.getStatusCode() == 302) { //  HTTP/1.1 302 Found
            System.out.println("\nFile " + file.getName() + " successfully uploaded.\n"
);
           } else {
                     System.out.println("Return Code: " +
httppost.getStatusLine().toString());
           }

        } finally {
            // release any connection resources used by the method
          GetMethod logout = new
GetMethod(aURL.getProtocol()+"://"+aURL.getHost()+"/?logout");
          client.executeMethod( logout );
            httppost.releaseConnection();
        }
    }
}
```

5.3 Perl download sample:

```
#!/usr/bin/perl -w
######################################################################
# Copyright 2008, FINRA
#
# File: firm_download_test_file.pl
#
#
# Purpose:
#     Example program on how to interact with WebEFT to
#     view files in a directory or to download files.
#
# No warranties are expressed or implied.
# This example is UNSUPPORTED and is only intended as an example
# on how to interact with WebEFT.
#
# This requires knowledge of Perl and the libraries below, in
# particular Mechanize, and some encryption libraries for HTTPS
#
# See cpan.org for more information about the libraries.
#
#
# Usage:
#
# firm_download_test_file.pl [url] [BD#] [Password]
# - If the [url] is a directory, this will get the listing of reports
#   in the path.
#
# firm_download_test_file.pl [url] [BD#] [Password] [local_file_name]
# - If the [url] is a filename, the filename will be downloaded as the
#   [local_file_name]
#
# [url] for production is fsg.finra.org or some path using that.
#     fsg.finra.org
#     fsg.finra.org/reports
#     fsg.finra.org/reports/filename.zip
#
#
# Example:
# firm_download_test_file.pl https://fsg.finra.org BD12345 secret_password
#  - Get the listing of files in the main directory
# firm_download_test_file.pl https://fsg.finra.org/reports/example.zip BD12345
secret_password file.zip
#  - Download the example.zip file from the reports directory and save it as file.zip
locally.
#
#
######################################################################
use strict;

use WWW::Mechanize;
use HTML::TokeParser;
use IO::Socket::SSL;
use Net::SSLeay;
use HTTP::Cookies;
use LWP::UserAgent;
use HTTP::Request::Common;

#
# Check for Arguments - Less than 3 arguments?
#
```

```
if ((++$#ARGV) < "3") {
      print "To get a listing of home directory of BD###\n";
      print "\t$0 https://fsgtest6.finra.org BD### BD###'s_Password \n";
      print "To get a listing of report directory of BD###\n";
      print "\t$0 https://fsgtest6.finra.org/reports BD### BD###'s_Password\n\n";
      print "To get a listing of report directory of BD###\n";
      print "\t$0 https://fsgtest6.finra.org/reports/filename.zip BD###
BD###'s_Password\n\n";
      print "To save report to a specific file:\n";
      print "\t$0 https://fsgtest6.finra.org/reports/filename.zip BD### BD###'s_Password
c:\\filename.zip";

      # --- Exit
      exit 2;
}

#################### --- CONFIGURATION --- ##############################
#
my $SITE = $ARGV[0];
my $USER = $ARGV[1];
my $PASSWORD = $ARGV[2];
my $saved_filename = $ARGV[3];
my $STInterface=1;                  # to turnoff ST interface (will be slower and have
html tags)
#
########################################################################

#
# --- How long to wait for the script to run.
#
my $TIMEOUT = 60;

#
# --- Return codes for the script.
#
my $CRITICAL = 2;
my $WARNING = 1;
my $OK = 0;

#
# --- No errors found yet.
#
my $errors_found = "Failed on ";
my $error_code = $OK;

#
# --- Configure timeout to occur if it takes too long so the
#     script doesn't run forever.
#
$SIG{'ALRM'} = sub {
      print "Taking too long to complete.  Timed out after $TIMEOUT seconds\n";
      print "Failed";
      exit $CRITICAL;
};
alarm ($TIMEOUT);

#
# --- Create our agent and make sure it uses cookies.
#
my $agent = WWW::Mechanize->new ();
$agent->agent_alias ('Windows IE 6');
my $cookie_jar = HTTP::Cookies->new (file => "cookie.txt", autosave => 1);
```

```
$agent->cookie_jar ($cookie_jar);

#
# --- Get the filename / directory  - this will require authentication
#
my $response = $agent->get ("$SITE");
if (! $agent->success) {
      print ("Can't get to login page " . $agent->success . "\n");
      print "Failed.";
      exit $CRITICAL;
}

#
# --- Check if ST interface is turned on (1 => no HTML - faster)
#
if ($STInterface == 1) {
      $agent->add_header (USER_AGENT => 'SecureTransport');
}
#
# --- We should be redirected to this login page
#     Fill in the user and password and click on the
#     button.
#
$response = $agent->submit_form (
      form_number => 1,
      fields => { 'user' => $USER, 'password' => $PASSWORD, },
);

#
# --- Make sure we can log in.
#
if (! $agent->success) {
      print ("Can't get to login verification");
      print "Failed";
      exit $CRITICAL;
}

#
# --- Get the filename / directory again now that we've logged in
#
$response = $agent->get ("$SITE");
if (! $agent->success) {
      print ("Can't get to file");
      print "Failed";
      exit $CRITICAL;
}

#
# --- If we get a filename, save it as a file
#
if ($saved_filename) {
      $agent->save_content ($saved_filename);
      print ("WebEFT file saved to $saved_filename\n");
} else {
      #
      # --- Save the file - put it someplace meaningful.  If this is a directory,
      #     it'll be the listing of the files in the directory.
      #
      $agent->save_content ('/tmp/listing.log');

      #
      # Yay!  Report success.
```

```
      #
      print ("WebEFT file or directory listing downloaded and saved in
/tmp/listing.log.\n");
}

#
# --- Logout - please log out
#
$response = $agent->get ("$SITE/?logout");
if (! $agent->success) {
      print ("Bah, failed logout");
      print "Failed";
      exit $CRITICAL;
}

exit $OK;
```

5.4 Perl upload sample:

```
#!/usr/bin/perl -w
######################################################################
#
# firm_submit_test_filing.pl
#
# An example program on how to submit a test filing to Web EFT.
# Disclaimer: This is an example of perl script that was developed for an internal proof-
of-concept test.
# The script did work at that time, running on internal FINRA machines within the FINRA
network.
#  We cannot guarantee that the script as written will work on any given platform or
machine.
#
# This requires the libraries below, in particular Mechanize,
# and some encryption libraries for HTTPS
#
# See cpan.org for more information.
#
######################################################################
use strict;

use WWW::Mechanize;
use HTML::TokeParser;
use IO::Socket::SSL;
use Net::SSLeay;
use HTTP::Cookies;
use LWP::UserAgent;
use HTTP::Request::Common;

#################### --- CONFIGURATION --- ##############################
#
# --- User/password into the system.
#
#my $USER = 'BDXXXX';
#my $PASSWORD = 'SECRET PASSWORD GOES HERE';

#my $SITE = "fsg.finra.org";        #--- PRODUCTION
my $SITE = "fsgtest6.finra.org";          #--- FIRM TESTING
#
######################################################################

#
# --- String to search for.
#
my $filename = $ARGV[0];
my $search_string = "incoming";

#
# --- How long to wait for the script to run.
#
my $TIMEOUT = 60;

#
# --- Return codes for the script.
#
my $CRITICAL = 2;
my $WARNING = 1;
my $OK = 0;

#
```

```
# --- No errors found yet.
#
my $errors_found = "Failed on ";
my $error_code = $OK;


#
# --- Configure timeout to occur if it takes too long so the
#     script doesn't run forever.
#
$SIG{'ALRM'} = sub {
      print "Taking too long to complete.  Timed out after $TIMEOUT seconds\n";
      print "Failed";
      exit $CRITICAL;
};
alarm ($TIMEOUT);

#
# --- Create our agent and make sure it uses cookies.
#
my $agent = WWW::Mechanize->new ();
$agent->agent_alias ('Windows IE 6');
my $cookie_jar = HTTP::Cookies->new (file => "cookie.txt", autosave => 1);
$agent->cookie_jar ($cookie_jar);

#
# --- Get the home page
#
my $response = $agent->get ("https://$SITE/");
if (! $agent->success) {
      print ("Can't get to login page " . $agent->success . "\n");
      print "Failed.";
      exit $CRITICAL;
}

#
# --- We should be redirected to this login page
#     Fill in the user and password and click on the
#     button.
#
$response = $agent->submit_form (
      form_number => 1,
      fields => { 'user' => $USER, 'password' => $PASSWORD, },
);

if (! $agent->success) {
      print ("Can't get to login verification");
      print "Failed";
      exit $CRITICAL;
}

#
# --- Check the content to verify we're on the correct page.
#
if ($agent->content =~ /$search_string/ ) {

} else {

      print "Could not match string after logging in.";
      print "Failed";
      exit $CRITICAL;
}
```

```
#
# --- Go to the file upload page
#
$response = $agent->get ("https://$SITE/incoming/?T");
if (! $agent->success) {
      print ("Can't get to file upload page");
      print "Failed";
      exit $CRITICAL;
}

#
# --- Upload the file
#
$response = $agent->request (POST "https://$SITE/incoming/",
                        Content => [
                              'File' => [$filename],
#                             'File' => ["test1.nrq"],
                        ],
                        'Content_Type' => 'form-data',
                    );

if (! $agent->success) {
      print ("Failed to post WebEFT file.\n");
      print "Failed";
      exit $CRITICAL;
}

#
# Yay!  Success.
#
print ("WebEFT file successfully uploaded.\n");

#
# Need to add code to logout
#
exit $OK;
```

5.5 VBS download sample:

```
'''Disclaimer: This is an example of vbs script that was developed for an internal proof-
of-concept test.
'''The script did work at that time, running on internal FINRA machines within the FINRA
network.
'''We cannot guarantee that the script as written will work on any given platform or
machine.
'''
'''To run from command prompt, type cscript download-example.vbs status.xml
'''Uses MSXML2.XMLHTTP library to interact with website
'''Uses ADODB.STREAM to save the webserver response for either directory listing or file
content to localdisk
'''For example:
'''c:\>cscript download-example.vbs archive/test.20070702.210651.nrq
'''archive-test.20070702.210651.nrq is downloaded to c:\temp\
'''

''' MAIN'''

''' Set some variables
''' Specify directory to download
sDestFolder = "c:\temp\"
'''strURL = "https://fsg.finra.org/" '''Production
strURL = "https://fsgtest6.finra.org/" '''QC
ID="BD#"
IDPassword="passwd"

''' Check for argument'''
If Wscript.Arguments.count <= 0 Then
      wscript.echo "Please provide following as the argument. For example:"
      wscript.echo "download-example.vbs status.xml"
      wscript.echo "download-example.vbs archive/"
      wscript.echo "download-example.vbs processed/"
      wscript.echo "download-example.vbs reports/"
      wscript.echo "download-example.vbs archive/filename-in-archive.ext"
      wscript.Quit 1
end if

'''Get the URL from the standard input'''
sImageFile=WScript.Arguments(0)

'''Call function to get the directory listing or the file found in directory listing from
URL
GetHttpFile sDestFolder,strURL,sImageFile


function GetHttpFile(argDestFolder,argSrcUrl,argImageFile)

      dim xmlhttp
      set xmlhttp = CreateObject("MSXML2.XMLHTTP")
      '''Initializes an MSXML2.XMLHTTP request and specifies the method, URL, and
authentication information '
      '''for the request '''
       xmlhttp.open "GET", argSrcUrl & "/template/login", False, ID, IDPassword

      '''Specifies the name of an HTTP header
      xmlhttp.setRequestHeader "USER-AGENT", "SecureTransport"

      '''Sends an HTTP request to the server with authentication information and receives
a response
      xmlhttp.send "username=ID&password=IDPassword"
```

```
      '''If successfully logged in,
      if xmlhttp.status = "200" then
            xmlhttp.open "GET", argSrcUrl & sImageFile, False
            xmlhttp.setRequestHeader "USER-AGENT", "SecureTransport"
            xmlhttp.send

            '''Remove the / and ? from the URL for the filename to be saved to localdisk
            argImageFile = replace(argImageFile, "/", "-")
            argImageFile = replace(argImageFile, "?", "")

            '''Initializes an ADODB.STREAM object for saving the https response content
to localdisk
            set oStream = createobject("ADODB.STREAM")
            Const adTypeBinary = 1
            Const adSaveCreateOverWrite = 2
            oStream.type = adTypeBinary
            oStream.open

            '''Saving the https response content to localdisk
            oStream.write xmlhttp.responseBody
            oStream.savetofile argDestFolder & argImageFile, adSaveCreateOverWrite

            wscript.echo argImageFile & " is downloaded to " & argDestFolder

            '''Close connections when done
            xmlhttp.open "GET", argSrcUrl & "?logout", False
            set oStream = nothing
            set xmlhttp = nothing
      else
            wscript.echo "Authentication failed"
            set oStream = nothing
            set xmlhttp = nothing
            exit function
      end if

end function
```

5.6 VBS upload sample:

```
'''Disclaimer: This is an example of vbs script that was developed for an internal proof-
of-concept test.
'''The script did work at that time, running on internal FINRA machines within the FINRA
network.
'''We cannot guarantee that the script as written will work on any given platform or
machine.
'''
'''To run from command prompt, type
'''   cscript upload-example.vbs sample.nrq
'''Uses MSXML2.XMLHTTP library to interact with website
'''Uses ADODB.STREAM to save the webserver response for either directory listing or file
content to localdisk
'''For example:
'''c:\>cscript upload-example.vbs c:\sample.nrq
'''sample.10-103023.nrq successfully uploaded to https://fsg.finra.org/
'''
'''

'''MAIN'''
dim strURL
'''strURL = "https://fsg.finra.org" ''' Production
strURL = "https://fsgtest6.finra.org/"  ''' QC

'''set sType to ascii'''
sType=2

'''Set Username and password
ID="BD#"
IDPassword="BD#'s password"

'''Set Path and nrq filename '''
''' Check for argument'''
If Wscript.Arguments.count <= 0 Then
      wscript.echo "Please provide the path and nrq filename argument. For example:"
      wscript.echo "upload-example.vbs c:\sample.nrq"
      wscript.Quit 1
end if

'''Get the URL from the standard input'''
sfileName=WScript.Arguments(0)

'''Call the function sendit with filename and filetype as parameters
call sendit(sfileName, sType)

function sendit( sfileName, sType)

      '''Read contents of drive:path\filename in ascii format'''
      Data = getFileBytes(sfileName, sType)

      sDate = Day (Date) & "-" & hour(now) & minute(now) & second(now)  ''' Get the Day-
HHMMSS

      ''' Get basename of file: eg sample.nrq
      basefileName= mid(sfileName, InstrRev(sFileName,"\")+1,len(sfileName))

      ''' Strip the extension: eg sample
      basefileName = left(basefileName, InstrRev(basefileName, "."))

      '''Create the sData data object, with header, unique filename eg: sample.Day-
HHMMSS.nrq and data, '''
```

```
      '''separated by sBoundary, to be uploaded'''
      sBoundary = "--------------------------7d338d12c0614"
      sData = "--" & sBoundary & vbCrLf
      sData = sData & _
                "Content-Disposition: form-data;" & _
                " name=""File""; filename=""" & _
                basefileName & sDate & ".nrq" & """" & vbCrLf
      sData = sData & Data & vbCrLf

      set xmlhttp = CreateObject("MSXML2.XMLHTTP")
      '''Initializes an MSXML2.XMLHTTP request and specifies the method, URL, and
authentication information '
      '''for the request '''
      xmlhttp.Open "GET", strURL & "/template/login", false, ID, IDPassword

      '''Specifies the name of an HTTP header
      xmlhttp.setRequestHeader "USER-AGENT", "SecureTransport"

      '''Sends an HTTP request to the server with authentication information and receives
a response
      xmlhttp.send "username=ID&password=IDPassword"

      '''If successfully logged in, get ready to post nrq file to /incoming
      if xmlhttp.status = "200" then
            xmlhttp.Open "POST", strURL & "/incoming", false

            '''Specifies the name of an HTTP header
            xmlhttp.setRequestHeader "USER-AGENT", "SecureTransport"
            xmlhttp.setRequestHeader "Content-Type", "multipart/form-data; boundary=" &
sBoundary

            '''Send the sData data object header, boundary and content of nrq file
            xmlhttp.Send sData

            if xmlhttp.status = "200" then
                  wscript.echo basefileName & sDate & ".nrq" & " successfully uploaded to
" & strURL
            else
                  wscript.echo basefileName & sDate & ".nrq" & " NOT uploaded."
            End if
      else
            wscript.echo "Failed login to " & strURL & ". Please check ID and password."
      END if

      '''Close connection
      xmlhttp.Open "GET", strURL & "/?logout", false
      set xmlhttp=Nothing
End function

function getFileBytes(flnm, sType)

      Dim objStream
      Set objStream = CreateObject("ADODB.STREAM")
      if sType="on" then
            objStream.Type = 1 ' adTypeBinary
      else
            objStream.Type = 2 ' adTypeText
            objStream.Charset ="ascii"
      end if
      objStream.Open
      objStream.LoadFromFile flnm
      if sType="on" then
```

```
            getFileBytes=objStream.Read
      else
            getFileBytes= objStream.ReadText
      end if
      objStream.Close
      Set objStream = Nothing
end function  ''' getFileBytes
```

5.7 C# script sample for .net:

```
using System;
using System.Net;
using System.Text;
using System.IO;
using System.Windows.Forms;

namespace TestEFT_NET
{
	/// <summary>
	/// Summary description for frmMain.
	/// </summary>
	public class frmMain : System.Windows.Forms.Form, IWin32Window
	{
	  #region Member Variables

	  private System.Windows.Forms.GroupBox groupBox1;
	  private System.Windows.Forms.GroupBox groupBox2;
	  private System.Windows.Forms.GroupBox groupBox3;
	  private System.Windows.Forms.Label label1;
	  private System.Windows.Forms.Label label2;
	  private System.Windows.Forms.Label label3;
	  private System.Windows.Forms.Label label4;
	  private System.Windows.Forms.Label label5;
	  private System.Windows.Forms.Label label6;
	  private System.Windows.Forms.OpenFileDialog openFileDialog1;
	  private System.Windows.Forms.SaveFileDialog saveFileDialog1;
	  private System.Windows.Forms.TextBox txtHostName;
	  private System.Windows.Forms.TextBox txtUserName;
	  private System.Windows.Forms.TextBox txtPassword;
	  private System.Windows.Forms.TextBox txtUploadFilename;
	  private System.Windows.Forms.Button cmdUploadBrowse;
	  private System.Windows.Forms.Button cmdUpload;
	  private System.Windows.Forms.TextBox txtDownLoadFileName;
	  private System.Windows.Forms.Button cmdDownloadBrowse;
	  private System.Windows.Forms.TextBox txtSaveFileName;
	  private System.Windows.Forms.Button cmdDownload;
	  private System.Windows.Forms.Button cmdCancel;

	  /// <summary>
	      /// Required designer variable.
	      /// </summary>
	      private System.ComponentModel.Container components = null;

	  #endregion

	  #region Construction/Destruction

	      public frmMain()
	      {
	            //
	            // Required for Windows Form Designer support
	            //
	            InitializeComponent();

	            //
	            // TODO: Add any constructor code after InitializeComponent call
	            //
	      }

	      /// <summary>
```

```
            /// Clean up any resources being used.
            /// </summary>
            protected override void Dispose( bool disposing )
            {
                  if( disposing )
                  {
                        if (components != null)
                        {
                              components.Dispose();
                        }
                  }
                  base.Dispose( disposing );
            }

        #endregion

            #region Windows Form Designer generated code
            /// <summary>
            /// Required method for Designer support - do not modify
            /// the contents of this method with the code editor.
            /// </summary>
            private void InitializeComponent()
            {
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.label3 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.label1 = new System.Windows.Forms.Label();
            this.txtPassword = new System.Windows.Forms.TextBox();
            this.txtUserName = new System.Windows.Forms.TextBox();
            this.txtHostName = new System.Windows.Forms.TextBox();
            this.groupBox2 = new System.Windows.Forms.GroupBox();
            this.cmdUpload = new System.Windows.Forms.Button();
            this.cmdUploadBrowse = new System.Windows.Forms.Button();
            this.txtUploadFilename = new System.Windows.Forms.TextBox();
            this.label4 = new System.Windows.Forms.Label();
            this.groupBox3 = new System.Windows.Forms.GroupBox();
            this.txtDownLoadFileName = new System.Windows.Forms.TextBox();
            this.label6 = new System.Windows.Forms.Label();
            this.label5 = new System.Windows.Forms.Label();
            this.cmdDownloadBrowse = new System.Windows.Forms.Button();
            this.txtSaveFileName = new System.Windows.Forms.TextBox();
            this.cmdDownload = new System.Windows.Forms.Button();
            this.cmdCancel = new System.Windows.Forms.Button();
            this.openFileDialog1 = new System.Windows.Forms.OpenFileDialog();
            this.saveFileDialog1 = new System.Windows.Forms.SaveFileDialog();
            this.groupBox1.SuspendLayout();
            this.groupBox2.SuspendLayout();
            this.groupBox3.SuspendLayout();
            this.SuspendLayout();
            //
            // groupBox1
            //
            this.groupBox1.Controls.AddRange(new System.Windows.Forms.Control[] {

this.label3,

this.label2,

this.label1,

this.txtPassword,
```

```
this.txtUserName,

this.txtHostName});
            this.groupBox1.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.groupBox1.Location = new System.Drawing.Point(8, 8);
            this.groupBox1.Name = "groupBox1";
            this.groupBox1.Size = new System.Drawing.Size(496, 120);
            this.groupBox1.TabIndex = 0;
            this.groupBox1.TabStop = false;
            this.groupBox1.Text = "General Section";
            //
            // label3
            //
            this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label3.Location = new System.Drawing.Point(16, 88);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(64, 16);
            this.label3.TabIndex = 4;
            this.label3.Text = "Password";
            //
            // label2
            //
            this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label2.Location = new System.Drawing.Point(16, 56);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(64, 16);
            this.label2.TabIndex = 2;
            this.label2.Text = "Username";
            //
            // label1
            //
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label1.Location = new System.Drawing.Point(16, 24);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(64, 16);
            this.label1.TabIndex = 0;
            this.label1.Text = "Hostname";
            //
            // txtPassword
            //
            this.txtPassword.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
            this.txtPassword.Location = new System.Drawing.Point(88, 88);
            this.txtPassword.Name = "txtPassword";
            this.txtPassword.TabIndex = 5;
            this.txtPassword.Text = "Eagle12$";
            //
            // txtUserName
            //
            this.txtUserName.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
            this.txtUserName.Location = new System.Drawing.Point(88, 56);
            this.txtUserName.Name = "txtUserName";
            this.txtUserName.TabIndex = 3;
            this.txtUserName.Text = "monitor";
```

```
            //
            // txtHostName
            //
            this.txtHostName.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
            this.txtHostName.Location = new System.Drawing.Point(88, 24);
            this.txtHostName.Name = "txtHostName";
            this.txtHostName.TabIndex = 1;
            this.txtHostName.Text = "63.251.86.151";
            //
            // groupBox2
            //
            this.groupBox2.Controls.AddRange(new System.Windows.Forms.Control[] {

this.cmdUpload,

this.cmdUploadBrowse,

this.txtUploadFilename,

this.label4});
            this.groupBox2.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.groupBox2.Location = new System.Drawing.Point(8, 136);
            this.groupBox2.Name = "groupBox2";
            this.groupBox2.Size = new System.Drawing.Size(496, 96);
            this.groupBox2.TabIndex = 1;
            this.groupBox2.TabStop = false;
            this.groupBox2.Text = "Upload Section";
            //
            // cmdUpload
            //
            this.cmdUpload.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.cmdUpload.Location = new System.Drawing.Point(8, 56);
            this.cmdUpload.Name = "cmdUpload";
            this.cmdUpload.Size = new System.Drawing.Size(480, 32);
            this.cmdUpload.TabIndex = 3;
            this.cmdUpload.Text = "&Upload File";
            this.cmdUpload.Click += new System.EventHandler(this.cmdUpload_Click);
            //
            // cmdUploadBrowse
            //
            this.cmdUploadBrowse.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
            this.cmdUploadBrowse.Location = new System.Drawing.Point(408, 24);
            this.cmdUploadBrowse.Name = "cmdUploadBrowse";
            this.cmdUploadBrowse.Size = new System.Drawing.Size(80, 23);
            this.cmdUploadBrowse.TabIndex = 2;
            this.cmdUploadBrowse.Text = "&Browse...";
            this.cmdUploadBrowse.Click += new
System.EventHandler(this.cmdUploadBrowse_Click);
            //
            // txtUploadFilename
            //
            this.txtUploadFilename.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
            this.txtUploadFilename.Location = new System.Drawing.Point(160, 24);
            this.txtUploadFilename.Name = "txtUploadFilename";
```

```
            this.txtUploadFilename.Size = new System.Drawing.Size(240, 20);
            this.txtUploadFilename.TabIndex = 1;
            this.txtUploadFilename.Text = "";
            //
            // label4
            //
            this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label4.Location = new System.Drawing.Point(8, 24);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(152, 16);
            this.label4.TabIndex = 0;
            this.label4.Text = "Upload filename with path";
            //
            // groupBox3
            //
            this.groupBox3.Controls.AddRange(new System.Windows.Forms.Control[] {

this.txtDownLoadFileName,

this.label6,

this.label5,

this.cmdDownloadBrowse,

this.txtSaveFileName,

this.cmdDownload});
            this.groupBox3.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.groupBox3.Location = new System.Drawing.Point(8, 240);
            this.groupBox3.Name = "groupBox3";
            this.groupBox3.Size = new System.Drawing.Size(496, 128);
            this.groupBox3.TabIndex = 2;
            this.groupBox3.TabStop = false;
            this.groupBox3.Text = "Download Section";
            //
            // txtDownLoadFileName
            //
            this.txtDownLoadFileName.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
            this.txtDownLoadFileName.Location = new System.Drawing.Point(160, 24);
            this.txtDownLoadFileName.Name = "txtDownLoadFileName";
            this.txtDownLoadFileName.Size = new System.Drawing.Size(328, 20);
            this.txtDownLoadFileName.TabIndex = 1;
            this.txtDownLoadFileName.Text = "status.xml";
            //
            // label6
            //
            this.label6.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label6.Location = new System.Drawing.Point(8, 56);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(152, 16);
            this.label6.TabIndex = 2;
            this.label6.Text = "Save filename with path";
            //
            // label5
            //
```

```
            this.label5.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.label5.Location = new System.Drawing.Point(8, 24);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(152, 16);
            this.label5.TabIndex = 0;
            this.label5.Text = "Download filename with path";
            //
            // cmdDownloadBrowse
            //
            this.cmdDownloadBrowse.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
            this.cmdDownloadBrowse.Location = new System.Drawing.Point(408, 56);
            this.cmdDownloadBrowse.Name = "cmdDownloadBrowse";
            this.cmdDownloadBrowse.Size = new System.Drawing.Size(80, 23);
            this.cmdDownloadBrowse.TabIndex = 4;
            this.cmdDownloadBrowse.Text = "B&rowse...";
            this.cmdDownloadBrowse.Click += new
System.EventHandler(this.cmdDownloadBrowse_Click);
            //
            // txtSaveFileName
            //
            this.txtSaveFileName.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
            this.txtSaveFileName.Location = new System.Drawing.Point(160, 56);
            this.txtSaveFileName.Name = "txtSaveFileName";
            this.txtSaveFileName.Size = new System.Drawing.Size(240, 20);
            this.txtSaveFileName.TabIndex = 3;
            this.txtSaveFileName.Text = "";
            //
            // cmdDownload
            //
            this.cmdDownload.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
            this.cmdDownload.Location = new System.Drawing.Point(8, 88);
            this.cmdDownload.Name = "cmdDownload";
            this.cmdDownload.Size = new System.Drawing.Size(480, 32);
            this.cmdDownload.TabIndex = 5;
            this.cmdDownload.Text = "&Download File";
            this.cmdDownload.Click += new System.EventHandler(this.cmdDownload_Click);
            //
            // cmdCancel
            //
            this.cmdCancel.DialogResult = System.Windows.Forms.DialogResult.Cancel;
            this.cmdCancel.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
            this.cmdCancel.Location = new System.Drawing.Point(8, 376);
            this.cmdCancel.Name = "cmdCancel";
            this.cmdCancel.Size = new System.Drawing.Size(496, 32);
            this.cmdCancel.TabIndex = 3;
            this.cmdCancel.Text = "&Cancel";
            this.cmdCancel.Click += new System.EventHandler(this.cmdCancel_Click);
            //
            // saveFileDialog1
            //
            this.saveFileDialog1.AddExtension = false;
            //
            // frmMain
            //
```

```
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.CancelButton = this.cmdCancel;
            this.ClientSize = new System.Drawing.Size(512, 413);
            this.Controls.AddRange(new System.Windows.Forms.Control[] {
                                                                        this.groupBox3,
                                                                        this.groupBox2,
                                                                        this.groupBox1,

this.cmdCancel});
            this.Name = "frmMain";
            this.Text = "Test EFT .NET";
            this.Load += new System.EventHandler(this.frmMain_Load);
            this.groupBox1.ResumeLayout(false);
            this.groupBox2.ResumeLayout(false);
            this.groupBox3.ResumeLayout(false);
            this.ResumeLayout(false);

        }
            #endregion

        #region Main Entry Function

            /// <summary>
            /// The main entry point for the application.
            /// </summary>
            [STAThread]
            static void Main()
            {
                  Application.Run(new frmMain());
            }

        #endregion

        #region Member Functions

        private bool ValidateLoginInfo()
        {
            // Validate that host name is available
            if( txtHostName.Text.Length == 0 )
            {
                MessageBox.Show( this, "Please enter a hostname.", "Data Error",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation );
                txtHostName.Focus();
                return false;
            }

            // Validate that user name is available
            if( txtUserName.Text.Length == 0 )
            {
                MessageBox.Show( this, "Please enter a usename.", "Data Error",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation );
                txtUserName.Focus();
                return false;
            }

            //Validate that password is available
            if( txtPassword.Text.Length == 0 )
            {
                MessageBox.Show( this, "Please enter a password.", "Data Error",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation );
                txtPassword.Focus();
                return false;
```

```
            }

            // No error found
            return true;
        }

        private string Login()
        {
            string cookie;

            try
            {
                // Login
                WebClient client = new WebClient();
                client.Credentials = new NetworkCredential( txtUserName.Text,
txtPassword.Text );
                client.Headers.Add( "User-Agent", "none");

                client.DownloadData( "https://" + txtHostName.Text );

                // Save the cookie
                cookie = client.ResponseHeaders.GetValues( "Set-Cookie" )[0];
                cookie = cookie.Split( ";".ToCharArray(), 2 )[0];
            }
            catch( WebException ex )
            {
                MessageBox.Show( this, ex.Message, "Logon Error", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation );
                cookie = "";
            }

            return cookie;
        }

        #endregion

        #region Event Handlers

        private void frmMain_Load(object sender, System.EventArgs e)
        {

        }

        private void cmdUpload_Click(object sender, System.EventArgs e)
        {
            if( false == ValidateLoginInfo() )
            {
                return;
            }

            if( txtUploadFilename.Text.Length == 0 )
            {
                MessageBox.Show( this, "Please enter a filename to upload.", "Data
Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation );
                txtUploadFilename.Focus();
                return;
            }

            // Indicate that processing is occurring
            Cursor.Current = Cursors.WaitCursor;

            string cookie = Login();
```

```
            if( cookie.Length > 0 )
            {
                try
                {
                    WebClient client = new WebClient();
                    client.Headers.Add( "User-Agent", "Mozilla/4.0");
                    client.Headers.Add( "Accept", "*/*" );
                    client.Headers.Add( "Cookie", cookie );

                    FileInfo infoFile = new FileInfo( txtUploadFilename.Text );
                    client.UploadFile(
                        "https://" + txtHostName.Text + "/incoming/" + infoFile.Name,
                        "POST", txtUploadFilename.Text );

                    MessageBox.Show( this, "File " + infoFile.Name + " upload
successful.", "Upload Success", MessageBoxButtons.OK, MessageBoxIcon.Information );
                }
                catch( WebException ex )
                {
                    MessageBox.Show( this, ex.Message, "Upload Error",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation );
                }
                finally
                {
                    // Turn off the wait cursor
                    Cursor.Current = Cursors.Default;
                }
            }
        }

        private void cmdDownload_Click(object sender, System.EventArgs e)
        {
            if( false == ValidateLoginInfo() )
            {
                return;
            }

            if( txtDownLoadFileName.Text.Length == 0 )
            {
                MessageBox.Show( this, "Please enter a filename to download.", "Data
Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation );
                txtDownLoadFileName.Focus();
                return;
            }

            if( txtSaveFileName.Text.Length == 0 )
            {
                MessageBox.Show( this, "Please enter a filename to save the downloaded
file.", "Data Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation );
                txtSaveFileName.Focus();
                return;
            }

            // Indicate that processing is occurring
            Cursor.Current = Cursors.WaitCursor;

            string cookie = Login();

            if( cookie.Length > 0 )
            {
                try
```

```
                {
                    WebClient client = new WebClient();
                    client.Headers.Add( "User-Agent", "Mozilla/4.0" );
                    client.Headers.Add( "Accept", "*/*" );
                    client.Headers.Add( "Cookie", cookie );

                    string sURL = "https://" + txtHostName.Text + "/" +
txtDownLoadFileName.Text;
                    client.DownloadFile( sURL, txtSaveFileName.Text );

                    MessageBox.Show( this, "Download of '" + sURL + "' was successful.",
"Download Success", MessageBoxButtons.OK, MessageBoxIcon.Information );
                }
                catch( WebException ex )
                {
                    MessageBox.Show( this, ex.Message,"Download Error",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation );
                }
                finally
                {
                    // Turn off the wait cursor
                    Cursor.Current = Cursors.Default;
                }
            }
        }

        private void cmdUploadBrowse_Click(object sender, System.EventArgs e)
        {
            openFileDialog1.ShowDialog(this);
            txtUploadFilename.Text = openFileDialog1.FileName;
        }

        private void cmdDownloadBrowse_Click(object sender, System.EventArgs e)
        {
            saveFileDialog1.ShowDialog(this);
            txtSaveFileName.Text = saveFileDialog1.FileName;
        }

        private void cmdCancel_Click(object sender, System.EventArgs e)
        {
            Application.Exit();
        }

        #endregion
      }
}
```