**ERA FTP Service**

2012-01-30

Michael Spence

**Audience:**

This documentation is provided as a configuration and usage guide. This could be useful for System Engineers or Application Developers.

**Preamble:**

This document describes the requirements for this FTP service. The solution described, adheres to these requirements.

This document describes the development and implementation of the solution for this FTP service. It DOES NOT describe possible other solutions which may have fit this purpose.

This document outlines the usage of this service, but DOES NOT specify exactly how this is to be performed.

**Topics of Discussion:**

1- Systems Requirements

2- Proposed Solution

3- OpenBSD Configuration

4- FTP Service Configuraion

5- Secure Copy configuration

6- Data Checking Script

a. Constants

b. File Processing

c. Process Logs

d. Cron Setup

7- Usage

Z- Procftp.sh

**Topics in Detail:**

**1- Systems Requirements**

This system is to be configured so that data can be transferred between Contact Tracker and the two ERA systems. The transport method available from Contact Tracker is FTP. The FTP method is required to be PASSIVE due to the configuration of the APEagers firewall.

Some data checking is required to verify destination server and data integrity. This check is not a thourough XML data integrity check.

A transport method is also required between this FTP server and the ERA systems. Available methods include EMAIL, FTP, SFTP, SCP, SMB (potentially).

All transactions performed on a file are to be logged.

Potentially, this system could also transfer data to other ERP systems, and hence needs to be extendible. This section below "Data Checking Script" demonstrates how this could be achieved, although differing transfer methods may be required.

Currently the specification has been defined as:

<?TEST XML FILE>

<LEAD.[server]>

<[data]>....</[data]>

.

.

</LEAD.[server]>

where "server" is the desitnation server identifier, currently known as 894 and 7312. And "data" is any appropriate XML data, not applicable to this application.

**2- Proposed Solution**

The author's proposed solution uses an OpenBSD server, configured to accept FTP transfer of files. A cron job is to be configured to run periodically to transfer via automated script, from this server to either ERA server. The transfer from this server to ERA servers is to be performed by SCP.

The script limitations require that all files transferred to this FTP server do not end in an extension of ".done", ".error" or ".fault".

**3- OpenBSD Configuration**

The server used has been installed as a typical installation of OpenBSD as described in "Open BSD Installation Guide" by Michael Spence. No specific services were turned off after installation, including the packet filter, for which rules were not altered.

Two standard users accounts have been added: one for the engineer "mspence" and one for "issadmin"; both accounts have explicit sudoer rights. This was done in accordance with the mentioned documentation.

**4- FTP Service Configuraion**

The FTP service on OpenBSD is a standard service installed at installation. To start this service, flags simply need to be added to the ftpd\_flags variable in the rc.conf configuration file. This variable has been set to "-DllUSA". Which perform the following roles:

D sets ftpd to run as a daemon

ll Logs all successful and failed login attempts, Logs all transactions

U Each concurrent session is logged to the file /var/run/utmp

S All Anonymous downloads are logged to the file /var/log/ftpd

A Allows only anonymous FTP connections

A specific user was added to the system. This account is named "ftp" and has a blank password. This user's home directory (/home/ftp) is the root of the ftp site and due to this requires specific permissions.

Firstly, the daemon will be run as root, which requires that all directories are accessible by root. This is easily achieved by changing the file mode to 555 for this location. This also stops files from being written to the home directory of the ftp user.

Subsequent directories from this location can be given permission with respect to their usage. For this application, the directories "etc", "log" and "pub" have been added. The "etc" is read only and is in place for site information files. The "pub" directory is the destination directory for Contact Tracker to place data files; as such it has permissions set so that users in the group "ftp" can write to it. Finally the "log" directory is to be filled with files by script; this directory needs to be writeable by the user executing the script, and not changeable by users external to the system.

**5- Secure Copy configuration**

By using secure copy, the need for storage of user name and password combinations within a script is removed. To validate credentials, certificates are used instead of passwords.

Simple security tokens can be generated which act as verification between two systems. This has been done for the user executing the processing scripts (at this stage this is the user "mspence") using the RSA security method:

ssh-keygen -t rsa

This creates two token files in the .ssh directory in the user's home path. The "id\_rsa.pub" file contains the matching response token, which must be added to the "authorized\_keys" file on the destination server. As this file is by default empty, this file was simply copied to the "authorized\_keys" files. If for some reason the authorized\_keys file is already populated the following command will append the required line to this file.

cat id\_rsa.pub >> .ssh/authorized\_keys

SCP tests were performed from the FTP server to the ERA servers to confirm that this functionality worked.

scp FROB mspence@172.20.3.34:. (for example)

Permissions for the .ssh directory on target system to be 700 and .ssh/.. to be 755.

**6- Data Checking Script**

The data checking script has been designed to be completely automated. It contains the necessary source and destination directories, matching strings and target servers as constants at the beginning of the file. The script processes all files within the source directory (with certain limitations) and logs all transactions upon these files. Logs are placed in the appropriate destination directory, and when a file has been processed it is renamed to an approprate holding name. To perform this function automatically, the script is added to the CRON jobs for the appropriate user.

**6.a Constants**

The following constants in the script define the Source and Destination directories used by the script.

FTPPUB="/home/ftp/pub"

FTPLOG="/home/ftp/log"

The following constants in the script define the Schema Strings used by the script.

STRSCHEMA="<?test>"

STRSYS1="LEAD.894"

STRSYS2="LEAD.7146"

The following constants in the script define the Target Systems used by the script.

TGTSYS1="172.20.3.34"

TGTSYS2="172.20.3.40"

**6.b File Processing**

The script processes all files in the FTPPUB directory, which do not have the extension of ".done" or ".error".

Acting upone one file at a time, the script verifies that the first line matches the STRSCHEMA string. Failure to match this string, causes the script to stop processing this file, which is then renamed to "[file].error".

After passing the schema match, the script continues to verify that the second and last lines match. This is performed by stripping the "/" from the last line and comparing the two. On an exact match, it is assumed that the file is valid XML. NOTE: this is not a thorough test for XML integrity, in fact it is very poor; this script is not intended to be an XML parser. If these lines do not match in this way, the file is renamed to "[file].error".

Following the XML test, the XML tags are examined. If the tag name matches either STRSYS1 or STRSYS2, the file is SCP'd to the appropriate target system, as defined by TGTSYS1 or TGTSYS2 respectively. If the XML tag does not match these strings, the file is renamed to "[file].error". The destination directory on the target system is

/home/mspence/trx

Finally, the file is renamed to "[file].done".

**6.c Process Logs**

At each process step in the file processing procedure described above, for any processed file, a log is created in FTPLOG. The log file for any processed file will be "[file].log". The following entries in a log file describe the success or failure of the described step.

**LOG ENTRY DESCCRIPTION**

START File processing has started on this file at this time

SCHEMA First line has matched (or not) the schema constant

PRIMARY Second and Last lines have matched (or not)

and (if has matched)

Second line matched (or not) a system constant string

SCOPY File has been (or not) scp'd to target system

FINISH File processing has ended on this file at this time

The logs are inteded to be not writable by anything but the script, however they are visible for verification of transfer though the ftp system.

**6.d Cron Setup**

The crontab for the user who is to execute this script was edited from his profile. This is done using:

crontab -e

The following entry was added to the cron, so that execution of the script was performed every 5 minutes

0-59/5 \* \* \* \* mspence /home/mspence/bin/procftp.sh

**7- Usage**

The entry point for this system is the FTP interface of the server "qld-tftp". The FTP must be configured in passive mode and all files must be placed in the pub directory; other directories are read only, and sub-directories of pub should not be created and will not be processed. The cron job will process the files in the pub directory, periodically (ie every 5 minutes), creating a log of the work performed and copying valid files to the respective ERA servers. The files can then be processed in any manner seen fit from the destination directory on the ERA systems.

**Z- Procftp.sh**

In all its glory... the script...

#!/bin/ksh

FTPPUB="/home/ftp/pub"

FTPLOG="/home/ftp/log"

STRSCHEMA="<?test>"

STRSYS1="LEAD.894"

STRSYS2="LEAD.7146"

TGTSYS1="172.20.3.34"

TGTSYS2="172.20.3.40"

for i in $( ls $FTPPUB | grep -v "\.done" | grep -v "\.error" ); do

a=$( head -1 $FTPPUB/$i )

b=$( head -2 $FTPPUB/$i | tail -1 )

c=$( tail -1 $FTPPUB/$i | sed "s/\///" )

t=$( date )

echo "START: $i at $t" >> $FTPLOG/$i

if [ "x$a" == "x$STRSCHEMA" ]; then

echo "SCHEMA: matched" >> $FTPLOG/$i

if [ "x$b" == "x$c" ]; then

echo "PRIMARY: start and end matched" >> $FTPLOG/$i

d=$( echo $b | sed "s/[<>]//g" )

case "x$d" in

(x$STRSYS1)

echo "PRIMARY: targeted for 894" >> $FTPLOG/$i

scp -B $FTPPUB/$i $TGTSYS1:~/trx/.

r=$?

if [ "x$r" == "x0" ]; then

echo "SCOPY: copy to s1 successful" >> $FTPLOG/$1

mv $FTPPUB/$i "$FTPPUB/${i}.done"

else

echo "SCOPY: copy to s1 successful" >> $FTPLOG/$1

mv $FTPPUB/$i "$FTPPUB/${i}.error"

fi

;;

(x$STRSYS2)

echo "PRIMARY: targeted for 7312" >> $FTPLOG/$i

scp -B $FTPPUB/$i $TGTSYS2:~/trx/.

r=$?

if [ "x$r" == "x0" ]; then

echo "SCOPY: copy to s2 successful" >> $FTPLOG/$1

mv $FTPPUB/$i "$FTPPUB/${i}.done"

else

echo "SCOPY: copy FAILED" >> $FTPLOG/$1

mv $FTPPUB/$i "$FTPPUB/${i}.error"

fi

;;

(\*)

echo "PRIMARY: target INDETERMINABLE" >> $FTPLOG/$i

mv $FTPPUB/$i "$FTPPUB/${i}.error"

;;

esac

else

echo "PRIMARY: start and end NOT MATCHED" >> $FTPLOG/$i

mv $FTPPUB/$i "$FTPPUB/${i}.error"

fi

else

echo "SCHEMA: NOT MATCHED" >> $FTPLOG/$i

mv $FTPPUB/$i "$FTPPUB/${i}.error"

fi

t=$( date )

echo "FINISH: $i at $t" >> $FTPLOG/$i

done