FileXfer
File Transfer Jobs

Raymond E. Marcil
<rmarcil@gci.com>

Revision 0.0.1 (August 25, 2016)

Abstract

The FileXfer application is a system for automated file transfer jobs for copying files. "There are 3 applications that make up the usage collection framework: `filexfer`, which does the actual file transfers; `filexfer-jobmonitor`, which is configured to monitor various aspects of jobs and create NMS alarms when necessary; and `filexfer-dataloader`, which bulk-loads file data into database tables. There are also housekeeping scripts called `filexfer-filearchive`, which keeps files in the data directory pruned and compressed, and `filexfer-fileunarchive`, which allows files to be pulled out of the archive so filexfer jobs can work with them again."[1]

[1] Usage Collection Framework (filexfer)

Contents

List of Figures

List of Tables

List of Definitions and Abbreviations

- **MOA** - Municipality of Anchorage

1 Introduction

FileXfer is a custom application that GCI OSS built whose primary purpose is to transfer files from point A to point B (most of the time via point C/itself). It can also be used to load the collected data into a database if the data fits within the constraints of MySQL Load Data Infile SQL Syntax. It supports FTP and SFTP for both gets and puts, and it also has limited support for HTTP gets (this feature is used to collect weather camera images off of the Terra mountain top sites for the FAA).[2]

FileXfer can also be used to prune the source server's target files to a certain number of days. And while the default is to keep the source file time, this feature can be toggle off on a per job basis, resulting in the files having the transfer time instead as some customers prefer to know when the file was dropped off and not when the file was generated. For performance reasons there is a cutoff feature as well which defaults to 5 days for new jobs. FileXfer will not look more than the cutoff days back to see if a file should be collected and/or exported. FileXfer jobs are also capable of running in audit mode, in which FileXfer will log all of the transfers but it won't physically transfer anything. This feature can be useful to get a feed caught up without transfering a bunch of files around if for whatever reason the backlog of files doesn't need to be processed by any customers.

FileXfer logs all file transfers and any errors. However, it is not considered an error if there are no files to collect. FileXfer also supports monitoring of transfer jobs, and can generate an alert for any reason that you can articulate with SQL. Some examples include late and/or missing files, load queue too large, file too small, no files transfered for a certain interval, etc. The monitoring supports internal only alerts, TAC visibile alerts, and/or emailing the alerts. The email feature also supports sending of texts to cell phones.

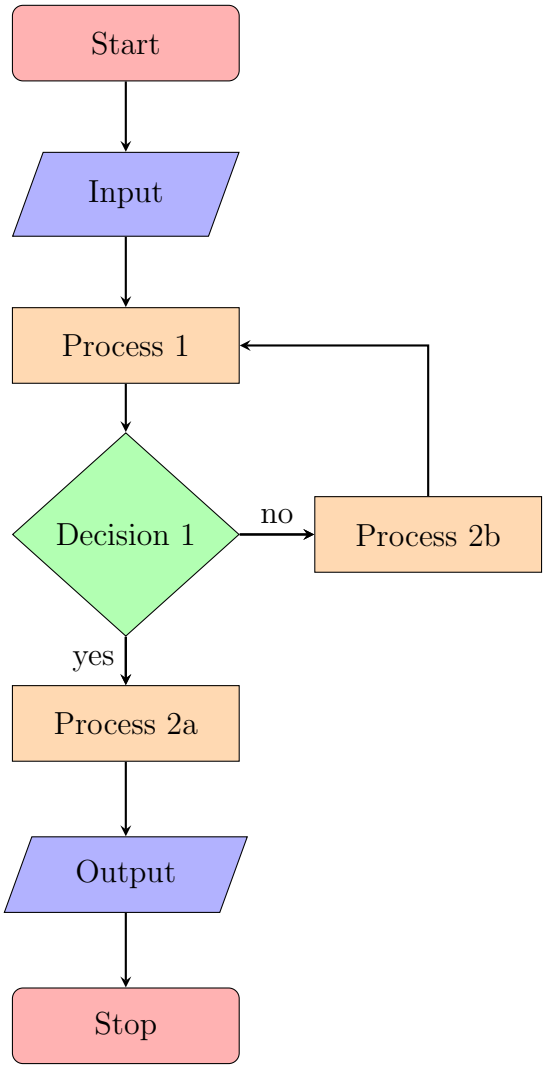
FileXfer will re-transfer a file if either the size and/or source file time changes, as that signals something about the file has changed. Some feeds leverage this concept as they may use a static filename in which the data is simply re-writen to same exact file at regular intervals.

The main FileXfer app server is `prod-prov4-cdr1.operations.gci.com` (`192.168.161.47`). The "ACS" / Project Seward FileXfer app server (which contains only ACS/Project Serward related jobs) is the SPS2 OSS Test app server, `osstest-em-provisioning.operations.gci.com` (`192.168.56.4`; public IP `66.223.155.33`).

[2] FileXfer.txt:3, GCI Network Services, OSS Mark Blum, Spring 2016

2 Design

[FIXME: Need data here...]



[FIXME: Customize flowchart FileXfer]

3 Schema

The main FileXfer database server is `sadc-cdr-mysql1.operations.gci.com` (`192.68.56.189`) and the ACS/Project Seward FileXfer database server is the SPS2 OSS Test database server, `osstest-db-provisioning.operations.gci.com` (`192.168.69.149`). For both database servers the FileXfer database is called `filexfer`.[3]

Here's a breakdown of the tables:

```
[mblum@development-mark ~]$ mysql -h sadc-cdr-mysql1 -sss \
-e "USE filexfer; SHOW TABLES"
DATABASECHANGELOG
DATABASECHANGELOGLOCK
errors
joblogs
jobs
loadjobs
loadqueue
logs
monitors

DATABASECHANGELOG
DATABASECHANGELOGLOCK
```

These two tables manage how updates to the database structure of FileXfer are preformed and stored.

3.1 errors

This table stores any errors FileXfer encounters.

3.2 joblogs

This is actually a view. Here's the create statement:

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `filexfer`@`%`
    SQL SECURITY DEFINER
VIEW `filexfer`.`joblogs` AS
    SELECT
        `j`.`idJob` AS `idJob`,
        `l`.`idLog` AS `idLog`,
```

[3] FileXfer.txt:11, GCI Network Services, OSS Mark Blum, Spring 2016

```
    `j`.`jobName` AS `jobName`,
    `j`.`neName` AS `neName`,
    `l`.`srcFileName` AS `srcFileName`,
    `l`.`srcFileSize` AS `srcFileSize`,
    `l`.`srcFileTime` AS `srcFileTime`,
    `l`.`destFileSize` AS `destFileSize`,
    `l`.`destFileName` AS `destFileName`,
    `l`.`dtTransferStart` AS `dtTransferStart`,
    `l`.`dtTransferStop` AS `dtTransferStop`
FROM
    (`filexfer`.`jobs` `j`
JOIN `filexfer`.`logs` `l` ON ((`j`.`idJob` = `l`.`idJob`)));
```

3.3 jobs

This stores all of the transfer jobs. Transfer jobs utilize a cron like schedule, and their run policy can be set to schedule (i.e. run as per the cron schedule), always, and never. There is also a wantSummary field that can be set to either yes or no. Some customers (mainly StarSolutions MSC Usage jobs) want summaries of the transfer file (number of records etc.) and this feature automatically generates that and transfer it along with the file. Priority is another feature of the FileXfer system. Given a limited amount of resources you can set differing priority levels for jobs (1 - 100 with 1 being the highest priority), so that jobs with higher priorities get preference over lower priority jobs when system resources are constrained. Generally production jobs get high priority (5) and lab/test jobs get low priority (100).

The `neName` field, which stands for Network Element Name, basically defines the directory where the files will be stored. They are stored in `/data/usage/[neName]`. Files are automatically archived if they are older than 2 days (to `/data/usage/[neName]/archive`), and archived files are deleted if they are older than 30 days. There is a script that can be used to unarchive files (say for retransfer purposes) called `/usr/bin/filexfer-fileunarchive`. It takes a file mask to unarchive and supports glob syntax.

The `idSite` field is an incomplete feature and not really used at this time.

3.4 `loadjobs`

This stores all of the load jobs. Not every transfer job has a corrisponding load job, and some transfer jobs may have more than one load job.

3.5 `loadqueue`

This is the queue for the load jobs. Entries in here have yet to be loaded. The loading is done in order.

Here's a SQL query that can be used to view the loadqueue:

```
# FileXfer Load Queue Status
SELECT idJob, idLoadJob, jobName, neName,
IF( idJob = 0,FROM_UNIXTIME( Count ), Count ) AS 'Count', fileName, fileTime
FROM
(
SELECT 0 AS 'idJob', 0 AS 'idLoadJob', 'Current Time' AS 'jobName',
        'neName', UNIX_TIMESTAMP( NOW() ) AS 'Count', 'fileName', 'fileTime'

UNION

SELECT idJob, idLoadJob, jobs.jobName, jobs.neName, COUNT(*),
        fileName, fileTime
FROM filexfer.loadqueue
JOIN filexfer.loadjobs USING ( idLoadJob )
JOIN filexfer.jobs USING ( idJob )
GROUP BY `idJob`
ORDER BY ( CASE WHEN `idJob` = 0 THEN 0 ELSE 1 END ) ASC, 5 DESC
) x;
```

3.6 logs

This is the table where all of the transfer logs are kept. In order to retransfer a specific file that has not changed you will have to delete the corrisponding log entry. Log entries are never pruned from this table.

3.7 monitors

This table encompasses all of the job monitors. Like load jobs, not every transfer job has a job monitor, and some transfer jobs may have more than one job monitor.

4 Implementation

[FIXME: Need data here...]

4.1 Dependencies

- cron
- inotify
- Perl v5.8.8
- Relevance

4.2 Configuration

`prod-prov4-cdr1`:

- `/etc/cron.d/filexfer`
- `/etc/cron.d/filexfer-userscripts`

4.3 Relevance

[FIXME: Need data here]

4.4 Scripts

The FileXfer application is made up of serveral scripts which are driven by `cron` as follows:[4]

```
[root@prod-prov4-cdr1 ~]# cat /etc/cron.d/filexfer
MAILTO=""
PERL5LIB=/opt

# This script preforms all of the get/collect jobs.
* * * * * filexfer /usr/bin/filexfer -c /etc/filexfer/filexfer-get.conf -t get
# This script preforms all of the put/export jobs.
* * * * * filexfer /usr/bin/filexfer -c /etc/filexfer/filexfer-put.conf -t put

# This script runs all of the job monitor jobs, and generates the relevant \
alerts and/or emails.
* * * * * filexfer /usr/bin/filexfer-jobmonitor -c /etc/filexfer/jobmonitor.conf

# This script runs all of the data loader jobs.
* * * * * filexfer /usr/bin/filexfer-dataloader -c /etc/filexfer/dataloader.conf
#* * * * * filexfer /usr/bin/filexfer-epg-dataloader.plx \
-c /etc/filexfer/epg-dataloader.conf

# These scripts take care of the archiving
2 0 * * * filexfer for dir in /data/usage/*; do /usr/bin/filexfer-filearchive \
$dir >>/var/log/filexfer/filearchive.log 2>&1; done
5 0 * * * filexfer find /data/usage -type f -name \*.sum -mmin +43200 | xargs rm
```

[4] FileXfer.txt:93, GCI Network Services, OSS Mark Blum, Spring 2016

Userscripts

FileXfer also has the concept of userscripts. These are scripts that are run after files are collected and before they are exported (in general).

The userscripts on `prod-prov4-cdr1`, are under `/usr/lib/filexfer/`:

```
$ ls -1
aubstats.sh
convert-wps-om-counters-report-part2.pl
convert-wps-om-counters-report.pl
dataloader-murderer.sh
encrypt-carrier-usage.sh
ericsson-oss-rl-reports-preprocess.sh
ericsson-oss-sts-reports-preprocess.plx
ExtractCarrierTurboZoneUsage
ExtractCarrierTurboZoneUsage_0.2
homisco-preprocess.sh
interop-mmsc-preprocess.sh
jm-clear-mailstat.sh
jm-get-mailstat.sh
loadWispData_config.xml
loadWispData.log
loadWispData.plx
processLegacyUsage2.plx
processLegacyUsage4.plx
processLegacyUsage5.plx
processLegacyUsage_config.xml
processLegacyUsage.log
processLegacyUsage.log.1.gz
processLegacyUsage.log.2.gz
processLegacyUsage.log.3.gz
processLegacyUsage.log.4.gz
processLegacyUsage.log.5.gz
processLegacyUsage.log.6.gz
processLegacyUsage.log.7.gz
processLegacyUsage.log.8.gz
rxnet01-get-latest
rxnet01-get-latest-PRIOR-20160830
stp-rop-logs-preprocess.sh
strip-spaces-from-csv.pl
teltronics-eos-preprocess.sh
terra-weather-camera-log-pruner.sh
zombiekiller.sh
$
```

[FIXME: Need description of individual userscripts]

Following is the crontab, `/etc/cron.d/filexfer-userscripts`, on `prod-prov4-cdr1` for scheduling the userscripts:

```
[root@prod-prov4-cdr1 ~]# cat /etc/cron.d/filexfer-userscripts
MAILTO=""
PERL5LIB="/opt"
PV_TEST_PERL=1

* * * * *               filexfer chmod 644 /data/usage/SDE01P/*
* * * * *               filexfer chmod 644 /data/usage/SDE01L/*
0-14 0 * * *            filexfer /usr/lib/filexfer/\
strip-spaces-from-csv.pl >>/var/log/filexfer/strip-spaces-from-csv.log 2>&1
0-14 1 * * *            filexfer /usr/lib/filexfer/\
convert-wps-om-counters-report.pl >>/var/log/filexfer/\
convert-wps-om-counters-report.log 2>&1
0-14 1 * * *            filexfer /usr/lib/filexfer/\
convert-wps-om-counters-report-part2.pl >>/var/log/filexfer/\
convert-wps-om-counters-report-part2.log 2>&1
*/15 * * * *            filexfer /usr/lib/filexfer/\
ericsson-oss-rl-reports-preprocess.sh -v /data/usage/OSS01/preprocess/rl*.out \
>>/var/log/filexfer/ericsson-oss-rl-reports-preprocess.log 2>&1
20 * * * *              filexfer /usr/lib/filexfer/\
ericsson-oss-sts-reports-preprocess.plx GCIMSC0 \
/data/usage/OSS01/preprocess/*GCIMSC0* >>/var/log/filexfer/\
ericsson-oss-sts-reports-preprocess.log 2>&1
20 * * * *              filexfer /usr/lib/filexfer/\
ericsson-oss-sts-reports-preprocess.plx ANAKB01 \
/data/usage/OSS01/preprocess/*ANAKB01* >>/var/log/filexfer/\
ericsson-oss-sts-reports-preprocess.log 2>&1
*/4 * * * *             filexfer /usr/lib/filexfer/\
stp-rop-logs-preprocess.sh -v /data/usage/{SADC5E,SDC5E}/\
preprocess/stp*.log >>/var/log/filexfer/stp-rop-logs-preprocess.log 2>&1
*/5 1 * * *             filexfer /usr/lib/filexfer/\
teltronics-eos-preprocess.sh -v /data/usage/EOS01/\
preprocess/BR*.IBR.primary >>/var/log/filexfer/teltronics-eos-preprocess.log 2>&1
* * * * *               filexfer /usr/lib/filexfer/\
teltronics-eos-preprocess.sh -v /data/usage/EOS01/\
preprocess/BR*.IBR.primary >>/var/log/filexfer/teltronics-eos-preprocess.log 2>&1
20-25 * * * *           filexfer /usr/lib/filexfer/\
interop-mmsc-preprocess.sh -v /data/usage/IOP01/preprocess/A*.dat \
>>/var/log/filexfer/interop-mmsc-preprocess.log 2>&1
*/5 * * * *             filexfer /usr/lib/filexfer/\
homisco-preprocess.sh -v /data/usage/HMSC01/preprocess/*.txt \
```

```
>>/var/log/filexfer/homisco-preprocess.log 2>&1
10,25,40,55 * * * *     filexfer /usr/lib/filexfer/\
ExtractCarrierTurboZoneUsage/ExtractCarrierTurboZoneUsage/\
ExtractCarrierTurboZoneUsage_run.sh --context_param fileMask=acs_15min_*.csv \
--context_param carrierId=ACS --context_param carrierPassphraseFilepath=\
/etc/filexfer/acs-gpg-passphrase.txt \
--context_param carrierFilenameFormat="'wifi-'yyyyMMddHHmm'.csv'" \
>>/var/log/filexfer/ExtractCarrierTurboZoneUsage_ACS.log 2>&1
00 06 * * *             filexfer /usr/lib/filexfer/processLegacyUsage4.plx
30 06 * * *             filexfer /usr/lib/filexfer/loadWispData.plx
20 02 * * *             filexfer /usr/lib/filexfer/aubstats.sh
```

4.5 Logging

Application Logging

The filexfer applications log to the `/var/log/filexfer` directory on `prod-prov4-cdr1.\ operations.gci.com`. The parent `filexfer` jobs log to `filexfer-get.log` and `filexfer-\ put.log`. The `jobmonitor` and `dataloader` applications log to `jobmonitor.log` and `dataloader.log`, The `filexfer` applications log to the `/var/log/filexfer` directory on `prod-prov4-cdr1.operations.gci.com`. The parent `filexfer` jobs log to `filexfer-get.\ log` and `filexfer-put.log`. The `jobmonitor` and `dataloader` applications log to `jobmonitor.log` and `dataloader.log`, respectively. Each file transfer job is executed as a child process and gets its own log file. The format is `filexfer-{neName}-{idJob}-{get,put}.log`.

By default, the jobs log at the warn level. Adjust the level to info to get a high-level view of the application's state. Adjust log verbosity by modifying the appropriate config file in `/etc/filexfer`. The changes will take effect after the next program execution.

Errors are also logged to a database table which can be browsed in the filexfer web interface under the '`Logs & Errors`' view. This view includes messages logged at `warn`, `error`, and `fatal` severity.[5]

File Transfer Logging

Every file transfer is recorded in a database table. There are two reasons for this table: first, it tells `filexfer` hich files have already been transferred, and second, it provides an audit trail for SOX compliance. The table is `filexfer.logs` on `sadc-cdr-mysql1.operations.gci.com`. Use the `filexfer.joblogs` view to easily find logs by job name or network element ID.

File transfer logs may also be viewed in the '`Logs & Errors`' page of the web interface.[6]

[5] Usage Collection Framework (filexfer)
[6] Usage Collection Framework (filexfer)

4.6 Test

[FIXME: Need data here...]

4.7 Issues

[FIXME: Need data here...]

5 Operation

There is a FileXfer GUI app on Relevance that can be used to create/delete/update transfer/load jobs and monitors, and view logs and errors. It is available on both presenter 4 on presenter 1 (but at this time it is safer to use presenter 4 to update jobs). Also the lab presenter (lab-presenter4) is currently pointed at the production ACS FileXfer instance.[7]

5.1 FileXfer

There are times it is desirable to run FileXfer outside of the FileXfer GUI app on Relevance. Perhaps when it is desirable to execute a single file transfer job. To do this login to GCI Network Services, OSS `prod-prov4-cdr1.operations.gci.com` (`192.168.161.47`).

The `filexfer` command is available from the command line. It has relative straightforward syntax:

```
[root@prod-prov4-cdr1 usage]# /usr/bin/filexfer
Usage:
    filexfer.plx -c configfile -t {get|put} [options]

[root@prod-prov4-cdr1 usage]# cd ~
[root@prod-prov4-cdr1 ~]# /usr/bin/filexfer
Usage:
    filexfer.plx -c configfile -t {get|put} [options]

[root@prod-prov4-cdr1 ~]# /usr/bin/filexfer --help
Usage:
    filexfer.plx -c configfile -t {get|put} [options]

Arguments:
    -c, --configfile
        Specify the configuration file to load. Must be in YAML format.

    -t, --transfertype
        One of "get" or "put". Get jobs download files and put jobs upload
        files.

Options:
    -d, --piddir
        Directory where the pid file will be written. Defaults to
        /var/run/filexfer.
```

[7] FileXfer.txt:149, GCI Network Services, OSS Mark Blum, Spring 2016

```
    --db
        Sets the database connection parameters. Valid keys are: server
        (default localhost), port (default 3306), driver (default mysql),
        uid, pwd, database, and table. Specify tags as key/value pairs,
        e.g.:

            --db server=localhost --db database=filexfer

    -e, --evengehost
        Address of the Evenge web server. Used to send indicators and events
        to the NMS system.

    --evengetimeout
        Timeout in seconds for communicating with the Evenge web server.
        Defaults to 10.

    -f, --cachefile
        Template cache file location. Defaults to
        /var/lib/filexfer/filexfer.kch.

    -h, --help
        Output this documentation.

    -m, --maxchildren
        Maximum number of child processes to spawn. Defaults to 50.

    -p, --pidfile
        PID file name. This will be appended with a ".pid" suffix.

    -r, --resource
        Resource name of this application. Used in indicator and event
        messages sent to the NMS system.

    --verbose, -v
        Log to the screen at increasingly verbose levels. This option may be
        repeated multiple times to increase the log level. For example, "-v"
        logs at info level, "-vv" logs at debug level, and "-vvv" logs at
        trace level.

[root@prod-prov4-cdr1 ~]#
```

The files transfered `filexfer` are under the `/data/usage/` directory.

```
$ ssh hnmadm@prod-prov4-cdr1
hnmadm@192.168.161.47's password:
```

```
Last login: Fri Aug 19 15:36:27 2016 from 10.103.193.217
[hnmadm@prod-prov4-cdr1 ~]$ cd /data/usage/
[hnmadm@prod-prov4-cdr1 usage]$ ls
AAA01       GA05  GA23  GA44  GA68  GC28  GC50    HAR01    P810     SPB02
ACS01       GA06  GA24  GA45  GA71  GC30  GC51    HCR01    P811     SPS02
AUB01       GA07  GA26  GA46  GA73  GC31  GC52    HMSC01   RBB01    SWPP01
BAL01       GA08  GA27  GA47  GA76  GC34  GC66    INC01    reports  SYNIVERSE
CAR01       GA09  GA28  GA48  GA77  GC35  GC87    INC02    RXNET01  TAL01
CBM01       GA10  GA29  GA49  GC05  GC36  GC88    INC03    SADC5E   TCS01
CDMA        GA11  GA30  GA50  GC15  GC37  GC89    IOP01    SBC01    TMP01
cdrReports  GA12  GA32  GA51  GC16  GC38  GC90    KUL01    SDC5E    TSUNAMI
CON01       GA13  GA33  GA52  GC17  GC39  GC91    legacy   SDE01    UNG01
CSKY01      GA14  GA34  GA53  GC18  GC41  GC93    LGCBX01  SDE01L   UNK01
CSKY02      GA15  GA35  GA55  GC19  GC42  GC94    MSW01    SDE01P   VORTEX01
DIM01       GA16  GA36  GA56  GC20  GC43  GC95    MTAS01   SDE02    WIFI01
EOS01       GA17  GA37  GA57  GC21  GC44  GC97    MUK01    SDE02L   WISP01
EPG01       GA18  GA38  GA58  GC22  GC45  GC98    Nov-18   SDE02P   WPS01
EPG02       GA19  GA39  GA59  GC23  GC46  GCT1    NRTRADE  SGSN01
GA01        GA20  GA41  GA65  GC24  GC47  GCT2    OCR01    SHU01
GA03        GA21  GA42  GA66  GC26  GC48  GGSN01  OSS01    SMSC01
GA04        GA22  GA43  GA67  GC27  GC49  GSM     P8       SPB01
[hnmadm@prod-prov4-cdr1 usage]$ sudo -s
Password:
[root@prod-prov4-cdr1 archive]#
```

The directory names under `/data/usage/` map to the `neName` column in the `filexfer.jobs` table that is displayed in the FileXfer GUI application under the File Transfer Jobs display in the `NE ID` field.

The `filexfer get` jobs currently executing can be inspected with:

```
[root@prod-prov4-cdr1 usage]# ps ax | grep get
17585 ?        S      0:00 /usr/bin/filexfer \
-c /etc/filexfer/filexfer-get.conf -t get P8
17586 ?        S      0:00 /usr/bin/filexfer \
-x /etc/filexfer/filexfer-get.conf -t get P811
17588 ?        SN     0:00 /usr/bin/filexfer \
-c /etc/filexfer/filexfer-get.conf -t get SMSC01
17593 ?        SN     0:00 /usr/bin/filexfer \
-c /etc/filexfer/filexfer-get.conf -t get KUL01
17594 ?        SN     0:00 /usr/bin/filexfer \
-c /etc/filexfer/filexfer-get.conf -t get CAR01
24286 ?        S      0:00 /usr/bin/filexfer \
-c /etc/filexfer/filexfer-get.conf -t get GCT2
```

```
[root@prod-prov4-cdr1 usage]#
```

The `filexfer put` jobs currently executing can be inspected with:

```
[root@prod-prov4-cdr1 usage]# ps ax | grep put
...
 3863 ?        S      0:00 /usr/bin/filexfer \
-c /etc/filexfer/filexfer-put.conf -t put GC93
 3864 ?        S      0:00 /usr/bin/filexfer \
-c /etc/filexfer/filexfer-put.conf -t put GC94
 3866 ?        S      0:00 /usr/bin/filexfer \
-c /etc/filexfer/filexfer-put.conf -t put GC37
 3867 ?        S      0:00 /usr/bin/filexfer \
-c /etc/filexfer/filexfer-put.conf -t put GC95
 3869 ?        S      0:00 /usr/bin/filexfer \
-c /etc/filexfer/filexfer-put.conf -t put GA32
 3870 ?        S      0:00 /usr/bin/filexfer \
...
[root@prod-prov4-cdr1 usage]#
```

The same filexfer syntax displayed in the process list can be using to execute an individual job to transfer files:

```
[root@prod-prov4-cdr1 ~]# /usr/bin/filexfer
Usage:
    filexfer.plx -c configfile -t {get|put} [options]

[root@prod-prov4-cdr1 ~]


[root@prod-prov4-cdr1 ~]# /usr/bin/filexfer -c /etc/filexfer/filexfer-put.conf -t put GC
```

5.2 Job Scheduling

Jobs are scheduled using a web interface at `nms.operations.gci.com/relevance`. Navigate to the "FileXfer" application and click the "File Transfer Jobs" link. Job execution happens on `prod-prov4-cdr1.operations.gci.com`. A `cron` job executes every minute from `/etc/cron.d/filexfer` to kick off the various `filexfer` scripts.[8]

New Job

The basic requirements for setting up a transfer new job are as follows:.[9]
POC: [for when problems occur]
IP: [source host for collect jobs/destination host for export jobs]
Protocol: [ftp/sftp]
Credentials: [username/password]
File Path:
File Mask: [supports glob syntax]
Schedule: [cron syntax]
Want Summary: [yes/no; generally only for StarSolutions CDR exports]

Job Timing

The parent `filexfer` script is responsible for spawning child processes for each job. Since a large number of jobs can be scheduled at any given interval, the parent process limits how many children can run concurrently. As long as the limit is reached and more jobs need to be spawned, the parent process must stay alive. Since this may take longer than 1 minute, it is possible for `filexfer` to miss certain scheduling intervals.

For example, if 500 jobs are scheduled to run at the top of every hour `(0 * * * *)` and the maximum child process limit is 50, there is a good chance `filexfer` will not execute any jobs scheduled to run at 1 minute past the hour `(1 * * * *)`. The best way to avoid this

[8] Usage Collection Framework (filexfer)
[9] FileXfer.txt:139, GCI Network Services, OSS Mark Blum, Spring 2016

is to use `0`, `15`, `30`, or `45` in the minute field of the job schedule. These intervals are always executed.[10]

5.3 Job Monitoring

The Relevance based FileXfer software has a Job Monitors section where Email notifications can be configured.

5.4 Dataloader

Dataloader jobs are configured using the web interface at `nms.operations.gci.com/relevance`. Navigate to the "`FileXfer`" application and click the "`Data Load Jobs`" link. These jobs are executed every minute as long as there are files in the load queue.[11]

5.5 Relevance

[FIXME: Need data here]

[10] Job Timing
[11] Usage Collection Framework (filexfer)

Appendix

Source

There are 3 primary FileXfer `perl` scripts on `prod-prov4-cdr1`:[12]

File name	Attributes	Description
`filexfer.plx`	181 lines	File transfer jobs
`filexfer-dataloader.plx`	132 lines	Data loader
`filexfer-jobmonitor.plx`	132 lines	Job Monitor

Table 1: FileXfer `perl` scripts on `prod-prov4-cdr1`

[12] `prod-prov4-cdr1.operations.gci.com` (`192.168.161.47`, NATed IP: `66.223.199.228`), data including CDRs and such under `/data/usage/` — Network Services, OSS.

filexfer.plx — File Transfer Jobs

```
1  #!/usr/bin/perl
2
3  use strict;
4  use warnings;
5
6  use Modules::App::FileXfer ();
7  our $VERSION = $Modules::App::FileXfer::VERSION;
8
9  # Core modules
10 use Clone qw( clone );
11 use File::Basename ();
12 use File::Spec ();
13 use POSIX ();
14
15 $SIG{CHLD} = \&Modules::App::FileXfer::REAPER;
16
17 MAIN: {
18     # Process and merge command-line and config file options
19     my $getopt    = Modules::App::FileXfer::
       ↪ get_command_line_options();
20     my $fileconf = Modules::App::FileXfer::read_config_file(
       ↪ $getopt->get_configfile );
21     Modules::App::FileXfer::merge_options( $getopt, $fileconf );
22
23     # Make sure we're the only instance running
24     Modules::App::FileXfer::check_pid_file( $Modules::App::
       ↪ FileXfer::Options->{pidfile} );
25
26     # Get logger and evenge objects
27     Modules::App::FileXfer::create_evenge_obj();
28     my $logger = Modules::App::FileXfer::create_logger_obj(
29         $Modules::App::FileXfer::Options->{logger}, $Modules::App
           ↪ ::FileXfer::Program );
30
31     # Get the ready jobs
32     my $fx        = Modules::App::FileXfer::create_filexfer_obj(
       ↪ $Modules::App::FileXfer::Program );
33     my $jobs      = Modules::App::FileXfer::get_jobs( $fx );
34     my $loadjobs = Modules::App::FileXfer::get_jobs_with_load_jobs
       ↪ ( $fx );
35     undef $fx;
36
```

```
37      for my $job ( @{ $jobs } )
38      {
39          # Enforce the "max children" constraint
40          $logger->info( 'Max child processes reached. Waiting for
                ↪ one to complete before starting job.' )
41              if ( scalar keys %Modules::App::FileXfer::Children
42                   >= $Modules::App::FileXfer::Options->{maxchildren
                         ↪ } );
43
44          sleep 1 while ( scalar keys %Modules::App::FileXfer::
                ↪ Children
45                          >= $Modules::App::FileXfer::Options->{
                              ↪ maxchildren} );
46
47          # Fork a child process for this job
48          $logger->info( sprintf( 'Spawning child process for job "%
                ↪ s".', $job->jobName ));
49
50          my $pid = fork;
51          defined $pid or Modules::App::FileXfer::log_event(
52              5, sprintf( "Can't fork for job \"%s\": %s", $job->
                    ↪ jobName, $! ), 'logdie' );
53
54          if ( $pid == 0 ) # child
55          {
56              # Set random seed for this child
57              srand();
58
59              # Lower the OS scheduling priority based on job
                    ↪ priority
60              POSIX::nice( Modules::App::FileXfer::pri_to_nice( $job
                    ↪ ->priority ));
61
62              # Add the NE name to our command line string
63              $0 .= " @ARGV " . $job->neName;
64              my $jobtag = Modules::App::FileXfer::get_jobtag( $job
                    ↪ );
65
66              # Set the subresource for this job in the evenge
                    ↪ object
67              $Modules::App::FileXfer::Evenge->subresourceName( $job
                    ↪ ->jobName );
68
69              # Create a logger specific to this child process
```

```
70              my $logopt = clone( $Modules::App::FileXfer::Options
                    ↪ ->{logger} );
71              my ( undef, $logdir ) = File::Basename::fileparse(
                    ↪ $logopt->{file}{filename} );
72              $logopt->{file}{filename} = File::Spec->catfile(
                    ↪ $logdir, "$jobtag.log" );
73
74              $logger->delete();
75              my $logger = Modules::App::FileXfer::create_logger_obj
                    ↪ ( $logopt, $job->jobName );
76              Log::Log4perl::MDC->put( 'idJob', $job->idJob );
77
78              # Make sure another instance isn't still running
79              Modules::App::FileXfer::check_pid_file( $jobtag );
80
81              # Create a FileXfer object for database updates
82              my $fx = Modules::App::FileXfer::create_filexfer_obj(
                    ↪ $jobtag );
83
84              # Execute the job
85              Modules::App::FileXfer::run_job( $fx, $job, $loadjobs
                    ↪ );
86
87              $logger->info( 'Child exiting.' );
88              exit 0;
89          }
90          else # parent
91          {
92              $logger->debug( sprintf( 'Spawned child process %d for
                    ↪  job "%s".', $pid, $job->jobName ));
93              $Modules::App::FileXfer::Children{ $pid } = $job->
                    ↪ jobName;
94          }
95      }
96
97      $logger->info( 'Main application exiting.' );
98  }
99
100 # Safely exit
101 $SIG{CHLD} = 'IGNORE';
102
103 __END__
104
105 =head1 NAME
```

```
106
107 filexfer -- Move a file from point A to point B over an IP network
108
109 =head1 VERSION
110
111 0.51
112
113 =head1 SYNOPSIS
114
115 filexfer.plx -c configfile -t {get|put} [options]
116
117 =head1 ARGUMENTS
118
119 =over 4
120
121 =item -c, --configfile
122
123 Specify the configuration file to load. Must be in YAML format.
124
125 =item -t, --transfertype
126
127 One of "get" or "put". Get jobs download files and put jobs upload
    ↪ files.
128
129 =back
130
131 =head1 OPTIONS
132
133 =over 4
134
135 =item -d, --piddir
136
137 Directory where the pid file will be written. Defaults to /var/run
    ↪ /filexfer.
138
139 =item --db
140
141 Sets the database connection parameters. Valid keys are: server (
    ↪ default
142 localhost), port (default 3306), driver (default mysql), uid, pwd,
    ↪ database,
143 and table. Specify tags as key/value pairs, e.g.:
144
145     --db server=localhost --db database=filexfer
```

```
146
147 =item -e, --evengehost
148
149 Address of the Evenge web server. Used to send indicators and
      ↪ events to the NMS system.
150
151 =item --evengetimeout
152
153 Timeout in seconds for communicating with the Evenge web server.
      ↪ Defaults to 10.
154
155 =item -f, --cachefile
156
157 Template cache file location. Defaults to /var/lib/filexfer/
      ↪ filexfer.kch.
158
159 =item -h, --help
160
161 Output this documentation.
162
163 =item -m, --maxchildren
164
165 Maximum number of child processes to spawn. Defaults to 50.
166
167 =item -p, --pidfile
168
169 PID file name. This will be appended with a ".pid" suffix.
170
171 =item -r, --resource
172
173 Resource name of this application. Used in indicator and event
      ↪ messages sent to the NMS system.
174
175 =item --verbose, -v
176
177 Log to the screen at increasingly verbose levels. This option may
      ↪ be repeated
178 multiple times to increase the log level. For example, "-v" logs
      ↪ at info level,
179 "-vv" logs at debug level, and "-vvv" logs at trace level.
180
181 =back
```

filexfer-jobmonitor.plx — Job monitor

```
1  #!/usr/bin/perl
2
3  use strict;
4  use warnings;
5
6  use Modules::App::FileXfer::JobMonitor ();
7  our $VERSION = $Modules::App::FileXfer::JobMonitor::VERSION;
8
9  MAIN: {
10     # Process and merge command-line and config file options
11     my $getopt    = Modules::App::FileXfer::JobMonitor::
          ↪ get_command_line_options();
12     my $fileconf = Modules::App::FileXfer::JobMonitor::
          ↪ read_config_file( $getopt->get_configfile );
13     Modules::App::FileXfer::JobMonitor::merge_options( $getopt,
          ↪ $fileconf );
14
15     # Make sure we're the only instance running
16     Modules::App::FileXfer::JobMonitor::check_pid_file(
17          $Modules::App::FileXfer::JobMonitor::Options->{pidfile} );
18
19     # Get logger and evenge objects
20     Modules::App::FileXfer::JobMonitor::create_evenge_obj();
21     my $log = Modules::App::FileXfer::JobMonitor::
          ↪ create_logger_obj(
22          $Modules::App::FileXfer::JobMonitor::Options->{logger},
23          $Modules::App::FileXfer::JobMonitor::Program
24     );
25
26     # Get a list of job monitors
27     my $jm = Modules::App::FileXfer::JobMonitor::
          ↪ create_jobmonitor_obj();
28     my $mons = Modules::App::FileXfer::JobMonitor::get_monitors(
          ↪ $jm );
29
30     for my $mon ( @{ $mons } )
31     {
32          Log::Log4perl::MDC->put( 'idJob', $mon->idJob );
33          $log->info( sprintf( 'Executing monitor "%s".', $mon->
               ↪ monitorName ));
34
```

```
35          # Set the subresource for this monitor in the evenge
               ↪ object
36          $Modules::App::FileXfer::JobMonitor::Evenge->
               ↪ subresourceName( $mon->monitorName );
37
38          # Execute the job monitor
39          eval { Modules::App::FileXfer::JobMonitor::run_monitor(
               ↪ $jm, $mon ) };
40          $@ and $log->error( "$@" );
41      }
42
43      $log->info( 'Main application exiting.' );
44  }
45
46  __END__
47
48  =head1 NAME
49
50  jobmonitor -- Monitor filexfer jobs for any condition and generate
       ↪  alerts
51
52  =head1 VERSION
53
54  0.51
55
56  =head1 SYNOPSIS
57
58  jobmonitor.plx -c configfile [options]
59
60  =head1 ARGUMENTS
61
62  =over 4
63
64  =item -c, --configfile
65
66  Specify the configuration file to load. Must be in YAML format.
67
68  =back
69
70  =head1 OPTIONS
71
72  =over 4
73
74  =item -a, --mailhost
```

```
75
76 Address of the mail server. Used to send email notifications.
      ↪ Defaults to localhost.
77
78 =item -d, --piddir
79
80 Directory where the pid file will be written. Defaults to /var/run
      ↪ /filexfer.
81
82 =item --db
83
84 Sets the database connection parameters. Valid keys are: server (
      ↪ default
85 localhost), port (default 3306), driver (default mysql), uid, pwd,
      ↪  database,
86 and table. Specify tags as key/value pairs, e.g.:
87
88     --db server=localhost --db database=filexfer
89
90 =item -e, --evengehost
91
92 Address of the Evenge web server. Used to send indicators and
      ↪ events to the NMS system.
93
94 =item -f, --cachefile
95
96 Template cache file location. Defaults to /var/lib/filexfer/
      ↪ jobmonitor.kch.
97
98 =item -h, --help
99
100 Output this documentation.
101
102 =item -i, --mailinterval
103
104 Minimum time, in seconds, before repeat emails may be sent for the
      ↪  same monitor. Defaults to 3600 (1 hour).
105
106 =item -l, --mailfrom
107
108 Sender's email address in email notifications. Defaults to [
      ↪ username]@[hostname].
109
110 =item -m, --mailstatfile
```

```
111
112 Mail status file location. Defaults to /var/lib/filexfer/
    ↪ jobmonitor-mailstat.kch.
113
114 =item -o, --mailport
115
116 Port on which the mail server is listening for SMTP traffic.
    ↪ Defaults to 25.
117
118 =item -p, --pidfile
119
120 PID file name. This will be appended with a ".pid" suffix.
121
122 =item -r, --resource
123
124 Resource name of this application. Used in indicator and event
    ↪ messages sent to the NMS system.
125
126 =item --verbose, -v
127
128 Log to the screen at increasingly verbose levels. This option may
    ↪ be repeated
129 multiple times to increase the log level. For example, "-v" logs
    ↪ at info level,
130 "-vv" logs at debug level, and "-vvv" logs at trace level.
131
132 =back
```

filexfer-dataloader.plx — Data Loader

```
1  #!/usr/bin/perl
2
3  use strict;
4  use warnings;
5
6  use Modules::App::FileXfer::DataLoader ();
7  our $VERSION = $Modules::App::FileXfer::DataLoader::VERSION;
8
9  MAIN: {
10     # Process and merge command-line and config file options
11     my $getopt   = Modules::App::FileXfer::DataLoader::
          ↪ get_command_line_options();
12     my $fileconf = Modules::App::FileXfer::DataLoader::
          ↪ read_config_file( $getopt->get_configfile );
13     Modules::App::FileXfer::DataLoader::merge_options( $getopt,
          ↪ $fileconf );
14
15     # Make sure we're the only instance running
16     Modules::App::FileXfer::DataLoader::check_pid_file(
17         $Modules::App::FileXfer::DataLoader::Options->{pidfile} );
18
19     # Get logger and evenge objects
20     Modules::App::FileXfer::DataLoader::create_evenge_obj();
21     my $logger = Modules::App::FileXfer::DataLoader::
          ↪ create_logger_obj(
22         $Modules::App::FileXfer::DataLoader::Options->{logger},
23         $Modules::App::FileXfer::DataLoader::Program
24     );
25
26     # Get the load jobs with pending files
27     my $dl   = Modules::App::FileXfer::DataLoader::
          ↪ create_dataloader_obj();
28     my $jobs = Modules::App::FileXfer::DataLoader::get_load_jobs(
          ↪ $dl );
29
30     for my $job ( @{ $jobs } )
31     {
32         next if 414 == $job->idJob;
33         Log::Log4perl::MDC->put( 'idJob', $job->idJob );
34         $logger->info( sprintf( 'Executing job "%s".', $job->
              ↪ jobName ));
35
```

```
36          eval {
37              # Get the list of pending load files
38              my $files = Modules::App::FileXfer::DataLoader::
                    ↪ list_load_files( $dl, $job );
39              next unless scalar @{ $files };
40
41              # Import the class for this job's files
42              Modules::App::FileXfer::DataLoader::import_file_class(
                    ↪  $files->[0]->{fileclass} );
43
44              # Bulk load the data from each file
45              for my $file ( @{ $files } )
46              {
47                  Modules::App::FileXfer::DataLoader::load_file_data
                        ↪ ( $dl, $job, $file );
48                  Modules::App::FileXfer::DataLoader::
                        ↪ dequeue_load_file( $dl, $job, $file );
49              }
50          };
51
52          $@ and $logger->error( "$@" );
53
54          # Close the external db handle
55          $dl->close_ext_dbh();
56      }
57
58      $logger->info( 'Main application exiting.' );
59      exit;
60  }
61
62  __END__
63
64  =head1 NAME
65
66  filexfer-dataloader -- Bulk load file data from filexfer into a
        ↪ database table.
67
68  =head1 VERSION
69
70  0.51
71
72  =head1 SYNOPSIS
73
74  filexfer-dataloader -c configfile [options]
```

```
75
76 =head1 ARGUMENTS
77
78 =over 4
79
80 =item -c, --configfile
81
82 Specify the configuration file to load. Must be in YAML format.
83
84 =back
85
86 =head1 OPTIONS
87
88 =over 4
89
90 =item -d, --piddir
91
92 Directory where the pid file will be written. Defaults to /var/run
   ↪ /filexfer.
93
94 =item --db
95
96 Sets the database connection parameters. Valid keys are: server (
   ↪ default
97 localhost), port (default 3306), driver (default mysql), uid, pwd,
   ↪  database,
98 and table. Specify tags as key/value pairs, e.g.:
99
100     --db server=localhost --db database=filexfer
101
102 =item -e, --evengehost
103
104 Address of the Evenge web server. Used to send indicators and
   ↪ events to the NMS system.
105
106 =item --evengetimeout
107
108 Timeout in seconds for communicating with the Evenge web server.
   ↪ Defaults to 10.
109
110 =item -f, --cachefile
111
112 Template cache file location. Defaults to /var/lib/filexfer/
   ↪ dataloader.kch.
```

```
113
114 =item -h, --help
115
116 Output this documentation.
117
118 =item -p, --pidfile
119
120 PID file name. This will be appended with a ".pid" suffix.
121
122 =item -r, --resource
123
124 Resource name of this application. Used in indicator and event
    ↪ messages sent to the NMS system.
125
126 =item --verbose, -v
127
128 Log to the screen at increasingly verbose levels. This option may
    ↪ be repeated
129 multiple times to increase the log level. For example, "-v" logs
    ↪ at info level,
130 "-vv" logs at debug level, and "-vvv" logs at trace level.
131
132 =back
```

[FIXME: Need data here]

Table 2 – FileXfer directories and files on `prod-prov4-cdr1`

Directory	File(s)
/etc/filexfer/	*.conf
/usr/bin/	filexfer-dataloader
	filexfer-dataloader.plx
	filexfer-dataloader.plx.mbak
	filexfer-epg-dataloader.plx
	filexfer-filearchive
	filexfer-filearchive.sh
	filexfer-fileunarchive
	filexfer-fileunarchive.sh
	filexfer-jobmonitor
	filexfer-jobmonitor.plx
	filexfer.plx
/usr/lib/filexfer/	*.gz, *.sh, *.plx
	ExtractCarrierTurboZoneUsage*
/usr/share/filexfer/	filexfer.changelog-*.xml

Continued on next page

Table 2 – continued from previous page

Directory	File(s)
	filexfer.changelog-master.xml
	liquibase.sh
	.gnupg/pubring.gpg
	.gnupg/random_seed
/var/cache/yum/build/packages/	filexfer-0.52-1.el5.centos.noarch.rpm
/var/lib/filexfer/	dataloader.kch
	dataloader_temp.kch
	filexfer-aaa01-13-get.kch
	...
	filexfer-wps01-706-get.kch
	filexfer.kch
	jobmonitor-mailstat.kch
	jobmonitor.kch
/var/log/filexfer	*.log
	ExtractCarrierTurboZoneUsage_ACS.log
	archive
	convert-wps-om-counters-report-part2.log
	convert-wps-om-counters-report.log
	dataloader.log
	dataloader_temp.log
	dataloadinsert.log
	datarecovery.log
	epg-dataloader.log
	ericsson-oss-rl-reports-preprocess.log
	ericsson-oss-sts-reports-preprocess.log
	filearchive.log
	filexfer-aaa01-13-get.log
	filexfer-aaa01-14-put.log
	...
	/var/log/filexfer/filexfer-wps01-706-get.log

LaTeX - Examples and Formatting

Comments

COMMENTS Comment — *Sean Weems, Spring 2003*
We should get the COMMENTS column searchable via the landrecords application before we do much anything else – shouldn't be too hard.

Errata: Plats spanning multiple sections
A few anomalies can be observed in the AKPLATS table. Specifically plats exist that span multiple sections. Since the table only has a single column, SCODE, that accepts a single section code, SGU (Status Graphics Unit) has handled this problem by entering multiple rows in the table, each with a different section that point to the same plat or file. Multiple section plats are indicated by setting the TCODE column to the value 37, and making an appropriate notation like *Section 24-25-26-27* in the REMARKS column.
[FIXME: Perhaps the SCODE column should accept an array of sections?]

Links

A Guide to LaTeX
http://www.astro.rug.nl/ kuijken/latex.html

LaTeX - From Wikibooks,the open-content textbooks collection
http://en.wikibooks.org/wiki/LaTeX

LaTeX Notes
http://luke.breuer.com/time/item/LaTeX_Notes/180.aspx