Setting up usage statistics

To monitor the use of your FileSender installation, other than by examining the various logs in the Administrator view of the Filesender user interface, you can install additional software to gather various kinds of statistics.

Webserver log analysers

Log analysers like AWStats and Webalizer can give you a good general overview of the use of your FileSender webserver, with reports about unique visitors (IP based), type of clients, where they came from and more. Note that these analysers don't give you exact details about the number of files, uploads, downloads and vouchers over time.

To monitor the more specific FileSender resources, SCRE and AARNet are collecting usage statistics with the Zabbix and MRTG monitoring tools. The following notes and files can get you started.

Zabbix

filesender_zabbix.pl - A Zabbix sensor as contributed by Emir Imamagic (SCRE)

Note from Emir:

```
I attached a sensor for Zabbix that we use to get statistics at Srce.

Gathered parameters are:
"filesender.users_total"
"filesender.users_active"
"filesender.files_active"
"filesender.files_total"
"filesender.vouchers_active"
"filesender.vouchers_total"
"filesender.upload_num"
"filesender.upload_size"
"filesender.download_num"
"filesender.download_size"
I think they are pretty self-explanatory, but I can add more text if
needed :)

By default configuration should be placed in /etc/filesender/config:
HOST=FS_DB_HOST
DB=FS_DB_NAME
USER=FS_DB_USER
PASS=FS_DB_OASS
ZABBIX_HOST=FS_ZABBIX_HOST
ZABBIX_SERVER=CENTAL_ZABBIX_SERVER

Sensor can be executed as Zabbix check or simply from cron. All errors
are printed to syslog.
```

MRTG

filesender_grabber.pl - A MRTG collector based on the SCRE Zabbix sensor by David Jericho (AARNet).

Note from David:

```
First I use MRTG to collect the stats:

WorkDir: /srv/www/cloudstor.aarnet.edu.au/mrtg/www
Logdir:  /srv/www/cloudstor.aarnet.edu.au/mrtg/logs

Options[_]: nopercent,noinfo,growright,noo,gauge
MaxBytes[_]: 10000000000000000000
ShortLegend[_]: &nbsp;
LegendO[_]:
LogFormat: rrdtool

Target[cloudstor.download_size]: `/srv/www/cloudstor.aarnet.edu.au/mrtg/bin/filesender_grabber.pl download_size`
Title[cloudstor.download_size]: Cloudstor Bytes Downloaded
PageTop[cloudstor.download_size]: <h1>Cloudstor Bytes Downloaded</h1>
YLegend[cloudstor.download_size]: bytes
LegendI[cloudstor.download_size]: &nbsp;Downloaded bytes:

And then finally, I fire off a bunch of rrdtool graphing commands, similar to:

rrdtool graph cloudstor.download_size-year.png -a PNG -A --title="Total downloaded bytes" \
  --vertical-label "bytes" \
  'DEF:probe1=cloudstor.download_size.rrd:ds0:AVERAGE' \
  'AREA:probe1#00eb0c:bytes' \
  'GPRINT:probe1:LAST:Total downloaded count\: %2.0lf bytes' \
  -s '-1 years'
```

Munin

Using the SQL from the zabbix-plugin. It uses a limited database-user "filesender_readonly" which has 'GRANT SELECT' on everything and nothing else, then it is possible to filter out the lookups on the database (every fve minutes) in the postgresql munin-plugins.

Setup:

```
[filesender*]
user postgres
env.PGPORT 5432
env.PGPASSWORD somepassword
env.PGUSER filesender_readonly
```

This shows numbers of files/vouchers/users that are yet to expire.

```
#!/bin/sh
# -*- sh -*-

if [ "$1" = "autoconf" ]; then
        echo yes
        exit 0
fi

if [ "$1" = "config" ]; then

        echo 'graph_title Filesender active'
        echo 'graph_category filesender'
        echo 'users_active.label users'
        echo 'users_active.type GAUGE'
        echo 'files_active.label files'
        echo 'files_active.type GAUGE'
        echo 'vouchers_active.label vouchers'
        echo 'vouchers_active.type GAUGE'
        exit 0
fi
```

```
PSQL='psql -h localhost -U filesender_readonly -At -d filesender -c '

echo 'users_active.value' `$PSQL "select count(distinct fileauthuserid) from files where filestatus = 'Available';"`
echo 'files_active.value' `$PSQL "select count(distinct fileuid) from files where filestatus = 'Available';"`
echo 'vouchers_active.value' `$PSQL "select count(*) from files where filestatus = 'Voucher';"`
```

This shows totals. Always growing, therefore split into its own plugin.

```
#!/bin/sh
# -*- sh -*-

if [ "$1" = "autoconf" ]; then
        echo yes
        exit 0
fi

if [ "$1" = "config" ]; then

        echo 'graph_title Filesender totals'
        echo 'graph_args -l 0 --base 1000'
        echo 'graph_category filesender'
        echo 'users_total.label users'
        echo 'users_total.type GAUGE'
        echo 'files_total.label files'
        echo 'files_total.type GAUGE'
        echo 'vouchers_total.label vouchers'
        echo 'vouchers_total.type GAUGE'
        echo 'upload_num.label uploads'
        echo 'upload_num.type GAUGE'
        echo 'download_num.label downloads'
        echo 'download_num.type GAUGE'
        exit 0
fi

PSQL='psql -At -d filesender -U filesender_readonly -h localhost -c '

echo 'users_total.value' `$PSQL "select count(distinct fileauthuserid) from files;"`
echo 'files_total.value' `$PSQL "select count(distinct fileuid) from files where filestatus = 'Available' or (filestatus = 'Closed' and fileexpirydate <> '1970-01-01 01:00:00');"`
echo 'vouchers_total.value' `$PSQL "select count(distinct fileuid) from files where filestatus LIKE 'Voucher%' or (filestatus='Closed' and fileexpirydate = '1970-01-01 01:00:00');"`
echo 'upload_num.value' `$PSQL "select count(*) from logs where logtype='Uploaded';"`
echo 'download_num.value' `$PSQL "select count(*) from logs where logtype='Download';"`
```

This last one shows uploads and downloads.

```
#!/bin/sh
# -*- sh -*-

if [ "$1" = "autoconf" ]; then
        echo yes
        exit 0
fi

if [ "$1" = "config" ]; then

        echo 'graph_title Filesender total up/downloads'
        echo 'graph_args -l 0 --base 1000'
        echo 'graph_category filesender'
        echo 'download_size.type GAUGE'
        echo 'download_size.graph no'
        echo 'upload_size.label bytes'
        echo 'upload_size.type GAUGE'
        echo 'upload_size.negative download_size'
        exit 0
fi

PSQL='psql -At -d filesender -U filesender_readonly -h localhost -c '

echo 'upload_size.value' `$PSQL "select coalesce(sum(logfilesize),0) from logs where logtype='Uploaded';"`
echo 'download_size.value' `$PSQL "select coalesce(sum(logfilesize),0) from logs where logtype='Download';"`
```

SQL-trick for getting statistics per domain

Make a view just for log-crunching:

```
CREATE VIEW aggregated_logs AS
  SELECT
    logs.logid,
    logs.logtype,
    split_part(lower(logs.logfrom::text), '@'::text, 1) AS logfrom_user,
    split_part(lower(logs.logfrom::text), '@'::text, 2) AS logfrom_domain,
    logs.logdate,
    logs.logauthuserid,
    logs.logfilesize
  FROM logs;
```

You now have the domain in "logfrom_domain" and can GROUP BY it. Preferrably each unique "logfrom_user" should be replaced with a token so as to anonymize the data, this has not been done here. "logauthuserid" is used to check for vouchered users.