Onepixd Manual

For daemon release 1.1

Contents

1. Overview

The name, onepixd, stands for “One PIXel gif server Daemon.” It is a persistent daemon that runs in the background on Unix based machines. The onepixd was designed with the following high-level requirements in mind:

- The daemon must continue to run despite network or system failures.
- The daemon must record minimally all errors encountered.
- The daemon must be secure and must not allow intrusion.
- The daemon must contain hooks for debugging and diagnosing problems.

With these high level requirements in mind, the onepixd was written utilizing the following software requirements:

- The daemon must be written in C for best performance.
- The daemon must be multi-threaded to avoid forks under Unix.
- The daemon must accept connections using standard TCP/IP.
- The daemon must support the standard HTTP protocol.
- The daemon must be completely configurable with a configuration files.

The onepixd is built using autoconf so that it may be easily ported to new and unforeseen operating systems and version of operating systems.

The onepixd program is entirely configuration file driven. One way to run it by hand might look like the following:

```
./onepixd -c onepixd.conf
```

This will launch the onepixd program as a constantly running daemon until killed. Because the onepixd program is configuration file centric, we describe that file first and follow that by the other aspects of the program.

2. The onepixd.conf file

The *onepixd* daemon employs a single configuration file. The file is typically called, *onepixd.conf*.

The *onepixd.conf* file provides most of the information needed for the *onepixd* daemon to run and function properly on your machine and inside your environment.

Empty lines in the configuration file and lines that begin with a # character are ignored. Each line may only specify a single configuration item. There may be

arbitrary white space surrounding the name, the =, or the value. Quotation marks have no special meaning.

The validity of a configuration file can be checked by running the daemon with a -C command-line switch (See “The -C switch” on page 10). If the configuration file is good, FILE IS OKAY TO USE is printed and the program exits with a zero value:

```
./onepixd -C onepixd.conf
onepixd.conf: FILE IS OKAY TO USE
```

If there is any error, the error is printed and the program exits with a non-zero return value:

```
./onepixd -C reader.conf
Config item: “send_404_on_error”: Bad value: “bob”
```

Be certain to re-run this check until all errors are corrected. The program will refuse to run so long as there is any error in its configuration file.

Finally note that unknown configuration items are silently ignored.

2.1. disk_archive_days=

The **disk_archive_days=** specifies the number of days that data will be archived. The data viewing interface can restrict reports to windows of time, so there is little risk to archiving data for a longer interval.

```
disk_archive_days = 6
```

The **disk_archive_days=** is mandatory and must be listed in the configuration file. If the value specified is non-numeric or negative, it will result in the following error:

```
./onepixd -C onepixd.conf
Config item: “disk_archive_days”: Bad value: “-3”
```

There is no maximum for the number of days. This setting is entirely up to your sense of disk and inode space. This item may only be specified once.

2.2. disk_archive_path=

The **disk_archive_path=** specifies the directory in which data will be saved. If this item is omitted, it becomes the default directory “*data*” in the home directory of the *onepixd* daemon. If this item is specified, it must be a directory that already exists, the *onepixd* daemon will not create the directory. Note that

the *onepixd* daemon requires user level read and write permission to that directory.

```
./onepixd -C onepixd.conf
Config item: “disk_archive_path”: Value: “/usr/local/data”: No such file or
directory

./onepixd -C onepixd.conf
Config item: “disk_archive_path”: Value: “/net/local/server”: Not a directory

./onepixd -C onepixd.conf
Config item: “disk_archive_path”: Value: “/etc”: Not owner
```

This configuration item may only be specified once.

2.3. send_404_on_error=

The *onepixd* daemon will careful screen each query to insure that the base directory of the query matches that specified by the **gif_httpbase=** configuration item, and that each equate following the “?” in the query matches the order and names of the columns specified with the **gif_column=** configuration item. If they are not all perfect, an error condition is raised. Because only errors are logged, this condition will always be logged.

If you want to return a **404** error for all failed matches, any email messages that contains that link will see a broken image. Once your system is setup and running, that effect should be of little risk. But for development and initial testing, you might want to return a one pixel GIF image no matter what.

What is returned on error is specified by this **send_404_on_error=** configuration item. Set it to **TRUE** to return **404** on errors, set it to **FALSE** to return a one pixel **GIF** images on an error:

```
send_404_on_error = true    ← On error, return a 404 reply
send_404_on_error = false   ← On error return a one pixel GIF image
```

Note that if the letter which begins the true/false express is Y, y, T, t, S, s, or 1, the value will evaluate to TRUE, whereas if the letter which begins the true/false express is N, n, F, f, or 0, the value will evaluate to FALSE. If it is none of these characters you will see the following error:

```
./onepixd -C onepixd.conf
Config item: “send_404_on_error”: Bad value: “bob”
```

2.4. become_user=

The **become_user=** configuration item specifies under what user identity this daemon should run. For example, if it is the is the user *bob*, it will change its identity and run as *bob*. This is important because the daemon should never run as *root*.

```
become_user = 1234    ← Specify a user-id number
become_user = bob     ← Specify a usr login name
```

Because this daemon writes to files based on input from the outside world, it is vital that it not run as root. If the user specified is not known the following error will result:

```
Config item: “become_user”=“foo” You may specified a bad user.
```

We suggest you create a unique user for this daemon. Perhaps call it onepixd the same as the daemon. On Linux, you could create such a daemon with commands like this:

```
/usr/sbin/useradd -d /usr/local/onepixd -n onepixd
```

This will create both the user and group called onepixd, and set the home directory to where you plan to run the daemon. The user you select must exist before you can run the daemon. This configuration item can only be specified once.

2.5. home_dir=

The *onepixd* daemon can configurably be installed in most any directory. Locations selected in the past include:

```
/usr/local/onepixd
/opt/shar/onepixd
/packages/onepixd
```

This **home_dir=** line specifies the directory where the *onepixd* daemon will find all its other needed files and directories. Because the *onepixd* daemon performs a *chdir*() to that location upon startup, this is also the location where core files, if any, will be written.

This **home_dir=** item is mandatory and must be present and properly defined. It must be an already existing directory that is owned by, and writable by the user defined by the **become_user=** configuration item (See “become_user=” on page 6).

This **home_dir=** directory may only be defined once.

2.6. pidfile=

If this item is specified, the *onepixd* daemon will write its process ID value into the file specified.

```
pid_file_name   = onepixd.pid
```

If a relative name is used as above it will be written into the directory specified by the **home_dir=** item (See “home_dir=” on page 6). But if it is a full path name, that full path is used instead:

```
pid_file_name = /vav/run/bmsrdaemon.pid
```

Beware, however, that this file is written after the run time user is set. The directory into which this file is written must be owned by the user set with the **become_user=** configuration item (See “become_user=” on page 6).

This **pidfile=** item is optional and may be omitted without harm. But note that on some Unix systems, this pid file is needed to stop the daemon upon system shutdown.

This **pidfile=** may only be specified once in the configuration file.

2.7. numthreads=

The *onepixd* daemon uses a thread pool model. You specify the number of worker threads to launch at program startup. Those threads will launch and block and wait for inbound connections to handle.

```
numthreads     = 500
```

There is no minimum, just a maximum of 9996 threads. Because three administrative threads are always running, this effectively restricts you to 9999 threads.

The number of threads you select depends on your available memory (RAM). With too little memory and too many threads, some thread paging may occur. You should also find the default limit on the number of file descriptors available to root. Adjust the number of threads to no more than one third that maximum number of available file descriptors.

2.8. log_facility=

The *onepixd* daemon always logs errors using *syslog*. The default *syslog* facility used is **LOG_MAIL**. This configuration item may be used to specify a different facility:

```
log_facility = local3
```

If specified, the value given for this **log_facility=** item is any of the *syslog.h* facilities listed on your system, with the "LOG_" prefix removed. The above declaration is the equivalent to the **LOG_LOCAL3** syslog facility.

2.9. _interface=

The *onepixd* daemon has three interfaces:

- The Gif Interface is used download one pixel gifs and to record the query.
- The Data Interface is used to fetch data reports from the daemon.
- The Upload Interface is used to upload data to be join to the gif data for data reports.

Each interface uses a configuration setting to specify the interface to which to bind:

- gif_interface=
- data_interface=
- upload_interface=

All three must be specified once and no more than once in the configuration file. Each takes as its value an IPaddress. There are three forms that the address can take. Below we show for the GIF interface, for example, the three forms:

```
gif_interface = 192.168.0.55
gif_interface = 0.0.0.0
#gif_interface =
```

Here, the first expression causes the GIF interface to *bind*() to the address 192.168.0.55. The second expression above causes the GIF interface to *bind*() to all network devices, virtual and real. And the last expression, where the interface is undefined, causes the onepixd daemon to *bind*() to INADDR_ANY, which is all real network devices, but excludes virtual devices.

Note that you may not specify a host nor a domain name.

2.10. _port=

The *onepixd* daemon has three interfaces:

- The Gif Interface is used download one pixel gifs and to record the query.
- The Data Interface is used to fetch data reports from the daemon.
- The Upload Interface is used to upload data to be joined to the gif data for data reports.

Each interface uses a configuration setting to specify the port to which to bind:

- gif_port=
- data_port=
- upload_port=

All three must be specified once and no more than once in the configuration file. Each takes as its value an integer expression. For example:

```
gif_port = 8080
data_port = 8081
upload_port = 8182
```

Remember that the *onepixd* daemon never runs as *root*. As a consequence, no service may *bind*() to a privileged interface. On some systems that is any port below 1028, while on other it is any port below 4096. If you want the GIF interface to answer queries on port 80, you will need to set up virtual host/port mapping in your router.

Not that you may not specify a symbolic port from /etc/services.

2.11. _timeout=

The *onepixd* daemon has three interfaces:

- The Gif Interface is used download one pixel gifs and to record the query.
- The Data Interface is used to fetch data reports from the daemon.
- The Upload Interface is used to upload data to be joined to the gif data for data reports.

Each interface uses a configuration setting to specify the connection timeout for that interface:

- gif_timeout=
- data_timeout=
- upload_timeout=

All three may be specified once and no more than once in the configuration file. Each takes as its value an integer expression. For example:

```
gif_timeout = 5
data_timeout = 20
upload_timeout = 10
```

If a connection is made to an interface, and if not data or request is received within the timeout interval, the connection will be dropped by the onepixd daemon. For queries that should be lighting fast, such as the Gif Interface, the timeout may be small. For manually composed queries, such as on the Data Interface, the timeout may be large. Timeouts prevent denial of service attacks that can tie up a server with idle connections.

2.12. Other Configuration Items

Because the remaining configuration items are specific to Interface Services, we continue the discussion of those configuration items in those services.

3. Command-Line Arguments

The *onepixd* daemon is a program that is started using the command-line. That command line can either be executed by hand or can be executed by a startup script to run unattended. If run with no arguments the following is printed:

```
Usage: onepixd [-c /path/onepixd.conf or -C /path/onepixd.conf] -d what [-f]
```

Note that only the -c or -C switch is mandatory. All the others are optional.

3.1. The -c switch

The -c switch tells the *onepixd* daemon the location of its configuration file. The daemon reads and parses its configuration and tries to start running.

3.2. The -C switch

The -C switch tells the *onepixd* daemon the location of its configuration file. The daemon reads and parses its configuration and reports any errors found. If no errors are found it reports FILE IS OKAY TO USE.

3.3. The -d switch

The **-d** command-line switch turns on debugging. This switch takes a single argument which tells it what to debug. Currently only a single argument is supported:

```
./onepixd -f -d all -c onepixd.conf
```

Here, **-d all** turns on all debugging. Note that the **-f** switch must be present or debugging is turned off. That is, if **-d** is specified, it must always be prefixed by a **-f**.

Note that a finer division of debugging will be offered in a future release.

3.4. The -f switch

Unless told to do otherwise, the *onepixd* daemon will always run as a Unix daemon, in the background and detached from controlling terminals and groups. To prevent this behavior, perhaps for testing or perhaps to wrap in a baby-sitting script, you may use this **-f** command-line switch. The **-f** command-line switch causes the daemon to remain in the foreground, attached to the controlling terminal.

Omitting the **-f** command-line switch not only puts the daemon in the background, it also disconnects from its standard input, standard output and error output. This has a side effect of disabling debugging output when run in the background.

4. The Gif Interface

The Gif Interface is the one connected to by other sites on the Internet to request a one-pixel GIF image file:

```
<img src="http://your.site.com/secure_path/?column_name=column_value....
```

Two configuration items define how this interface is to be used:

- **gif_file=** is the optional name of a GIF file whose contents to return
- gif_path_columns= is the secure/path used to validate each query followed by the name part (the column_name) for each column.

We first describe the configuration items then show how it all fits together.

4.1. gif_file=

There are two different one-pixel GIF images available for the onepixd daemon to return. One is built in and is quick and easy to return. To use the built in one-pixel GIF image, simply omit defining this gif_file= configuration item:

```
# gif_file = nothing        ← Use the built-in image.
```

By omitting the gif_file = definition, you cause the daemon to return its built-in GIF image.

There may be times, however, when you already embed a tiny GIG into an HTML formatted email, you may wish to use that GIF in its present role, rather than use the built-in one. To use your own GIF simply specify its path/filename to this gif_file = configuration item:

```
gif_file = /usr/local/images/web/email/hspace.gif
```

If the file specified exists and is readable by the daemon, it will be loaded at startup and used in place of the built-in GIF image. If the file is missing, or is not readable, the daemon will log an error and will refuse to start.

4.2. gif_path_columns=

An HTTP base is the *path/filename* that follows the hostname in an HTTP query:

```
http://your.domain.com/directory/      ← A directory
http://your.domain.com/file            ← A file
```

You use this **gif_path_columns=** configuration item to specify the text that will fit between the slashes following the hostname. One name is specified per **gif_path_columns=** item, is the text to the left of the colon character:

```
gif_path_columns = dog:.....
```

The values listed here may only contain numbers, letters, and the underscore character. They are case insensitive so if you specify dog and the html query specifies DOG the two will still match.

The data to be saved for later data reports comes from the part of the HTML query following the question-mark.

```
http://www.yourdomain.com/path/?       ← data follows the ? character
```

What follows is a sequence of one or more name=value pairs each separated from the next with an ampersand character:

```
?store=macys&item=sweater
```

The **gif_path_columns**= configuration item is used to list the required names that must appear following the ? in the HTML query. The **gif_path_columns**= lists one or more column names following the colon:

```
gif_path_columns = dog:opened_id,opened_key
```

The above would require all HTML queries to list the names following the ? in the left-to-right order:

```
?opened_id=value&opened_key=value
```

If a query contains different names, or too few, or too many, or the correct names but in the wrong order, the query is rejected. (See “send_404_on_error=” on page 5)

The values listed here may only contain numbers, letters, and the underscore character. They are case insensitive so if you specify open_id and the html query specifies OPEN_ID the two will still match.

4.3. GIF Interface Summary

The GIG Interface uses six configuration items:

If the above were configured properly, the following query might be made, as for example:

```
<img src=”http://your.com/dog?opened_id=bob@example.com&opened_key=1234”
width=”0” height=”0” border=”0”>
```

If this query arrived at noon on August 23, 2010, the data would be placed under the data directory specified by the **disk_archive_path=** configuration item.

Essentially all data is placed into buckets of one-day duration. The name of the data file is constructed from the **gif_path_columns=** configuration items and the order they were specified, as for example:

```
dog,opened_id,opened_key
```

The HTML query’s data is appended to this file as a single line of CSV style data:

```
12345678901234,bob@example.com,1234
```


The first column in the file is a Unix time() expression of seconds.

The only data accepted from the outside world is the values that are appended as a single line to the appropriate data file. No data item may be longer than 128 characters and any non-ASCII data will cause the value to be ignored.

5. The Upload data Interface

The Upload Interface is the one connected to by your internal machines to upload additional data. Perhaps a record of every email sent to compare to the open rate from the GIF interface. The upload interface uses html GET statements the same as the GET interface, but prefixed by upload_ instead of gif_:

```
upload_path_columns = email_sent:sent_id,sent_tz
```

5.1. The upload_pubkey=

In general, the upload interface will be protected by a firewall. In the event that it faces the open Internet, a means is provided to sign each upload using a private/publid key pair. To see how this works look at the examples under the tools directory.

```
upload_pubkey = MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBALIzCrSt5kWAV6KogF4l4XYFis6
pBOl1GVJyFvMBv+apyHq6n2D1VKTZNtQ9UGva4ZGjRXWTAVWHOrgAmamCMicCAwEAAQ==
```

You can use the makekey.sh script to create a public/private key pair. The file key.pub.pem's contents can be appended to the configuration file as shown above.

6. The Data Interface

The data report interface is entirely defined by four configuration items:

To use the data interface, just connect to the correct port on the server machine:

```
http://www.yourserver.com:8100/
```

6.1. The data_passphrase=

At a well run site, the data interface will be protected by a firewall. In the event that the data interface faces the Internet, a login facility has been provided. Please note that this login facility is not nearly as strong as a firewall would be.

To set up a login account, just list a user name and password separated by a colon following the data_passphrase=, as for example:

```
data_passphrase         = bob:November
data_passphrase         = fred:Tango Dancer
```

The user login name and the password are case sensitive. That is, if fred configured, only fred can be used to log in, FRED will fail. There may be as many accounts as you wish.

7. How to Install

The distribution is for download. For this description we presume you already downloaded the distribution:

```
onepixd-1.1.tar.gz
```

Unpack like this:

```
% tar xzvf onepixd-1.1.tar.gz            ← Linux
% gzcat onepixd-1.1.tar.gz | tar xvf -   ← Solaris
```

Change into the distribution directory

```
onepixd-1.1
```

There you run the following two commands to build the daemon:

```
./configure
make -s
```

To install the daemon run:

```
make install
```

Be certain to set up a script in /etc/init.d to start the daemon when your machine reboots. Because such scripts vary so widely between flavors of Linux and between Solaris and OS-X, we leave the creation of such a script to you.

You may use the run.sh script in the source directory as one way to insure that the daemon runs even if it faults and fails.

8. Known Bugs

A few problems are currently not detected, so we treat these areas as potential bugs.

8.1. Missing Gif file

If you specify your own gif file using the gif_file= configuration item, and if that file vanishes while the daemon is running, the missing file will not be detected until the daemon next starts. At that point, the failure will prevent the daemon from restarting.

8.2. Download gif failures invisible

If your gif image is downloaded with a setting of:

```
wigth="1" height="1" border="0"
```

A failed gif download will remain invisible. For this reason, you may want to set up your mail generation code to insert a different reference for email sent to your internal QA team. Perhaps:

```
widgth="14" height="14" border="3"
```

With this setting, a broken image will show up for your QA folk and they can immediately notify production.