Installation and Setup of

# GigA+ (from version 0.1-8.50 and higher)

## Install the package, get a license

Download the package suitable for your system from the website, then untar it and execute the installation script as root. For instance, on a Debian-derived Linux distro:

```
~/tmp$ tar -xzvf ./gigaplus_0.1.9-0-gigaplus.x86_64.deb.tar.gz
gigaplus.deb
Install-deb.sh

~/tmp$ sudo ./install-deb.sh ./gigaplus.deb
Preparing to unpack ./gigaplus.deb ...
Unpacking gigaplus (0.1-9.0) ...
...
~/tmp$
```

Your copy is not licensed yet, so if you run gxng, you'll see the following message:

```
$ gxng -V
gxng (Relay engine) 0.1-9.0 (gigaplus) regular [Ubuntu 16.04.3 Linux
(amd64) - 4.4.0-131-generic/x86_64] (udefined) 65c1091d/0; built on
2018-07-25
==
= ERROR: License not found, please contact support@gigapxy.com ==
==
License validation failed, application will quit.
```

A license is issued based on a key, generated by GigA+. The key ties it up to the server the application would run on. Get the key as:

```
$ gigaplus key
System key:  [4083b942140de80b8f2bbac0ee7be479e59a11c7dc8f429d] (0x410a)
```

NB: **`sudo gigaplus key`** under *FreeBSD*

Copy the full output (including the value in round brackets) into an email to request a demo license from support@gigapxy.com. Your license will arrive in a tarball containing `gxa.lic` - the license file. Copy `gxa.lic` to `/etc` directory and run `gxng` module to check that the license works.

```
$ gxng -V
gxng (Relay engine) 0.1-9.0 (gigaplus) regular [Ubuntu 16.04.3 Linux
(amd64) - 4.4.0-131-generic/x86_64] (2018-12-31) 65c1091d/2253082; built on
2018-07-25
```

NB: **`sudo -u gigaplus gxng -V`** under *FreeBSD*

You're now licensed till January 31st, 2018 `(2018-12-31)` - but **please** make sure to **renew in advance**.

**NB:** If license check fails: make sure UDP `port 123 (NTP)` **is open** for outgoing requests.

## Set up the environment

GigA+ requires a system-scope initialization file - `gigaplus.env`. The file's location depends on the OS: `/etc` for Linux, `/usr/local/etc` - for FreeBSD. Environment setup is invoked via the `gigaplus setup, gigaplus` being the application's control script.

The set up presents a series of dialogs and the menu to review selected choices. Before you launch the setup, make sure `mail/mailx` is installed and functions on your box, test-send a couple of messages to your **sysadmin email** to check. Prepare for the following questions:

1. How many GigA+ engines (gxng's) will run on this box?
2. Will you be using gxng or NginX for HLS-segment delivery?
3. Will you be using HLS + HTTP, HTTP-only or HLS-only streaming?
4. Do you want to **pin** each gxng to a distinct CPU core?
5. Do you want to use a special non-system partition for core dumps? (GigA+ enables those.)

When ready, run from the console:

```
$ sudo gigaplus setup
```

The setup results get saved at `/etc/gigaplus.env` (Linux) or `/usr/local/etc/gigaplus.env`.

## Create configuration (gigaplus.conf)

Master configuration file, `gigaplus.conf`, is to be created using a template at `/opt/gxa/etc/gigaplus-default.conf`. Use any text editor you like.

The reason to edit `gigaplus.conf` manually is simple: you'll need make corrections there a few times yet. Before we proceed to editing, I suggest reading (at least) the GigA+ core manual:

```
$ man gigaplus
```

Let's copy the template and edit it:

```
$ cp /opt/gxa/etc/gigaplus-default.conf /tmp/gigaplus.conf
$ vim /tmp/gigaplus.conf
```

There'll be a few sections to address, related to different modules.

If you're beginning to wonder about the meaning of all the parameters with no adjacent comment), you may read a respective man page or study a commented config file:

```
$ man gxws
$ less /usr/share/doc/gigaplus/examples/gigaplus.conf
```

NB: Use `/usr/local/share/doc/`… on FreeBSD.

### CONFIG: Request listeners (ws.listener)

Make sure you agree with the default access ports and the **network interfaces** the requests would be available from. Mind that **admin** requests might be sensitive to allow on all interfaces. Use your judgement.

`Example:`

```
   listener: {
        admin = { ifc = "lo"; port = "4047"; default_af = "inet";  };
        user  = { ifc = "eth0"; port = "4046"; default_af = "inet"; };
    };
```

In the above example we allow admin requests only on the local interface (same box) and let user requests come from eth0, ports not changed.

### CONFIG: Multicast interface (ws.multicast_ifc)

This is also set to “**all**” by default, but should be a distinct network interface instead; otherwise the OS will pick one for you and will assume that all multicast data should originate there.

Example:

```
multicast_ifc  = "eth1";
```

### CONFIG: HLS settings (ws.hls.*)

The most important setting here is `ws.hls.enabled`. If you **don’t** plan to use HLS, just set it to `false` and move on to the `ng.* (gxng)` section of the config.

```
$ man gxng
```

### CONFIG: Buffer subsystem (ng.bufd.*)

Buffer subsystem is responsible for caching HTTP-stream data. If you are **not** planning to use linear, non-HLS streams, then you don’t need it. Turn it off, it consumes a lot of memory (it needs it).

Example:

```
bufd.enabled = false;
```

If you do need HTTP streams, then you’ll need to do a few experiments with the settings in this section until you find your perfect combination of start mode, segment number and size.

### CONFIG: Playlist manager (gpm.*)

If you only use HTTP streaming and do **not** use **HLS**, then you’re done - go to the end of this chapter, save and copy the config.

Or read on the next module:

```
$ man gxpm
```

### CONFIG: Item-URL prefix (gpm.item_url_prefix)

Initially set to an intentionally bogus value:

```
item_url_prefix = "http://acme.tv:4046/hls-fra/";
```

You need to make sure the prefix makes sense for your box and your access port. Let’s say, the host, where you install, has an external address **187.12.22.5**, then (assuming you’ve not changed the default **gxws user-request port 4046** to something else), the initial part is **http://187.12.22.5:4046**.

The **hls-fra** segment means one would be using **gxws**/**gxng** to transfer HLS segments to clients. If that's the plan, then the edited setting would become:

```
item_url_prefix = "http://187.12.22.5:4046/hls-fra/";
```

However, if your chose **NginX** (or any other 3rd party tool) for uploading HLS segments, you need to provide the prefix that would route to the segment data. Let's assume that you've configured access to the data via **port 8088** and **tvdata** segment.

Then your prefix becomes:

```
item_url_prefix = "http://187.12.22.5:8088/tvdata/";
```

In the end, **gxpm** will use the prefix to form URL's within HLS playlists. For a **cinemax** (channel) segment, such a URL might be:

Example:

```
http://187.12.22.5:8088/tvdata/cinemax/2018-08-01/104933021.ts
```

#### CONFIG: client playlists per channel (gpm.max_src_tasks)

This number specifies how many distinct playlists can there be per channel. There is only **one LIVE** playlist per channel but as **many DVR** playlists as there are clients (negative time-shift DVR playlists are the exception, there it's one playlist per shift). If using DVR, adjust this number (not to exceed **512**).

#### Finalizing

For the simplest scenario (no load balancing or replication), this is all that needs to be done. We save `gigaplus.conf` and copy it over to `/etc` or `/usr/local/etc` (on FreeBSD).

```
$ cp gigaplus.conf /etc
```

## Add a channel (HLS)

Each stream the will be served via HLS requires a specification file. The master script (`gigaplus`) handles creation of channel specs via a series of N-curses dialogs. However, before the script is called to create a channel, let's gather some information (needed for each of the channels to be created).

1. **URL** of the source stream (in ffmpeg-compatible format). It makes sense to check (from the installation host, mind you!) if the URL yields any data. Tools: `ffprobe, wget, ncl` (GigaTools);

2. How many minutes/hours worth of data are you going to store or the server? If you're not doing DVR, leave it to the default 10 minutes, otherwise it's your call. If you're handling large amounts of data, should you be using shards?
3. Are you going to spread HLS-segment requests across multiple hosts (i.e. replicate segments)?

Once you have the answers, we proceed:

```
$ sudo gigaplus create
```

At the end we should have the spec file ready and the channel recognized by the master script:

```
$ ls -l /opt/gxa/etc/spec/
total 4
-rw-r--r-- 1 gigaplus adm 2833 Aug  2 10:09 boleslavska.spec
$ gigaplus status
[gxpm] is not running.
[gxws] is not running.
----- channels -----
(SUSP) boleslavska [1 MB]  (ALERT!)
```

Note that the channel has been immediately *suspended* so that it does not get started automatically. When ready, you need to *enable* it.

```
$ sudo gigaplus enable boleslavska
ENABLED channel[boleslavska]
$
```

Now it's time to start the show.

```
$ sudo gigaplus start
Starting gxpm ...
boleslavska [10743] running; uptime: 00d 00h:00m:00s
----- channels -----
(ON) boleslavska [1 MB] uptime: 00d 00h:00m:00s
--------------------
gxpm [10695] running; uptime: 00d 00h:00m:01s | U[10720]
[gxws] is not running.
Starting gxws ...
Starting (regular) gxng1 ...
Starting (regular) gxng2 ...
Pausing for 1 second(s).
gxws [10894] STARTED; uptime: 00d 00h:00m:03s | U[10919]
gxng [10952] STARTED; uptime: 00d 00h:00m:02s | U[10978]
gxng [11009] STARTED; uptime: 00d 00h:00m:01s | U[11037]
gxpm [10695] STARTED; uptime: 00d 00h:00m:04s | U[10720]
$
```

The channel is ready to be watched. You can see which modules are running, their PIDs in square brackets. An `upkeep` process is watching every module, shown here as `U[NN]`, where NN = upkeep's own PID.

Now we can check the status:

```
$ gigaplus status
gxpm [10695] running; uptime: 00d 00h:10m:14s | U[10720]
gxws [10894] running; uptime: 00d 00h:10m:14s | U[10919]
gxng [10952] running; uptime: 00d 00h:10m:13s | U[10978]
gxng [11009] running; uptime: 00d 00h:10m:12s | U[11037]
----- channels -----
(ON) boleslavska [565 MB] uptime: 00d 00h:10m:14s
--------------------
```

Let's watch the channel. From a client we run:

```
$ cvlc -vvv http://192.168.1.173:4046/hls-m3u/boleslavska/playlist.m3u8
```

Should be watching it now.

So, the first channels in online. Run `sudo gigaplus create`, add another channel. Repeat until you've got them all. Congratulations!

## Troubleshooting

### Check the logs

The answer to the question "*What went wrong?*" is often in the logs. Logs for the core modules can be found in `/opt/gxa/log` directory. Logs for the channel-bound modules/scripts are in the channel directory, for instance, `/opt/gxa/channel/boleslavska`. The first log to look at is `vsm.log`. `vsm` is the shell script responsible for ingesting the channel stream.

In `vsm.log` you may find references to other logs that are also in the channel directory. If something is wrong with the stream (bad URL, no connection, no data. etc.), one of `gxseg` logs is most likely to hold a clue.

### Cannot connect to gxws(1)

If you simply cannot connect to the primary port (usually `4046`), corresponding to `gxws(1)`, then the first thing to check is whether `gxws(1)` receives your requests. See if `/opt/gxa/log/gxws.log` holds the clue to that. If no requests come through, it might be the firewall (allow `4046/tcp`).

**Probe the stream**

Another thing to consider, when things don't work out with a channel is: *how is the stream doing*? See if it runs at all, probe it with `ffprobe` to see if its format is ok. Don't probe from your desktop, do it from the server where the issue is, please.

**Seek outside help**

If all of the above fails, write to support (support@gigapxy.com). There is also a Google+ community where all sorts of information could be found. A Telegram-based support channel is there for interactive help.

Enjoy GigA+!