



# Packet Ship *Timeline* IPTV Recorder

**ICG-TL**

Installation & Configuration Guide  
for 1.2 “Hooke” release

Document version 1.2.0-DRAFT1

Documents software versions:  
ps-captured 1.2.1



# Contents

<b>Introduction.....</b>	<b>3</b>
Overall architecture.....	3
<b>Installation .....</b>	<b>5</b>
Prerequisites.....	5
Installing on Debian.....	5
Installing on CentOS / Red Hat.....	6
Manual installation.....	7
Trying it out.....	8
<b>Systems Administration.....</b>	<b>10</b>
Binaries.....	10
init.d script.....	10
Configuration.....	10
Logging.....	10
Processes & Threads.....	11
<b>General server configuration.....</b>	<b>12</b>
Logging.....	12
Background daemon.....	13
Watchdog restart.....	13
User/group identity.....	13
Licence file.....	13
<b>Multicast capture configuration.....</b>	<b>14</b>
Persistent capture state.....	14
Capture configuration.....	14
Shared capture parameters.....	15
Specific capture parameters.....	16
<b>EPG capture configuration.....</b>	<b>18</b>
Disk storage.....	18
Huffman-encoded tables.....	18
EPG data streams.....	20
<b>Messaging configuration.....</b>	<b>21</b>
HTTP Server.....	21
XMLMesh Connection.....	21
Message bindings.....	21



## Introduction

Packet Ship Timeline is a Linux server component which provides a server-side disk recorder for IPTV streams and EPG data, and a fast EPG navigation engine. This document describes how to install and configure Packet Ship Timeline on a Linux server. You should be familiar with Linux systems administration, video-on-demand concepts and XML in order to use this guide.

---

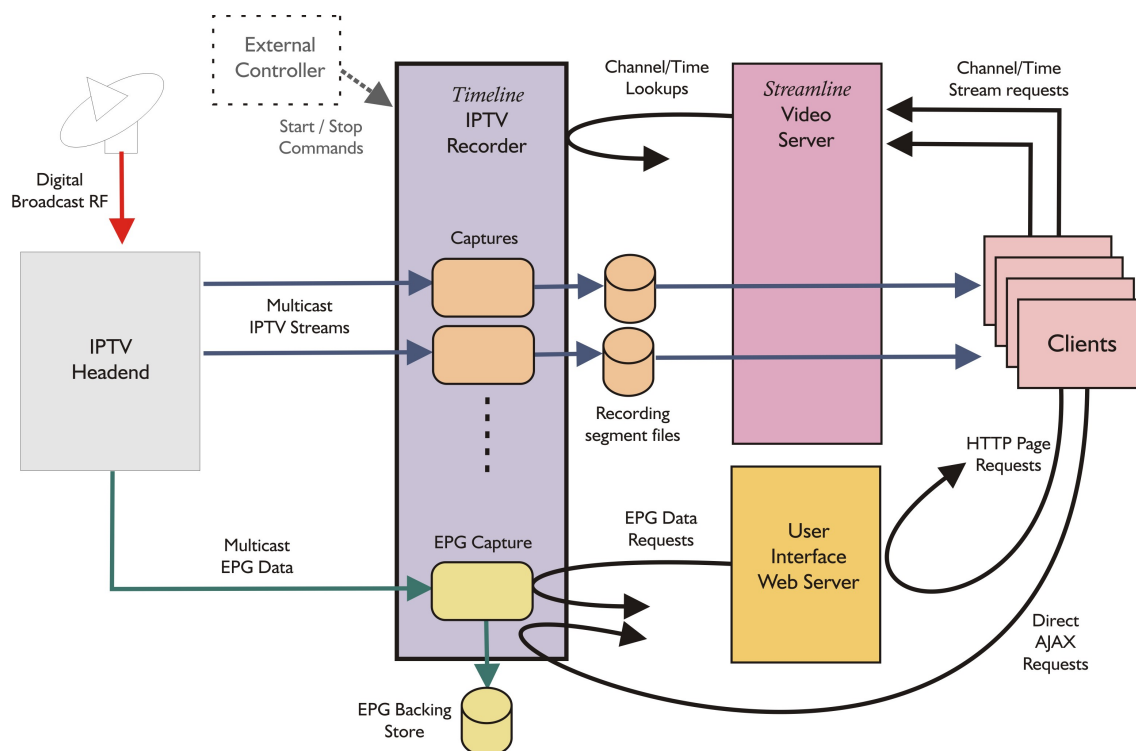
## Overall architecture

The Packet Ship Timeline software is a single Linux daemon, 'ps-captured' (Pea-Ess-Capture-Dee).

The overall function of the daemon is as follows:

- Capturing of IPTV streams (MPEG-2 Transport Streams) from multicast outputs from an IPTV head-end, and writing them to disk.
- Maintaining a circular buffer of recorded content for each stream, stored as a group of disk files each holding a fixed-length segment of time.
- Recording specific programmes to external disk files on request
- Real-time indexing of the streams for variable bit-rate streaming and trick mode fast-forward/rewind in the Streamline video server.
- Providing an interface for the Streamline video server to obtain a playlist for a given channel and time in the past.
- Capturing both retrospective and forward EPG data from one or more separate multicast streams containing MPEG-2 Event Information Table (EIT) data, and holding them in memory with disk backup.
- Providing an interface for a Web server or Javascript application to obtain EPG data to create EPG user interfaces.

The following diagram shows how the Timeline IPTV recorder fits into an integrated IPTV and EPG solution:



## Integration with the Streamline video server

The default configuration as supplied assumes the Timeline IPTV Recorder will be run as an independent system as a pure recorder. However it will most usually be used integrated with the Streamline video server as shown above. To do this requires some configuration changes in both daemons – please see Application Note AN-TL-1102 for details.



# Installation

Packet Ship Timeline uses Debian and CentOS Linux as its reference platforms, and by default the software is supplied as a Debian package (.deb) and Red Hat package (.rpm), for both 32-bit and 64-bit Intel/AMD architectures. One of these will work on most Linux distributions, but it is also possible to install it by hand – see later section.

## Unix-friendly

Packet Ship Timeline is designed for integration with existing Linux platforms, so `ps-captured` runs as a standard Unix daemon, just as you would expect from standard services like 'bind' and 'apache'. Like most Unix software it is configured in text files kept in `/etc`: specifically `/etc/packetship/captured.cfg.xml`. Most of this guide concerns the contents of this file, but you'll find the comments fairly self-explanatory if you don't like reading manuals!

Also like conventional Unix daemons, `ps-captured` has an `'init.d'` script and sends its logging to `/var/log/` – so, managing a Timeline installation is pretty much like managing any other Linux server.

---

## Prerequisites

Packet Ship Timeline requires Linux with a 2.6 kernel. The standard release is built for Debian 6.0 ("squeeze") and CentOS 5.

## Licensing

The Timeline server defaults to a "Demonstration Edition" mode which can record only a single IPTV stream for up to 24 hours. **This is licensed for evaluation, development and demonstration purposes only, not for use in a production system.**

For use in production and for higher capacity the server requires a cryptographically signed licence file to operate, which sets the licensed capabilities and capacities of the server. Licence files are locked to a particular server's MAC address, and need to be generated by Packet Ship specifically for each hardware installation. Once you start production there is an online service where you can order and download licences yourselves.

If you have purchased a full licence you will receive a licence XML file, and this should be copied to `/etc/packetship/licence.xml` (although the location can be changed in configuration if required). If you copy the licence file after installation, the daemon will need to be restarted to obtain the extra capacity.

---

## Installing on Debian

The Debian package (.deb) provided will install on either Debian 6.0 ('squeeze') or 5.0 ('lenny'), or other distributions based on Debian, such as Ubuntu 10.04 LTS. The package name is **ps-captured**.



## Using an APT source

If you have your own APT package source set up, you can just place the package file in it, then use 'apt-get' to install it on target machines:

```
# apt-get install ps-captured
```

This is the best way to maintain a number of servers, since you can add upgrades to your distribution and have them automatically upgraded on the servers which use it.

## Installing the packages manually

If you want to install the package manually, use 'dpkg' to install it: (your version number may vary: tab completion is your friend!)

```
# dpkg -i ps-captured_1.1.0-1_i386.deb
```

If you are installing on a 64-bit system use the 'amd64' package instead of 'i386'.

## What happens underneath

Like most Debian server packages, the daemon package installs the server binary, the configuration file and the 'init.d' files in the usual places, then starts it up. For more details of what goes where, see the section on manual installation below.

---

## Installing on CentOS / Red Hat

The Red Hat package (.rpm) provided will install on CentOS 5, RHEL 5 and most versions of Fedora. The package name is **ps-captured**.

## Using a repository

If you have your own 'yum' repository set up, you can just place the package file in it, then use 'yum' to install it on target machines.

```
# yum install ps-captured
```

This is the best way to maintain a number of servers, since you can add upgrades to your distribution and have them automatically upgraded on the servers which use it.

## Installing the packages manually

If you want to install the package manually, and use 'rpm' to install it: (your version number may vary: tab completion is your friend!)

```
# rpm -i ps-captured_1.1.0-1.i386.rpm
```



## Manual installation

If you need to install Packet Ship Timeline on a distribution where the standard package doesn't work, and you can't get 'alien' to do it for you, you may need to manually install the software. In any case, it's useful and interesting to know what goes where...

If you have specially requested it, we will have supplied you with a zipped tarball of the software called something like `packet-ship-timeline-1.1.0-i386-deb.tgz`. This is a flat group of files which you need to copy manually – we haven't attempted to structure it to unpack at the root directory because that would be dangerous without understanding your system. As usual, to unpack the tarball, move it to an empty directory and type:

```
# tar xfvz packet-ship-timeline-1.1.0-i386-deb.tgz
```

You'll then have a directory full of the following files:

- `ps-captured` (server binary)
- `captured.cfg.xml` (configuration file)
- `ps-captured.init.d` (init.d start/stop script)
- `ps-captured.logrotate.d` (logrotate script)

## Binaries

The server binary, `ps-captured`, needs to be copied to `/usr/sbin/`, or your usual place for system binaries. It should be owned `root.root` and (for security) writeable only by `root`. There's no particular reason to stop ordinary users reading or executing it since they may wish to create their own non-privileged service – beware however that recording is a heavy load for a multi-user machine, and you may wish to prevent them running it for this reason.

```
# cp ps-captured /usr/sbin/  
# chown root.root /usr/sbin/ps-captured  
# chmod 755 /usr/sbin/ps-captured
```

## Configuration files

The configuration file `captured.cfg.xml` needs to be copied to a new directory `/etc/packetship`. Note that this location for the configuration file is 'hardwired' into the server binary, but can be overridden by passing an alternate configuration file as a command-line argument (this is the only command-line option it has).

```
# mkdir /etc/packetship  
# cp captured.cfg.xml /etc/packetship/
```

Any licence file purchased separately also needs to be copied into `/etc/packetship` as `licence.xml` (it will have a customer and machine-specific filename by default).

```
# cp licence-MyCo-345B452A0C29.xml /etc/packetship/licence.xml
```



## Start/stop scripts

If your system uses System V init.d scripts, you can copy the script provided into `/etc/init.d/`, and create the usual links from the `/etc/rc2.d` directory, plus `/etc/rc3.d`, `rc4.d` and so on as well if your system ever gets to runlevels above 2.

```
# cp ps-captured.init.d /etc/init.d/ps-captured
# ln -sf /etc/init.d/ps-captured /etc/rc2.d/S20ps-captured
```

## Log directories

By default – although this is configurable – the daemon puts its log files in `/var/log/packetship` – so this needs to be created:

```
# mkdir /var/log/packetship
```

## Capture directory

The conventional place to put captured data is `/var/lib/packetship/capture/`:

```
# mkdir -p /var/lib/packetship/capture
```

## Log rotation scripts

If you use ‘logrotate’ to maintain your logs, you can copy the `ps-captured.logrotate.d` script into the `/etc/logrotate.d` directory. The defaults should be sensible, but check your logrotate manual if you want to change them.

```
# cp ps-streamd.logrotate.d /etc/logrotate.d/ps-captured
```

---

## Trying it out

If you installed from the packages your daemon should already be running, but just for the experience, stop it now. If you installed manually, skip this bit.

```
# /etc/init.d/ps-captured stop
Stopping Packet Ship IPTV capture daemon: ps-captured.
```

Now restart the daemon:

```
# /etc/init.d/ps-captured start
Starting Packet Ship IPTV capture daemon: ps-captured.
```

The daemon should start cleanly with no errors here.





## Watching the processes

If you use 'ps axf' you should see two ps-captured processes:

```
# ps axf
...
27415 ?      Ss      0:00 /usr/sbin/ps-captured
27416 ?      Sl      0:00 \_ /usr/sbin/ps-captured
...
```

There are two processes because the daemon has an in-built “watchdog” parent process which will restart the server in the unlikely event it should fail. This can be disabled in configuration if another watchdog system (e.g. init) is being used.

You can also see what’s happening in the log:

```
# less /var/log/packetship/captured.log
```

It should be something like this:

```
Wed 08 Jun 14:35:58.889 [2]: Packet Ship capture daemon version 1.2.1 starting
Wed 08 Jun 14:35:58.889 [2]: Configuring server permanent state
Wed 08 Jun 14:35:58.890 [2]: Connecting to XMLMesh at 127.0.0.1:29167
Wed 08 Jun 14:35:58.891 [2]: Starting HTTP SOAP server at 0.0.0.0:11180
Wed 08 Jun 14:35:59.264 [2]: Configuring server dynamic state
Wed 08 Jun 14:35:59.264 [2]: Starting capture 'BBC1' from 225.0.7.1:11111 with 6 segments of 600
sec, and index
Wed 08 Jun 14:35:59.265 [2]: Starting capture 'BBC2' from 225.0.7.2:11111 with 6 segments of 600
sec, and index
Wed 08 Jun 14:35:59.265 [2]: Starting EPG capture 'PSB1' from 225.0.7.250:11111
Wed 08 Jun 14:35:59.265 [2]: EPG - could not open file /var/lib/packetship/capture/epg.xml for
reading: No such file or directory - assuming new start
```



## Systems Administration

The Packet Ship Timeline recorder, `ps-captured`, is a standard Unix daemon and is managed in the same way as any other. The following is a quick guide for systems administrators and anyone else who needs to start, stop, restart and observe the daemon as it is running.

---

### Binaries

The daemon is installed by default as `/usr/sbin/ps-captured`.

---

### init.d script

Like most daemons `ps-captured` comes with a System V init script `/etc/init.d/ps-captured`. This takes the following command parameters:

- **start**: Starts the daemon
- **stop**: Stops the daemon
- **restart**: Stops the daemon and then starts it again (hard restart)
- **reload**: Reloads configuration but keeps any existing captures running (soft restart)

For example:

```
# /etc/init.d/ps-captured start
# /etc/init.d/ps-captured reload
# /etc/init.d/ps-captured stop
```

The installation package installs links to the `init.d` script into the default `rc<n>.d` directories.

The “reload” command actually sends a `SIGHUP` to the daemon – this can be done directly if an automated way to reload after configuration is required.

---

### Configuration

The `ps-captured` daemon's core configuration file is `/etc/packetship/captured.cfg.xml`, containing a top-level `<captured>` element.

Fully licensed versions are enabled by a licence file in `/etc/packetship/licence.xml`. The same licence may also cover other Packet Ship products.

---

### Logging

By default `ps-captured` logs everything to `/var/log/packetship/captured.log`. It also installs a `logrotate` script in `/etc/logrotate.d/ps-captured` which rotates this on a weekly basis with a 2 week history.



Log lines consist of a date stamp and millisecond clock, a log level (in square brackets) and the log text. Filtering for “[1]” is a good way to detect errors for automatic monitoring.

---

## Processes & Threads

The daemon runs as two processes. The parent is a 'watchdog' process which will restart the child (which is the main operation) in the unlikely event it should exit unexpectedly. It will start doing this quite aggressively, but will then back off exponentially up to a minute between retries.

The daemon makes heavy use of threads. There is a baseline of around 10 threads, plus one thread for each capture or EPG stream.



## General server configuration

There are a number of general configuration options which are standard across all Packet Ship server daemons:

### Logging

The logging output for the server is configured with the **<log>** element at the top level of the configuration file.

The **<log>** element has the following attributes:

- The **level** attribute gives the level of logging produced, as follows:

Level	Output
0	Nothing
1	Errors only
2	Summary of major activities
3	Details of activities
4	<i>Debugging information</i>
5	<i>Full dumps of everything</i>

Production release versions (which you will receive by default) only have code to log up to level 3 – this makes them more efficient. You won't need a debug version unless we specifically ask you to give us information on an issue. The default level is 2, which is the level at which it is sensible to run most production systems.

- The **file** attribute gives the log file to log to, for production versions. The default for ps-captured is `/var/log/packetship/captured.log`.
- The **timestamp** attribute gives the format of the timestamp prefixed to each line in the log file. The format is that used by 'strftime' (see 'man strftime'), and can include both variables prefixed with % and fixed text. In addition, the following special fields can be used:

Field	Default log file
%*S	Seconds including milliseconds
%*L	Log level (1-5, as above)

The default is `"%a %d %b %H:%M:%*S [%*L]: "`, which produces lines like this:

```
Tue 29 Jan 14:30:04.582 [2]: ...
```

The following is the default **<log>** configuration for ps-captured as shipped, which is probably fine for the vast majority of installations.

```
...
<log level="2" file="/var/log/packetship/captured.log"
  timestamp="%a %d %b %H:%M:%*S [%*L]:"/>
...
```



---

## Background daemon

By default the server becomes a background daemon and detaches itself from the terminal that started it, and logs to the file specified above. However for testing it's sometimes convenient to have the server run in foreground on the terminal and output logging to it. This is controlled by the **daemon** attribute of a top-level **<background>** element. It defaults to “yes”, set it to “no” to run in foreground.

```
...  
  <background daemon="yes"/>  
...
```

---

## Watchdog restart

In the unlikely event the server should fail, there is a built-in 'watchdog' process that can restart it automatically. This is controlled by the **restart** attribute of a top-level **<watchdog>** element. It defaults to “yes”, set it to “no” to disable the watchdog restart.

```
...  
  <watchdog restart="yes"/>  
...
```

---

## User/group identity

The server daemon is usually started as user 'root' in order to allow it to obtain the necessary restricted network ports, enable real-time priority and other root-only functions. However, once it has done this, as an additional security precaution in case of attack it is able to drop its root permissions and become an ordinary user – typically user “pscap” and group “pscap”, which has permission to write to /var/lib/packetship/capture/ where it stores its capture files.

The user to switch to is configured with the **user** and **group** attributes of a **<security>** element at the top level of the configuration file:

```
...  
  <security user="pscap" group="pscap"/>  
...
```

If absent, the server remains as the user it is run as (usually 'root').

---

## Licence file

The licence file which governs the licensed features of the server is usually kept in /etc/packetship/licence.xml. Should there be any reason to change this, the location can be set with the **file** attribute of a **<licence>** element at the top level:

```
...  
  <licence file="/etc/packetship/licence.xml"/>  
...
```



## Multicast capture configuration

The multicast IPTV channels to be captured by the server are configured in a **<captures>** element in the top level **<captured>** element of `captured.cfg.xml`.

---

### Persistent capture state

The daemon stores a list of active captures to an external file so that they can be recreated if the daemon is restarted. The file to store them in is configured in **<disk file>** within the **<captures>** element:

```
<captures>
  <disk file="/var/lib/packetship/capture/captures.xml"/>
  ...
</captures>
```

---

### Capture configuration

Each capture is configured in a **<capture>** element within the **<captures>** element, with default parameters which apply to all captures specified in a **<default>** element. Each **<capture>** element has an **id** attribute which is usually the short channel name:

```
<captures>

  <default>
    ...
    default parameters for all captures
    ...
  </default>

  <!-- Specific captures -->
  <capture id="BBC1">

    ...
    specific parameters for this capture
    - usually only the source address
    ...

  </capture>

</captures>
```

Parameters specified in a specific **<capture>** override any specified in **<default>**.



## Shared capture parameters

The following parameters are usually specified in the **<default>** element, to be shared across all captures, but can be overridden in an individual **<capture>** if required.

### File prefix and suffix

The capture daemon records received IPTV data to a set of disk files, each named after the channel ID being recorded and the date and time the recording segment starts. The directory and any other prefix required can be specified in a **prefix** attribute of a **<file>** element, and a **suffix** in the **suffix** attribute of the same element. The default configuration is to store files in `/var/lib/packetship/capture` with a `.ts` prefix:

```
<file prefix="/var/lib/packetship/capture/" suffix=".ts"/>
```

This produces files of the form:

```
/var/lib/packetship/capture/BBC1.2011_01_29.18_30_00.ts
```

### Index suffix

If the recorded content is to be played back through the Packet Ship Streamline video server, an index file for the recording will be required for variable bitrate streaming (almost always required in broadcast signals) and trick mode (visual fast-forward / rewind). The suffix to apply to the main file to create the index filename is configured in a **suffix** attribute of an **<index>** element:

```
<index suffix=".psi2"/>
```

The index suffix `.psi2` is the standard one expected by the Streamline video server, and generates index files of the form:

```
/var/lib/packetship/capture/BBC1.2011_01_29.18_30_00.ts.psi2
```

If the index suffix is empty or left out, no indexes are produced. Note that live indexing is a significant proportion of the processor load involved in capture, so if the recordings are going to be used in some other system than Streamline, it is worth disabling index generation.

### Start mode

By default the capture will start immediately, but this can be suppressed if capturing is to be controlled by an external controller through XML messaging (see below). The start mode is set by the **mode** attribute of a **<start>** element:

```
<start mode="auto"/>
```

The default is `“auto”`. Set this to `“external”` for externally controlled startup.

### Source port

IPTV multicasts are often sent to a standard port number such as 11111 or 1234. The port of the multicast source (actually the destination port of the packet!) can be configured in a **port** attribute of a **<source>** element:

```
<source port="11111"/>
```

Note that the multicast address itself is capture-specific and described below.



## Segment length

The capture daemon records data in a continuous loop, segmented into files of a given duration – for example, 24 files of one hour each, giving a 24-hour loop. The length and number of files to retain are configured in **length** and **count** attributes of a **<segment>** element:

```
<segment length="1 hour" count="24"/>
```

The **length** can be specified in seconds or number of minutes, hours or days – e.g. “600”, “10 mins”, “3 hours”, “1 day”

## EPG history length

The length of the EPG history kept for a particular channel is configured with a **history** attribute of an **<epg>** element:

```
<epg history="1 day"/>
```

This is usually set to be the same as the total record time for the channel, although it can be set longer (or even shorter) if required. The **history** can be specified in seconds or simple units, as above.

## Data stall times

If the multicast data for a captured channel stops for a configurable length of time, the capture daemon will close the current file and wait until the data returns for a further length of time before opening a new one. The effect of this is to properly record gaps in the recorded timeline, which allows the server both to accurately seek to parts of the channel it does have, and also to properly indicate whether a particular programme is available or not.

The time to wait before assuming the channel has stalled is configured in a **timeout** attribute of a **<receive>** element within the capture (default 1 second). The time to wait for data to become stable before starting again is configured by a **resume** attribute in the same element (default 30 seconds):

```
<receive timeout="1 second" resume="30 seconds"/>
```

## Specific capture parameters

The following parameter can only be specified in an individual **<capture>** because it makes no sense to share it between captures:

### Source address

The address of the multicast source (actually the destination address of the packet!) can be configured in an address attribute of a **<source>** element, usually in dotted-quad form, although a hostname can be used if required:

```
<source address="225.10.1.1"/>
```

If the port varies between captures this can be specified in the same element:

```
<source address="225.10.1.1" port="5001"/>
```





## Playlist lead / tail times

If the video server requests a channel with no start time (e.g. `rtsp://server/tv/BBC1`), the server returns only back to a configurable 'lead time', allowing the client only to rewind that far. The lead time is set by the **current-lead** attribute of a **<playlist>** element in the capture, and defaults to 1 hour.

If the video server specifies a particular start time (e.g. `rtsp://server/tv/BBC1/20111225T150000Z`) the capture daemon returns a playlist covering the entire programme with some extra time at the start and end in case of early or late start. The 'lead' and 'tail' times are configured in **programme-tail** and **programme-lead** attributes of **<playlist>**, and both default to 5 minutes.

The capture daemon can refuse to serve a programme if there is less than a certain percentage available (either because of signal loss or server restarts). The minimum percentage availability is set in a **min-percent** attribute of **<playlist>**, and defaults to 98%.

```
<playlist programme-lead="5 minutes"
           programme-tail="5 minutes"
           current-lead="1 hour"
           min-percent="98" />
```



## EPG capture configuration

The capture of electronic programme guide (EPG) information is a separate process in the capture daemon which runs independently except for the fact that individual channel captures influence how long the EPG data is kept for, as described above.

The EPG data is built up both from the forward event data and the now/next programme data to create both a backward-looking (retrospective) and forward-looking EPG.

The capture daemon keeps the entire EPG data in memory to make user interface queries extremely fast, and stores a backup to disk regularly so that it can be recovered if the daemon has to be restarted for some reason.

The EPG capture is configured within an **<epg>** element in the top level **<captured>** element of `captured.cfg.xml`.

---

### Disk storage

The following parameters control how the in-memory EPG data is backed up onto disk:

#### EPG filename

The disk file in which the EPG data is backed up is specified in a **file** attribute of a **<disk>** element inside **<epg>**:

```
<disk file="/var/lib/packetship/capture/epg.xml"/>
```

#### Save interval

The interval at which to save the file is specified as the **interval** attribute of a **<save>** element:

```
<save interval="1 min"/>
```

The default of one minute is a reasonable compromise between not missing very short programmes and unduly hammering the disk.

#### History prune interval

The EPG capture process periodically prunes the in-memory data, removing data for programmes that are beyond the history length specified in the capture for that particular channel. The interval to run this process is specified as the **interval** attribute of a **<prune>** element:

```
<prune interval="1 hour"/>
```

There is no need to run this process very frequently, although it is not a hugely expensive operation should you wish to do so.

---

### Huffman-encoded tables

Freeview HD and Freesat HD in the UK use Huffman encoding of EPG data (EIT table). The encoding tables are only available from DTG or the BBC under NDA as a way to enforce



implementation of HDCP in HD set-top boxes, since they act as a form of (insecure) encryption of the EPG data.

Despite requesting them for use in legitimate server-based applications Packet Ship have not been able to obtain the official tables, nor could we distribute them if we did have them. However, Open Source reverse-engineered versions of the tables exist in the VDR project, and we have implemented a generic Huffman decoder that can read the same table format (see below). We do not know what form the official tables take, but they should be trivially convertible into the format Timeline currently imports. If (as it is to be hoped) the official tables are made public at some point in the future, we will most likely implement an importer for that format as well.

**Note: Packet Ship do not distribute the Freeview/Freesat HD Huffman tables in any form and the Timeline product provides Huffman decoding as a generic feature suitable for any Huffman encoded data. Users wishing to make use of Freeview/Freesat HD EPG data will need to source the tables themselves through official channels or otherwise, at their sole choice and liability.**

## Table location

The location of the Huffman table files is configured in a **<huffman tables>** path name within the **<epg>** element:

```
<epg>
...
  <huffman tables="/etc/packetship/freesat.t%i"/>
...
</epg>
```

The path name should contain a single “%i” string which represents the table number. Hence the default matches `/etc/packetship/freesat.t1` and `freesat.t2` for tables 1 and 2, which are the standard filenames for the Open Source tables.

## Table format

The Huffman tables are described in a simple text format in one or more table files. The use of Huffman encoding is triggered by the first character of the text string being 0x1F. The next byte is then taken as the table number to use, replacing %i in the format above. Freeview/Freesat use table numbers 1 and 2, so the two valid prefixes are 0x1F 0x01 and 0x1F 0x02.

The Huffman encoding is context sensitive – that is to say, the bit string for each character is dependent on what the previous character was. The character encoding used is UTF-8.

The format of an individual table file is as follows:

- The file is textual line based, terminated with either LF or CR-LF.
- Blank lines are ignored
- All other lines should contain a mapping consisting of 3 parts, separated by colons (':'), with an optional end of line colon:
  - the previous value output
  - a bit sequence in binary 1's and 0's
  - the next value to output



- Values are usually verbatim ASCII characters, but some special characters are also recognised:
  - START - used for the previous value where no data has yet been read
  - STOP - indicates end of data
  - ESCAPE - indicates escaping from Huffman encoding. The data is then read literally in 8-bit chunks, terminated by any non-top-bit-set character which is then output and used as the new previous value.
  - Values can also be specified in 0xNN hex format, e.g. 0x3a = ':'

For example (not real data):

```
START:10101:A:
A:00011:STOP:
A:11100100:N:
A:111100110:2:
A:11001010:0x3a:
A:110011010:ESCAPE:
B:00:A:
B:01:C:
B:101:e:
```

## EPG data streams

The capture daemon can get 'now and next' data from the capture streams themselves if it is being streamed, but for a full forward-looking EPG a separate EPG data feed will normally be required. Most IPTV head-ends can be configured to output the EPG data (EIT) on one or more separate multicasts, either one per multiplex (and sometimes only one multiplex) or a single one with all multiplexes combined. If data from only one multiplex is available it usually includes data for all the other multiplexes as well, but sometimes on a longer refresh interval.

The EPG data sources can be configured in **<stream>** elements inside a **<streams>** element in **<epg>**. Each stream has an **id** attribute and a **<source>** **address** and **port** configured in the same way as a capture:

```
<streams>
  <stream id="PSB1">
    <source address="225.11.1.1" port="11111"/>
  </stream>
</streams>
```

The ID is only used for logging, and is usually named after the multiplex being captured.

There is no defaulting mechanism for EPG captures because there is nothing but the source address to configure, and there are normally only a small number of them.



## Messaging configuration

The Timeline capture daemon provides a number of XML messaging interfaces for integration with external systems and other Packet Ship products. The daemon supports both conventional (document-based) SOAP over HTTP and also XMLMesh messaging, which is a peer-to-peer messaging bus architecture which also uses SOAP messages.

The capture daemon provides a set of message handlers which perform particular functions such as starting and stopping captures, locating recorded programmes and querying EPG data for user interface display. These can be mapped to particular HTTP URLs and/or XMLMesh subjects.

---

### HTTP Server

The standard SOAP HTTP server can be configured with an **<soap>** element at the top level of `captured.cfg.xml`. The port on which the server will listen for HTTP requests is set by a **port** attribute of a **<server>** element inside this. If required the local interface address to bind to can be specified in an **address** attribute, which defaults to "localhost".

```
<soap>
  <server port="11180" address="0.0.0.0"/>
</soap>
```

Note that the SOAP server is **not authenticated** and hence, if enabled on an external interface, should be protected by firewall rules from external access. For security it is disabled in the standard configuration.

---

### XMLMesh Connection

The capture daemon can also accept messages through an XMLMesh message broker. The XMLMesh connection is configured in an **<xmlmesh>** element at the top level of `captured.cfg.xml`. The XMLMesh message broker server to connect to is configured with the **host** and **port** attributes of a **<server>** element inside **<xmlmesh>**:

```
<xmlmesh>
  <server host="localhost" port="29167"/>
</xmlmesh>
```

XMLMesh connection is disabled in the standard configuration file so that the Timeline capture daemon can run independently, but should be enabled if integration with the Streamline video server is required. In this case, the `ot-xmlmesh` package also supplied in the installation kit should also be installed.

---

### Message bindings

The capture daemon provides a number of message handlers which perform various functions. These can be 'bound' to particular HTTP URLs and/or XMLMesh message subjects. If a handler is not bound on one or both of the channels, then it is disabled.



The message bindings are defined in a **<messages>** element at the top level of `captured.cfg.xml`. Each child element of this defines a message handler by name.

The current message handlers are as follows:

Handler name	Message type
<b>&lt;capture-list&gt;</b>	<b>&lt;ps:capture-list&gt;</b>
<b>&lt;capture-add&gt;</b>	<b>&lt;ps:capture-add&gt;</b>
<b>&lt;capture-remove&gt;</b>	<b>&lt;ps:capture-remove&gt;</b>
<b>&lt;capture-start&gt;</b>	<b>&lt;ps:capture-start&gt;</b>
<b>&lt;capture-stop&gt;</b>	<b>&lt;ps:capture-stop&gt;</b>
<b>&lt;capture-playlist&gt;</b>	<b>&lt;ps:capture-playlist&gt;</b>
<b>&lt;capture-epg-services&gt;</b>	<b>&lt;ps:capture-epg-services&gt;</b>
<b>&lt;capture-epg-listing&gt;</b>	<b>&lt;ps:capture-epg-listing&gt;</b>

For more information on each of these messages, please see the Application Note “AN-TL-1101 Timeline XML Messaging”.

### HTTP URL bindings

Each message handler can be bound to a particular HTTP URL with a **<soap>** element inside the handler element. The **url** attribute gives the URL (or URL glob pattern) to bind to. For example:

```
<soap>
...
<messages>
...
  <capture-start>
    <soap url="/capture-start"/>
  </capture-start>
...
</messages>
...
</soap>
```

### XMLMesh subject bindings

Each message handler can also be bound to a particular XMLMesh subject pattern with an **<xmlmesh>** element inside the handler element. The **subject** attribute gives the message subject (or subject glob pattern) to bind to. For example:

```
<soap>
...
<messages>
...
  <capture-start>
    <xmlmesh subject="packetship.capture.start"/>
  </capture-start>
...
</messages>
...
</soap>
```



The two forms can be combined to offer the service on both messaging interfaces:

```
<soap>
  ...
  <messages>
    ...
    <capture-start>
      <soap url="/capture-start"/>
      <xmlmesh subject="packetship.capture.start"/>
    </capture-start>
    ...
  </messages>
  ...
</soap>
```

Note that the XMLMesh bindings are disabled in the default configuration.