



Packet Ship *Timeline* IPTV Recorder

Application Note

AN-TL-1101
v.1.2

Server- and Client-side Interfacing to Timeline IPTV Recorder with XML Messages

Documents ps-captured 1.2.1 release

CONFIDENTIAL

Provided subject to terms of
NDA and/or Reseller Agreement

Not for release to end customers



Contents

Introduction.....	3
XMLMesh architecture.....	3
Messaging through traditional SOAP over HTTP.....	4
Capture list requests.....	5
Request structure.....	5
Response structure.....	5
Capture add requests.....	6
Request structure.....	6
Response structure.....	6
Capture remove requests.....	7
Request structure.....	7
Response structure.....	7
Capture start requests.....	8
Request structure.....	8
Response structure.....	8
Capture stop requests.....	9
Request structure.....	9
Response structure.....	9
Capture playlist requests.....	10
Request structure.....	10
Response structure.....	11
Capture EPG services requests.....	12
Request structure.....	12
Response structure.....	12
Capture EPG listing requests.....	14
Request structure.....	14
Response structure.....	14
Appendix: XMLMesh Message Interfaces.....	16
The ot-xmlmesh-cli package.....	16
Sending messages with ot-xmlmesh-send.....	16



Introduction

This Application Note describes how external applications can inspect and modify the capture configuration of a Timeline IPTV Recorder using either the XMLMesh message bus (ot-xmlmesh) or HTTP SOAP interfaces.

XMLMesh architecture

The ObTools XMLMesh message bus is a ‘middleware’ application which allows independent components, either on the same machine or on different ones, to communicate with SOAP XML messages on a publish-subscribe basis. Unlike a traditional Remote Procedure Call (RPC) system such as traditional SOAP-over-HTTP, CORBA or Java RMI, a message is not directed at any particular receiver, but it sent out with a subject which one or more receivers may choose to subscribe to, act upon, and reply to.

Subjects & Subscription

Each XMLMesh message has a textual subject which by convention is split up into words separated by dots, like a reverse domain name, or Java package name. To avoid clashes, it is recommended that the messages begin with the sender’s reversed domain name, with the allowance that ‘.com’ can be left out: Hence all Packet Ship messages begin “**packetship.**”, and all messages to do with Timeline IPTV recording are prefixed with “**packetship.capture.**”. To begin receiving messages, clients attach to the XMLMesh at a well-known TCP port, and request a subscription to a given subject pattern. The pattern can be expressed as a standard ‘glob’ pattern, such as “**packetship.***”. This pattern would receive every single Packet Ship message which passed through the system: Probably overkill, but interesting to try!

Request & Response

Messages can be sent and received in a one-way ‘fire and forget’ metaphor, or with a more conventional request/response. Each message has an ID, and an ‘RSVP’ flag which says whether or not a reply is expected. A receiver can simply respond to a message by sending back a reply quoting the request’s ID.

The response can either be a full document or a simple “OK” response of the form **<x:ok/>**

Note that it is quite possible for more than one receiver to subscribe to a request, but only the first one to reply will have its reply forwarded back to the sender. However, as we shall see, the others can observe what is going on without having to reply.

Message Bodies

Within the ‘envelope’ (which is actually SOAP format) which carries the subject, ID etc., is the body of the message itself, which is just an XML document. These are usually very simple – for example:

```
<ps:capture-add-request xmlns:ps="http://packetship.com/ns"
                        id="BBC1">
  <source address="225.0.7.1" port="11111"/>
</ps:capture-add-request>
```



The 'ps:' namespace is standard for all Packet Ship messages, and is assumed even if no **xmlns** attribute is given. Note that the name (tag) of the XML element is quite independent of the 'subject' quoted in the envelope, although they are usually related, for obvious reasons.

Messaging through traditional SOAP over HTTP

Alternatively (or additionally), for applications without a convenient XMLMesh interface, the same messages can be exchanged using conventional SOAP over HTTP. The 'ps-captured' daemon can be configured to provide a simple HTTP server at a given port number which accepts different message types under different URLs.

The requests and response are identical to the XMLMesh versions, except that success for simple messages is indicated by an **<ps:ok/>** element.



Capture list requests

XMLMesh subject: **packetship.capture.list.request**

SOAP URL: **http://<server>:<port>/capture-list**

The capture list request message lists all currently existing capture sources.

Request structure

The capture list request message consists solely of a **<ps:capture-list-request>** element.

Example request

```
<ps:capture-list-request xmlns:ps="http://packetship.com/ns"/>
```

Response structure

The response to a capture list request message consists of a **<ps:capture-list-response>** element if successful. This element has the same structure as a captures element as found within the daemon's configuration file.

Example response

```
<ps:capture-list-response>
  <capture id="BBC1">
    <start mode="auto"/>
    <source address="225.0.7.1" port="11111"/>
    <segment count="6" length="00:10:00"/>
    <epg history="24:00:00"/>
    <file prefix="/var/lib/packetship/capture/" suffix=".ts"/>
    <index suffix=".psi2"/>
  </capture>
  <capture id="BBC2">
    <start mode="auto"/>
    <source address="225.0.7.2" port="11111"/>
    <segment count="6" length="00:10:00"/>
    <epg history="24:00:00"/>
    <file prefix="/var/lib/packetship/capture/" suffix=".ts"/>
    <index suffix=".psi2"/>
  </capture>
  <capture id="BBC3">
    <start mode="auto"/>
    <source address="224.0.0.123" port="11111"/>
    <segment count="6" length="00:10:00"/>
    <epg history="24:00:00"/>
    <file prefix="/var/lib/packetship/capture/" suffix=".ts"/>
    <index suffix=".psi2"/>
  </capture>
</ps:capture-list-response>
```



Capture add requests

XMLMesh subject: **packetship.capture.add.request**

SOAP URL: **http://<server>:<port>/capture-add**

The capture add request message adds a capture source to the list of captures.

Request structure

The capture add request message consists of a **<ps:capture-add-request>** element. This element has the same structure as a captures/capture element as found within the daemon's configuration file.

Example request

```
<ps:capture-add-request xmlns:ps="http://packetship.com/ns"
id="BBC1">
  <source address="225.0.7.1" port="11111"/>
</ps:capture-add-request>
```

Response structure

The response to a capture add request message consists of a **<ps:ok>** (SOAP) or **<x:ok>** (XMLMesh) element if successful, or a SOAP Fault message if not.

Example response

```
<ps:ok/>
```



Capture remove requests

XMLMesh subject: **packetship.capture.remove.request**

SOAP URL: **http://<server>:<port>/capture-remove**

The capture remove request message removes a capture source from the list of captures. Any related EPG information will also be removed.

Request structure

The capture remove request message consists of a **<ps:capture-remove-request>** element. This element has an **id** attribute specifying which capture to remove. Only captures added with a capture add request can be removed in this manner.

Example request

```
<ps:capture-remove-request xmlns:ps="http://packetship.com/ns"
id="BBC1"/>
```

Response structure

The response to a capture remove request message consists of a **<ps:ok>** (SOAP) or **<x:ok>** (XMLMesh) element if successful, or a SOAP Fault message if not.

Example response

```
<ps:ok/>
```



Capture start requests

XMLMesh subject: **packetship.capture.start.request**

SOAP URL: **http://<server>:<port>/capture-start**

The capture start request message starts a previously configured external mode capture recording to disk.

Request structure

The capture start request message consists of a **<ps:capture-start-request>** element. This element has a channel **id** attribute specifying which channel to start recording and an optional **duration** attribute – if specified this will limit the recording period. An optional path can be given in the content of the element. This is equivalent to the capture file prefix configuration setting, and will override it for the recording. Only captures with a start mode of **external** can be started in this manner.

Example request

```
<ps:capture-start-request xmlns:ps="http://packetship.com/ns"
id="BBC1" duration="1 hour">
  /usr/local/share/record/news
</ps:capture-start-request>
```

Response structure

The response to a capture start request message consists of a **<ps:ok>** (SOAP) or **<x:ok>** (XMLMesh) element if successful, or a SOAP Fault message if not.

Example response

```
<ps:ok/>
```




Capture stop requests

XMLMesh subject: **packetship.capture.stop.request**

SOAP URL: **http://<server>:<port>/capture-stop**

The capture stop request message stops a capture recording that was started with a capture start request.

Request structure

The capture stop request message consists of a **<ps:capture-stop-request>** element. This element has a channel **id** attribute specifying which channel to stop recording.

Example request

```
<ps:capture-stop-request xmlns:ps="http://packetship.com/ns"
id="BBC1"/>
```

Response structure

The response to a capture stop request message consists of a **<ps:ok>** (SOAP) or **<x:ok>** (XMLMesh) element if successful, or a SOAP Fault message if not.

Example response

```
<ps:ok/>
```



Capture playlist requests

XMLMesh subject: **packetship.capture.playlist.request**

SOAP URL: **http://<server>:<port>/capture-playlist**

The capture playlist request message returns a list of captured files and is used for Streamline video server integration.

Request structure

The capture playlist request message consists of a **<ps:capture-playlist-request>** element. Within this element are children identifying the channel and time to be queried.

The **id** attribute of the **<ps:channel>** element specifies the channel id.

The time period can be specified in one of two ways:

The **id** attribute of the optional **<ps:event>** element specifies the event ID for a particular programme. The server will add a configurable lead and tail to this and return a playlist which covers it.

Alternatively the **time** attribute of the optional **<ps:start>** element specifies the earliest time for which to return results for, as an ISO timestamp (the parser is however lenient for missing "T", seconds etc.). The absolute end of the period can be specified with the **time** attribute of an optional **<ps:end>** element, or alternatively the **time** attribute of an optional **<ps:duration>** element to specify a relative duration. The duration can be specified in seconds or in units – e.g. "30 mins", "1 hour".

If neither a **<ps:event>** nor a **<ps:start>** is given, the server will return a playlist for the near-live capture, rewound for a configurable lead time.

Example requests

Near live:

```
<ps:capture-playlist-request xmlns:ps="http://packetship.com/ns">
  <ps:channel id="BBC1"/>
</ps:capture-playlist-request>
```

Specified event

```
<ps:capture-playlist-request xmlns:ps="http://packetship.com/ns">
  <ps:channel id="BBC1"/>
  <ps:event id="1125"/>
</ps:capture-playlist-request>
```

Specified start time, to current time

```
<ps:capture-playlist-request xmlns:ps="http://packetship.com/ns">
  <ps:channel id="BBC1"/>
  <ps:start time="2011-06-18T10:00:00Z"/>
</ps:capture-playlist-request>
```



Specified start time and end time

```
<ps:capture-playlist-request xmlns:ps="http://packetship.com/ns">
  <ps:channel id="BBC1"/>
  <ps:start time="2011-06-18T10:00:00Z"/>
  <ps:end time="2011-06-18T10:30:00Z"/>
</ps:capture-playlist-request>
```

Specified start time and duration

```
<ps:capture-playlist-request xmlns:ps="http://packetship.com/ns">
  <ps:channel id="BBC1"/>
  <ps:start time="2011-06-18T10:00:00Z"/>
  <ps:duration time="30mins"/>
</ps:capture-playlist-request>
```

Response structure

The response to a capture add request message consists of a **<ps:capture-playlist-response>** element. Within this is a **<ps:playlist>** element and a **<ps:start>** element.

The **<ps:playlist>** element contains children **<ps:file>** elements that have a **path** attribute containing the file path of a stream recording.

The **npt** attribute of the **<ps:start>** element specifies the NPT at which the playlist begins.

Example response

```
<ps:capture-playlist-response>
  <ps:playlist>
    <ps:file path="/record/BBC1.2011_05_24.14_58_15.ts"/>
    <ps:file path="/record/BBC1.2011_06_01.15_00_00.ts"/>
  </ps:playlist>
  <ps:start npt="1903.2"/>
</ps:capture-playlist-response>
```



Capture EPG services requests

XMLMesh subject: **packetship.capture.epg.services.request**

SOAP URL: **http://<server>:<port>/epg-services**

The capture EPG services request message returns a list of EPG services Timeline has found EPG information for.

Request structure

The capture EPG services request message consists of a **<ps:capture-epg-services-request>** element.

Example request

```
<ps:capture-epg-services-request  
  xmlns:ps="http://packetship.com/ns"/>
```

Response structure

The response to a capture EPG services request message consists of a **<ps:capture-epg-services-response>** element. This contains a **<ps:epg-services>** element, which in turn contains children **<ps:epg-service>** elements.

The **id** attribute of the **<ps:epg-service>** element specifies its channel id.

The **name** attribute of the **<ps:epg-service>** element specifies the channel name.

The **provider** attribute of the **<ps:epg-service>** element specifies the network provider of the service.

The **service-id** attribute of the **<ps:epg-service>** element specifies the MPEG service id of the service.

The **base-time** attribute of the **<ps:epg-service>** element specifies the time from which EPG event information for the service exists.

The **history-length** attribute of the **<ps:epg-service>** element specifies the amount of time historical EPG event information for the service will be kept.



Example response

```
<ps:capture-epg-services-response
xmlns:ps="http://packetship.com/ns">
  <ps:epg-services>
    <ps:epg-service base-time="2011-06-20T15:00:00Z" history-
length="24:00:00" id="BBC1" name="BBC ONE" provider="BBC" service-
id="4162"/>
    <ps:epg-service base-time="2011-06-20T15:00:00Z" history-
length="24:00:00" id="BBC2" name="BBC TWO" provider="BBC" service-
id="4287"/>
    <ps:epg-service base-time="2011-06-20T15:00:00Z" history-
length="24:00:00" id="BBCNEWS" name="BBC NEWS" provider="BBC"
service-id="4352"/>
  </ps:epg-services>
</ps:capture-epg-services-response>
```



Capture EPG listing requests

XMLMesh subject: **packetship.capture.epg.listing.request**

SOAP URL: **http://<server>:<port>/epg-listing**

The capture EPG listing request message returns EPG event listings for a requested set of services over a given time period.

Request structure

The capture EPG services request message consists of a **<ps:capture-epg-listing-request>** element. This contains an optional **<ps:services>** element, and an optional **<ps:start>** and **<ps:end>** element. The **<ps:services>** element contains **<ps:service>** elements.

Each service listed in a **<ps:service>** element will have its EPG event listing returned. If the **<ps:services>** element is absent or empty, the information for all services will be returned.

The **id** attribute of the **<ps:service>** specifies the channel id of the service.

The **time** attribute of the **<ps:start>** element specifies the start time of the period for which to return EPG event information for. If absent it's value will be the current time.

The **time** attribute of the **<ps:end>** element specifies the end time of the period for which to return EPG event information for. If absent it's value will equal that of **<ps:start>**.

Example request

```
<ps:capture-epg-listing-request xmlns:ps="http://packetship.com/ns">
  <ps:services>
    <ps:service id="BBC1"/>
    <ps:service id="BBC2"/>
  </ps:services>
  <ps:start time="2011-06-21T15:30:00Z"/>
  <ps:end time="2011-06-21T15:30:00Z"/>
</ps:capture-epg-listing-request>
```

Response structure

The response to a capture EPG listing request message consists of a **<ps:capture-epg-listing-response>** element. Within this is a **<ps:epg-listing>** element that contains a collection of **<ps:epg-service>** elements, each of which contain a **<ps:epg-events>** element. Each **<ps:epg-events>** element contains a collection of **<ps:epg-event>** elements.

For information about the attributes of the **<ps:epg-service>** element please see the documentation about capture EPG services responses.

The **id** attribute of the **<ps:epg-event>** element specifies the ID of the EPG event.

The **name** attribute of the **<ps:epg-event>** element specifies the name of the EPG event.



The **description** attribute of the **<ps:epg-event>** element specifies the description of the EPG event.

The **start** attribute of the **<ps:epg-event>** element specifies the start time of the EPG event.

The **duration** attribute of the **<ps:epg-event>** element specifies the duration of the EPG event.

The **capture** attribute of the **<ps:epg-event>** element specifies whether or not the event has been recorded to disk.

Example response

```
<ps:capture-epg-listing-response>
  <ps:epg-listing>
    <ps:epg-service base-time="2011-06-20T15:00:00Z" history-
length="24:00:00" id="BBC1" name="BBC ONE" provider="BBC" service-
id="4162">
      <ps:epg-events>
        <ps:epg-event id="1124" capture="false" description="Live
coverage continues ..." duration="04:15:00" name="Wimbledon 2011"
start="2011-06-21T12:45:00Z"/>
        <ps:epg-event id="1125" capture="true" description="The
latest national and ..." duration="00:30:00" name="BBC News at Six"
start="2011-06-21T17:00:00Z"/>
        <ps:epg-event id="1126" capture="true" description="The
latest news ..." duration="00:30:00" name="Spotlight" start="2011-06-
21T17:30:00Z"/>
        <ps:epg-event id="1127" capture="true" description="Alex and
Matt are joined by Zoe Ball..." duration="00:30:00" name="The One
Show" start="2011-06-21T18:00:00Z"/>
        <ps:epg-event id="1128" capture="true" description="Masood's
desperate attempt ..." duration="00:30:00" name="EastEnders"
start="2011-06-21T18:30:00Z"/>
        <ps:epg-event id="1129" capture="true" description="In
Between Days: Medical drama ..." duration="01:00:00" name="Holby
City" start="2011-06-21T19:00:00Z"/>
      </ps:epg-events>
    </ps:epg-service>
    <ps:epg-service base-time="2011-06-20T15:00:00Z" history-
length="24:00:00" id="BBC2" name="BBC TWO" provider="BBC" service-
id="4287">
      <ps:epg-events>
        <ps:epg-event id="5903" capture="false" description="CBBC.
Sam and Mark ..." duration="00:30:00" name="Copycats" start="2011-06-
21T16:20:00Z"/>
        <ps:epg-event id="5904" capture="false" description="Sue
Barker presents ..." duration="02:10:00" name="Wimbledon 2011"
start="2011-06-21T16:50:00Z"/>
        <ps:epg-event id="5905" capture="false" description="John
Inverdale and guests ..." duration="01:00:00" name="Today at
Wimbledon" start="2011-06-21T19:00:00Z"/>
      </ps:epg-events>
    </ps:epg-service>
  </ps:epg-listing>
</ps:capture-epg-listing-response>
```



Appendix: XMLMesh Message Interfaces

To send and receive XMLMesh messages requires a message interface to properly format the message, connect to the bus and send and receive the message data. The XMLMesh platform currently supports the following message interfaces:

- A native library interface in C++ (initiating and responding)
- A native library interface in C (initiating only)
- A native PHP library (initiating only)
- A native Java library (initiating only)
- A generic pair of command-line tools for send and receive, which can be called/called by any scripting language such as Perl, shell or Python

Because it is the most general, this document describes the last option; using the others will be similar but in the native syntax of the calling language.

The ot-xmlmesh-cli package

Although it is not part of the standard Packet Ship release, we can supply on request an additional package, 'ot-xmlmesh-cli'. This is installed just like the standard packages:

```
# dpkg -i ot-xmlmesh-cli_1.2.0-1_i386.deb
```

This installs two utilities in /usr/bin/: **ot-xmlmesh-send** and **ot-xmlmesh-receive**. For interfacing to the cache you only need the former.

Sending messages with ot-xmlmesh-send

Running ot-xmlmesh-send with no arguments provides the following usage message:

```
Usage:
  ot-xmlmesh-send [options] <subject> [<file>]

Reads message from <file> or stdin, and sends it with the given subject
May output response to stdout if requested
Result code 0 for success, 1 for message failure, 2 for fatal error

Options:
  -c --check      Request response and check for OK, or output error to stderr
  -r --response   Request response and output body to stdout
  -s --soap       Show full SOAP response (only if -r)
  -v --verbose    More logging
  -q --quiet      No logging, even on error
  -h --host       Set XMLMesh host (default 'localhost')
  -p --port       Set XMLMesh port (default 29167)
  -? --help      Output this usage
```




Sending a capture list request and receiving the response

The capture list request returns a full response message, so we need to use the full form of the request, using the '-r' or '--response' option. It outputs any response it receives to standard output, even if it's an error (in which case it will be in the form of an **<env:fault>** element containing a SOAP fault structure):

```
$ ot-xmlmesh-send -r "packetship.capture.list.request"
<ps:capture-list-request xmlns:ps="http://packetship.com/ns"/>
[ctrl-D]
<ps:capture-list-response>
  <capture id="BBC1">
    <start mode="auto"/>
    <source address="225.0.7.1" port="11111"/>
    <segment count="6" length="00:10:00"/>
    <epg history="24:00:00"/>
    <file prefix="/var/lib/packetship/capture/" suffix=".ts"/>
    <index suffix=".psi2"/>
  </capture>
  <capture id="BBC2">
    <start mode="auto"/>
    <source address="225.0.7.2" port="11111"/>
    <segment count="6" length="00:10:00"/>
    <epg history="24:00:00"/>
    <file prefix="/var/lib/packetship/capture/" suffix=".ts"/>
    <index suffix=".psi2"/>
  </capture>
  <capture id="BBC3">
    <start mode="auto"/>
    <source address="224.0.0.123" port="11111"/>
    <segment count="6" length="00:10:00"/>
    <epg history="24:00:00"/>
    <file prefix="/var/lib/packetship/capture/" suffix=".ts"/>
    <index suffix=".psi2"/>
  </capture>
</ps:capture-list-response>
```