SES

SAT>IP

Protocol Specification

Version 1.1

Author	Date	Version	Comment
SES S.A.	12.10.2011	1.1	
British Sky Broadcasting Ltd			
Craftwork ApS			

Table of Contents

Acronyms

CSV	Comma Separated Values
DHCP	Dynamic Host Configuration Protocol
DiSEqC	Digital Satellite Equipment Control
DLNA	Digital Living Network Alliance
DSL	Digital Subscriber Line
DVB	Digital Video Broadcasting
DVB-S	DVB for Satellite
DVR	Digital Video Recorder
FEC	Forward Error Correction
GENA	General Event Notification Architecture
HTML	HyperText Markup Language
HTTP	Hyper Text Transfer Protocol
HSPA	High Speed Packet Access
IF	Intermediate Frequency
IGMP	Internet Group Management Protocol
IP	Internet Protocol
LAN	Local Area Network
LNB	Low Noise Block
MDU	Multi Dwelling Unit
MIME	Multipurpose Internet Mail Extensions
MPEG	Moving Picture Experts Group
MPTS	Multiple Program Transport Stream
MUDP	Multicast UDP
PID	Packet Identifier
PLC	Power Line Communication
PoE	Power over Ethernet
PSK	Phase Shift Keying
PVR	Personal Video Recorder
QPSK	Quaternary Phase Shift Keying
RFC	Request For Comments
RTP	Real-time Transport Protocol
RTCP	Real-time Transport Control Protocol
RTSP	Real Time Streaming Protocol
SDES	Source Description
SDP	Session Description Protocol
SI	Service Information
SMATV	Satellite Master Antenna Television
SOAP	Simple Object Access Protocol
SPTS	Single Program Transport Stream
SR	Sender Report
SSDP	Simple Service Discovery Protocol
STB	Set-Top-Box
TCP	Transport Control Protocol
TS	Transport Stream
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
UPnP AV	UPnP Audio and Video
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTF	UCS Transformation Format
WLAN	Wireless LAN
XML	Extensible Markup Language

References

UPnP Forum

[1] UPnP Device Architecture 1.1

DLNA

[2] DLNA Networked Device Interoperability Guidelines

IETF

[3] RFC 2131 – DHCP (Dynamic Host Configuration Protocol)

[4] RFC 2250 – RTP Payload Format for MPEG1/MPEG2 Video

[5] RFC 2279 – UTF-8, a transformation format of ISO 10646

[6] RFC 2326 – Real Time Streaming Protocol (RTSP)

[7] RFC 3376 – Internet Group Management Protocol, Version 3

[8] RFC 3550 – RTP: A Transport Protocol for Real-Time Applications

[9] RFC 3927 – Dynamic Configuration of IPv4 Link-Local Addresses

[10] RFC 4566 – SDP: Session Description Protocol

[11] draft-cai-ssdp-v1-03 – Simple Service Discovery Protocol/1.0

ITU / ISO

[12] ITU-T Recommendation H.222.0 / ISO/IEC 13818-1: "Information Technology - Generic Coding of moving pictures and associated audio information: Systems"

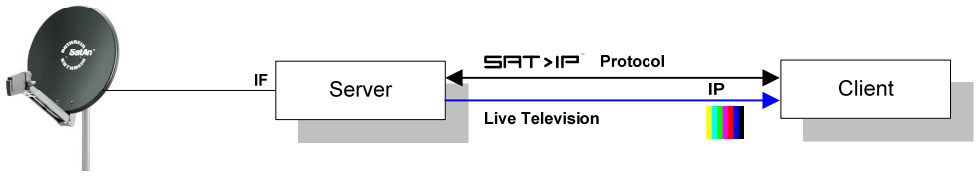
ETSI / DVB

[13] ETSI TS 101 154 V1.9.1; Digital Video Broadcasting (DVB); Specification for the use of Video and Audio Coding in Broadcasting Applications based on the MPEG-2 Transport Stream

1 Introduction

This document describes the SAT>IP Protocol and its usage.

The SAT>IP protocol provides a standardised way for IP Clients to access live media broadcasts from satellite reception servers on an IP network.



SAT>IP Clients

SAT>IP Clients may be – DVB compliant Set-Top-Boxes (STBs) with an IP interface or – Software applications running on programmable hardware such as Tablets, PCs, Smartphones, Connected Televisions. SAT>IP Clients always feature the possibility of rendering live television programs.

SAT>IP Servers

SAT>IP servers forward live television programs to SAT>IP clients. Servers may take various forms from large MDU headends covering whole buildings or cities, over in-home IP multiswitches, simple IP adapters or Master STBs to ultimately IP LNBs.

1.1 SAT>IP Concept

Unlike in today's satellite distribution schemes, the SAT>IP architecture allows the reception of satellite television programs also on devices which do not have a satellite tuner directly built-in. Satellite tuners and demodulators are moved into SAT>IP server devices. Clients only need to feature an IP interface and a video decoder.

This means that the reception of satellite delivered programming can be dealt with purely in software, provided a SAT>IP server is available on a network.

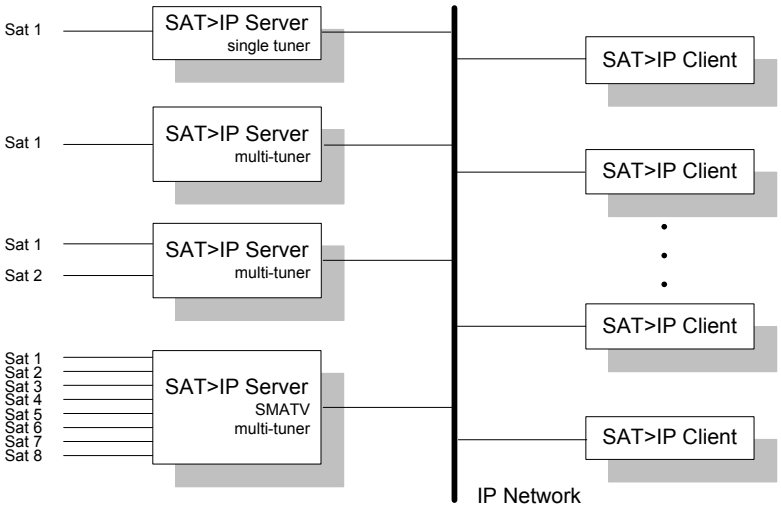
Satellite programs become available on devices which would never be capable of supporting satellite TV otherwise e.g. Tablets.

From the distribution point of view, satellite distribution becomes physical layer agnostic and satellite services can be forwarded over all the latest types of wired or wireless technologies such as Powerline (PLC), Wireless LANs, Optical Fiber Distribution, etc.

1.2 Network Topology

The SAT>IP protocol is designed to suit different application scenarios. The same SAT>IP client can communicate with various types of live media servers ranging from single satellite, single tuner servers to multi-satellite multi-tuner servers. SAT>IP clients can work unmodified in single home scenarios as well as in SMATV or large community systems.

The number of clients that can be simultaneously supported depends on the particular server implementation. Large servers can potentially serve an unlimited number of SAT>IP clients.



1.3 Thick Clients vs Thin Clients

The current version of the SAT>IP protocol was designed as a protocol for so-called Thick Clients to be able to communicate with different types of server devices.

We call Thick Clients hardware or software based implementations of DVB compliant reception devices. Thick Clients implement the typical DVB STB functionality such as decoding of DVB service information with a DVB service logic but expect services to be provided via IP rather than via an IF interface. Thick Clients have integrated DVB table parsers and expect DVB compliant transport streams to be forwarded to them. SAT>IP Thick Clients are a natural evolution of current generation DVB receivers and require only minimal adaptation in order to function in the SAT>IP environment.

The simplest Thick Clients can be normal DVB STBs from the horizontal market or from the vertical operator market that have an IP interface and can accept DVB programs delivered via the IP interface as well as via the analogue IF interface.

Thick Clients can be contrasted with Thin Clients. Thin Clients are receivers or renderers that do not implement DVB specific features (thus are not capable of understanding DVB Service Information). Thin Clients are DLNA type media players or other DLNA compliant devices. Such clients may be the object of a future release of this specification but are not addressed by this document.

The Thick Client Protocol was however designed in such a way as to use the same protocol features as the corresponding Thin Client will require. Both approaches are therefore complementary and the protocol suites were designed such as to be able to support DLNA clients later on.

It is therefore possible to implement a Thick Client that uses the SAT>IP protocol for accessing live media programs and the DLNA protocol for accessing stored network content, without having a protocol overlap or having to implement certain features in different ways.

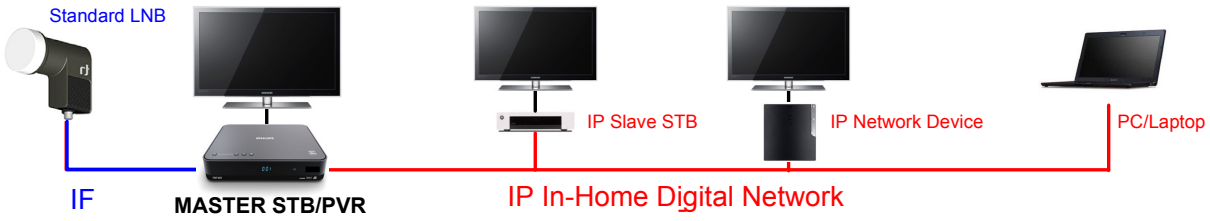
2 Usage Scenarios

The SAT>IP protocol has been designed with the following use cases in mind. This listing is not exhaustive, it is simply meant to describe the most common usage scenarios:

2.1 Multi-Room / Master-Slave / Whole Home DVR

A Master STB receives DVB-S2 satellite programming via its standard IF interfaces and forwards live television programs to one or more client STBs via an in-home IP network. Clients can also be dedicated software applications running on programmable devices (PCs, smartphones, tablets, etc.).

The SAT>IP protocol specifies how clients communicate with the master STB in order to access live streams.



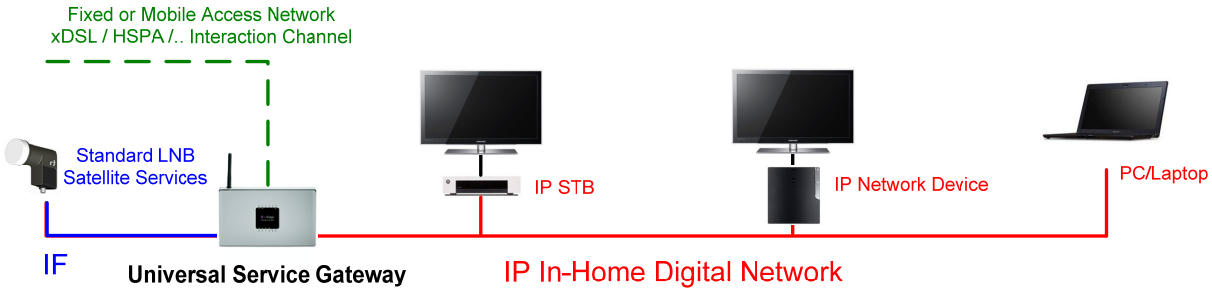
In the Whole Home DVR scenario the master STB additionally features the possibility of recording content. This document does not specify how recorded content is accessed nor how content scheduling is handled. These use cases are described in the DLNA protocol specifications. The SAT>IP protocol purely describes access to live media content.

2.2 IP Adapter / IP Converter

The IP Adapter is purely a converter device which converts (tunes, demodulates and demultiplexes) satellite IF signals towards IP without featuring itself the possibility to be directly connected to a display. The IP Adapter does not feature a built-in video decoder. IP Adapters may exist in various sizes and configurations. Some IP Adapters may feature a single tuner. Others may feature multiple tuners in order to serve multiple concurrent programs. IP Adapters can be situated anywhere in-home where IF satellite distribution meets with IP network connectivity.

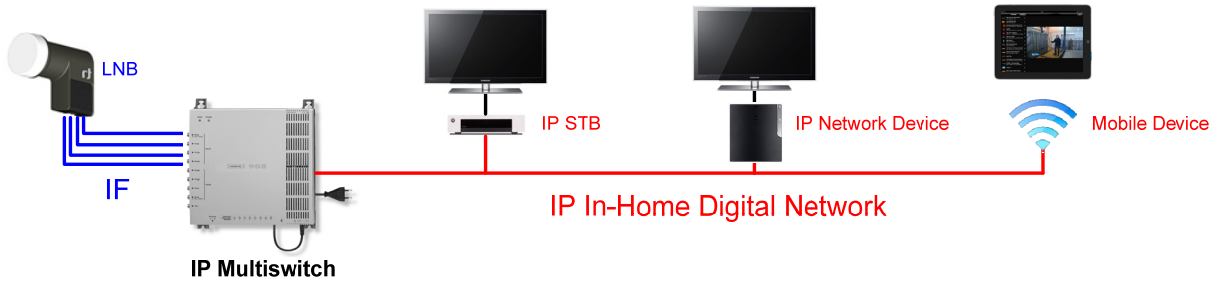
2.3 Universal Service Gateway

The Universal Service Gateway is an Internet Access device that features Fixed or Mobile Access Modems plus Satellite Reception Hardware. It combines access to both broadband and broadcast networks in one device.



2.4 IP Multiswitch

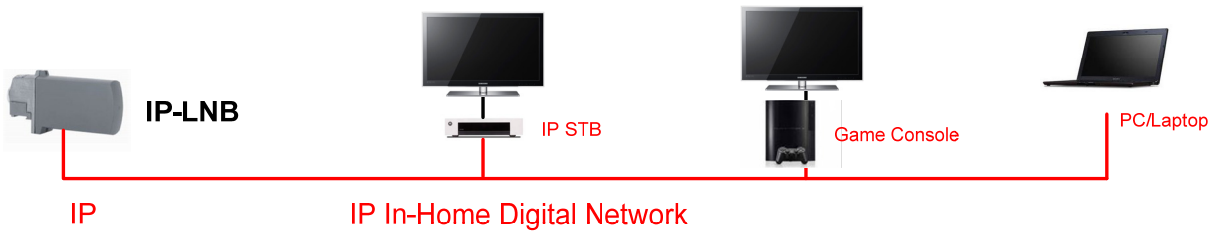
The IP Multiswitch is located close to the satellite antenna(s) and converts satellite programming towards IP for in-home distribution.



No assumptions are made on the number of satellite antenna inputs to the multiswitch nor on the number of tuners and demodulators within. The IP Multiswitch understands the SAT>IP protocol on its IP interface and is capable of answering requests coming from IP network clients. Services can be forwarded from the IP Multiswitch via PLC, WLAN, Cabled or Optical bridges.

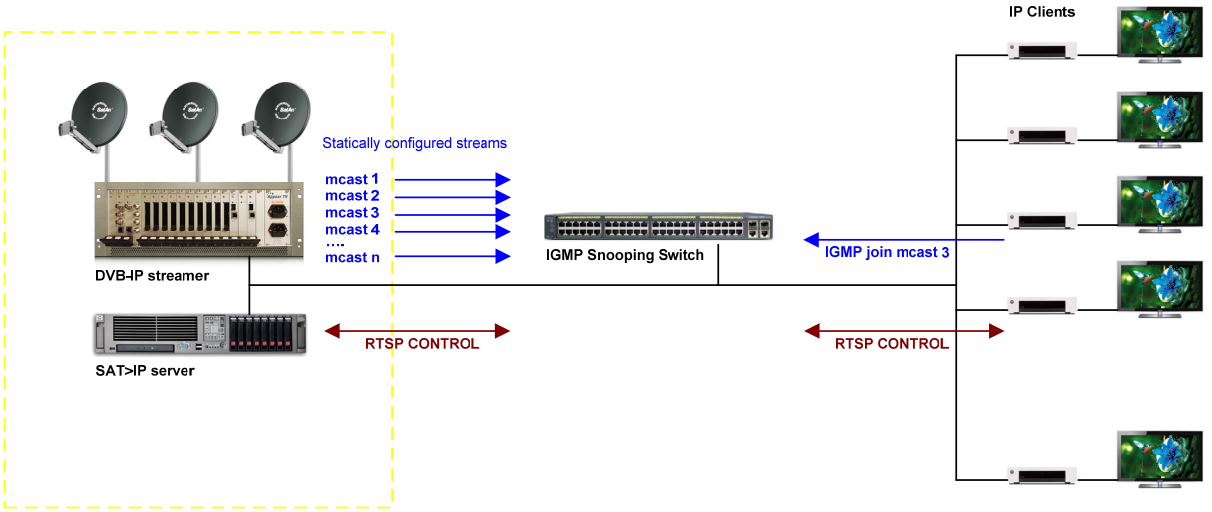
2.5 IP LNB

The end goal of the SAT>IP concept is to bring to the market IP LNB devices which feature direct Ethernet connectivity and are powered via PoE. This new generation of LNBs would no longer be connected via coaxial cable and would function as live media servers for the in-home digital network.



2.6 IP based SMATV / Multi-Dwelling Units

The SAT>IP protocol was designed in order to support also very large satellite reception installations. Such installations are often found in Multi-Family Housing Units, Communal Dish Installations, Hotels, Hospitals, Fiber based Greenfield Satellite Deployments.



The same protocol that allows a STB to control smaller SAT>IP installations, also allows such STBs to work on very large IP based networks. Invisibly to such STBs, MDU server installations may consist of professional DVB-IP stream server gateways and professional control servers.

MDU installations can be configured to run statically or dynamically as will be described below.

3 Protocol Specification

3.1 General

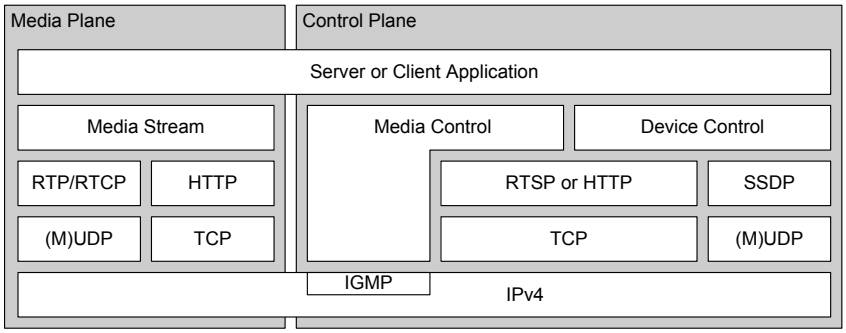
The SAT>IP protocol allows IP based clients to interact/communicate with IP based satellite servers for live media forwarding.

SAT>IP builds on industry standards and does complement those only where necessary i.e. where there are no established satellite specific extensions.

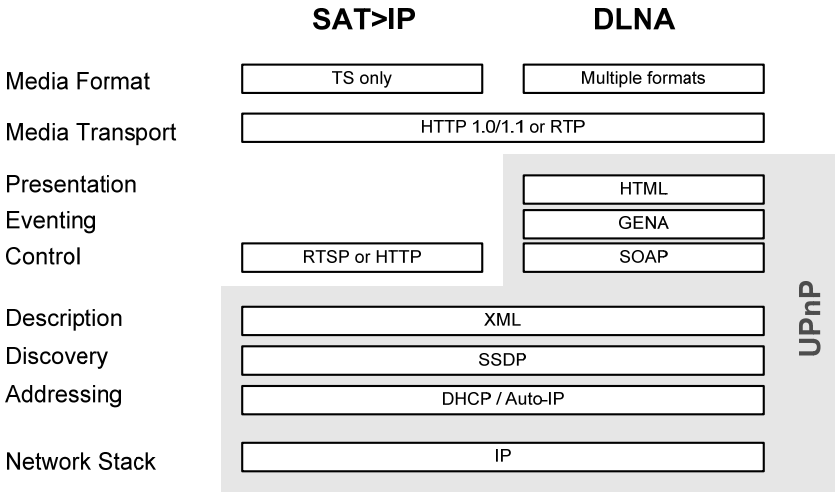
The SAT>IP protocol makes use of:

- UPnP for Addressing, Discovery and Description,
- RTSP or HTTP for Control,
- RTP or HTTP for Media Transport.

The SAT>IP protocol stack is organised in the following way:



SAT>IP uses a subset of the UPnP/DLNA architecture and protocols [1] [2] and SAT>IP devices can be extended to also become DLNA devices. As an example a SAT>IP client could access live media streams through the SAT>IP protocol and access recorded media streams through DLNA.



SAT>IP devices successively go through the following phases: Addressing, Discovery, Description, Control and finally Media Transport.

3.2 Addressing

Addressing is the process for a SAT>IP device (client or server) to obtain a network address. This is a pre-requisite before any communication can take place. SAT>IP follows the UPnP specification [1] by offering two options for address acquisition:

3.2.1 DHCP

Every SAT>IP device shall have a Dynamic Host Configuration (DHCP) client and search for a DHCP server when the device is first connected to the network [3]. The device from that point onwards shall use the IP address assigned to it.

3.2.2 Auto-IP

If no DHCP servers can be located, the network is unmanaged and SAT>IP devices shall autoconfigure by using the Auto-IP process to get an IP address [9].

3.3 Discovery

During the discovery phase SAT>IP servers advertise their presence to other SAT>IP servers and clients.

When joining a network, SAT>IP clients search the network for available SAT>IP servers.

3.3.1 SSDP

Discovery in SAT>IP relies on the Simple Service Description Protocol SSDP [11] as specified in the UPnP Device Architecture 1.1 [1].

As a minimum:

- a SAT>IP server is a UPnP Device and a UPnP Control Point,
- a SAT>IP client is a UPnP Control Point.

The device type of a SAT>IP device is:

urn:**ses-com**:device:**SatIPServer**:**1**

where:

- "ses-com" is the domain-name,
- "SatIPServer" is the deviceType name,
- and "1" is the version of the device type.

3.3.2 Multicast Address Range

In SAT>IP each server issues multicast addresses from its own multicast address range (which is not conflicting with multicast address ranges from other SAT>IP servers). This multicast address range is uniquely determined by a device_id value that each server allocates to itself during the device advertisement phase. The address range for a given SAT>IP server is determined as follows:

239. <device_id> . <0-254> . <0-254>

Example: the SAT>IP server with device_id=3 will issue addresses from the range: 239.3.x.x

3.3.3 Server Advertisements

SAT>IP servers joining the network multicast NOTIFY ssdp:alive messages containing the device_id in their header field. The initial device_id value is read from persistent memory of the server (last configured value).

Other SAT>IP servers already on the network (acting as control points) listen for these announcements.

As long as the device_id value received by these servers does not conflict with their own device_id value, they will not react.

In case of a device_id conflict, a server needs to defend its own device_id, by sending a unicast M-SEARCH message (within 1 second) containing the in-use device_id to the joining server. The joining server acknowledges receipt via a 200 OK response repeating the current device_id value.

As a consequence, the joining server generates a new device_id value different from the previous one and sends again a NOTIFY ssdp:alive message (multicast) containing the new device_id value.

If no unicast M-SEARCH message with a conflicting device_id field is received by the joining server within the timeout period (5 seconds), the joining server allocates itself the new device_id value and stores it in persistent memory.

A server leaving the network sends a NOTIFY ssdp:byebye message containing its device_id.

The process above is designed for the following mode of operation: on initial installation SAT>IP servers are connected to the same network and powered up one after the other. Each server will therefore sequentially acquire its own device_id value. After a power failure, server devices will simply read the last configured device_id value from persistent memory and re-boot more rapidly.

3.3.4 Client Search Requests

The multicast M-SEARCH message is issued by SAT>IP clients to discover SAT>IP servers on the local network.

SAT>IP clients are at a minimum only UPnP Control Points. As such they do not announce their presence. For this reason, a client leaving the network is not detectable at this level.

The SAT>IP protocol implements RTSP and IGMP which permit to detect the presence or absence of a client (RTSP session timeout, IGMP membership queries).

3.3.5 SSDP Messages

3.3.5.1 Presence Announcement

ssdp:alive

A SAT>IP server **joining** the network, declares its presence by sending a NOTIFY message using the NTS value ssdp:alive to the SSDP multicast channel:port 239.255.255.250:1900

The SAT>IP server as a UPnP root device must multicast **three** distinct messages with the NT and USN values as specified in table 1-1 of the UPnP Device Architecture 1.1 [1].

Multicast NOTIFY ssdp:alive messages are sent regularly by all SAT>IP servers in order to signal their presence.

Method		**NOTIFY**
Request	**Protocol**	MUDP
	Headers	As specified in the UPnP Device Architecture 1.1. A request from a SAT>IP server shall include the "DEVICEID.SES.COM:" field name with the device_id value of that server.
Response		No response is sent.
Example	**Request**	**NOTIFY * HTTP/1.1** **HOST: 239.255.255.250:1900** **CACHE-CONTROL: max-age=1800** **LOCATION: http://<SatIPServer_IP_Address>/<description>.xml** **NT: urn:ses-com:device:SatIPServer:1** **NTS: ssdp:alive** **SERVER: OS/version UPnP/1.1 product/version** **USN:uuid:01234567-0123-0123-0123-0123456789ab::urn:ses-com:device:SatIPServer:1** **BOOTID.UPNP.ORG: <bootID>** **CONFIGID.UPNP.ORG: <configID>** **SEARCHPORT.UPNP.ORG: <searchPort>** **DEVICEID.SES.COM: <device_id>** **<CRLF>**

ssdp:byebye

A SAT>IP server **leaving** the network signals its departure by sending a NOTIFY method with the NTS value ssdp:byebye.

3.3.5.2 Discovery Request

The **multicast** M-SEARCH method is issued by SAT>IP clients to **discover** SAT>IP servers on the local network.

Method		**M-SEARCH**
Request	**Protocol**	MUDP
	Headers	As specified in the UPnP Device Architecture 1.1.
Response	**Line**	**HTTP/1.1 200 OK**
	Protocol	UDP
	Headers	As specified in the UPnP Device Architecture 1.1.
Example	**Request**	**M-SEARCH * HTTP/1.1** **HOST: 239.255.255.250:1900** **MAN: "ssdp:discover"** **MX: 2** **ST: urn:ses-com:device:SatIPServer:1** **USER-AGENT: OS/version UPnP/1.1 product/version** **<CRLF>**
	Response	**HTTP/1.1 200 OK** **CACHE-CONTROL: max-age=1800** **DATE: <date>** **EXT:** **LOCATION: http://<SatIPServer_IP_Address>/<description>.xml** **SERVER: OS/version UPnP/1.1 product/version** **ST: urn:ses-com:device:SatIPServer:1** **USN:uuid:01234567-0123-0123-0123-0123456789ab::urn:ses-com:device:SatIPServer:1** **BOOTID.UPNP.ORG: <bootID>** **CONFIGID.UPNP.ORG: <configID>** **SEARCHPORT.UPNP.ORG: <searchPort>** **<CRLF>**

A SAT>IP server may issue **unicast** M-SEARCH messages during the autoconfiguration phase when **defending** its own device_id.

Method		**M-SEARCH**
Request	**Protocol**	UDP
	Headers	As specified in the UPnP Device Architecture 1.1. A request from a server shall include a "DEVICEID.SES.COM:" field name with the device_id value of that server. The target server acknowledges receipt in repeating the device_id value in the response. A "DEVICEID.SES.COM:" field shall not be included in a client request.
Response	**Line**	**HTTP/1.1 200 OK**
	Protocol	UDP
	Headers	As specified in the UPnP Device Architecture 1.1. The "DEVICEID.SES.COM:" field name with a device_id value shall be returned only if the request includes the "DEVICEID.SES.COM:" field name.
Example	**Request**	**M-SEARCH * HTTP/1.1** **HOST: <Target_SatIPServer_IP_Address>** **MAN: "ssdp:discover"** **ST: urn:ses-com:device:SatIPServer:1** **USER-AGENT: OS/version UPnP/1.1 product/version** **DEVICEID.SES.COM: 3** **<CRLF>**
	Response	**HTTP/1.1 200 OK** **CACHE-CONTROL: max-age=1800** **DATE: <date>** **EXT:** **LOCATION: http://<Target_SatIPServer_IP_Address>/<description>.xml** **SERVER: OS/version UPnP/1.1 product/version** **ST: urn:ses-com:device:SatIPServer:1** **USN:uuid:01234567-0123-0123-0123-0123456789ab::urn:ses-com:device:SatIPServer:1** **BOOTID.UPNP.ORG: <bootID>** **CONFIGID.UPNP.ORG: <configID>** **SEARCHPORT.UPNP.ORG: <searchPort>** **DEVICEID.SES.COM: 3** **<CRLF>**

3.3.5.3 Device_id Header Field

SAT>IP	UPnP	SSDP Message	Protocol	Header [1]
				DEVICEID.SES.COM
Server	**Device**	**NOTIFY**	MUDP	req.
		200 OK	UDP	opt.[2]
	Control Point	**M-SEARCH**	UDP	req.
Client	**Control Point**	**M-SEARCH**	MUDP	
			UDP	

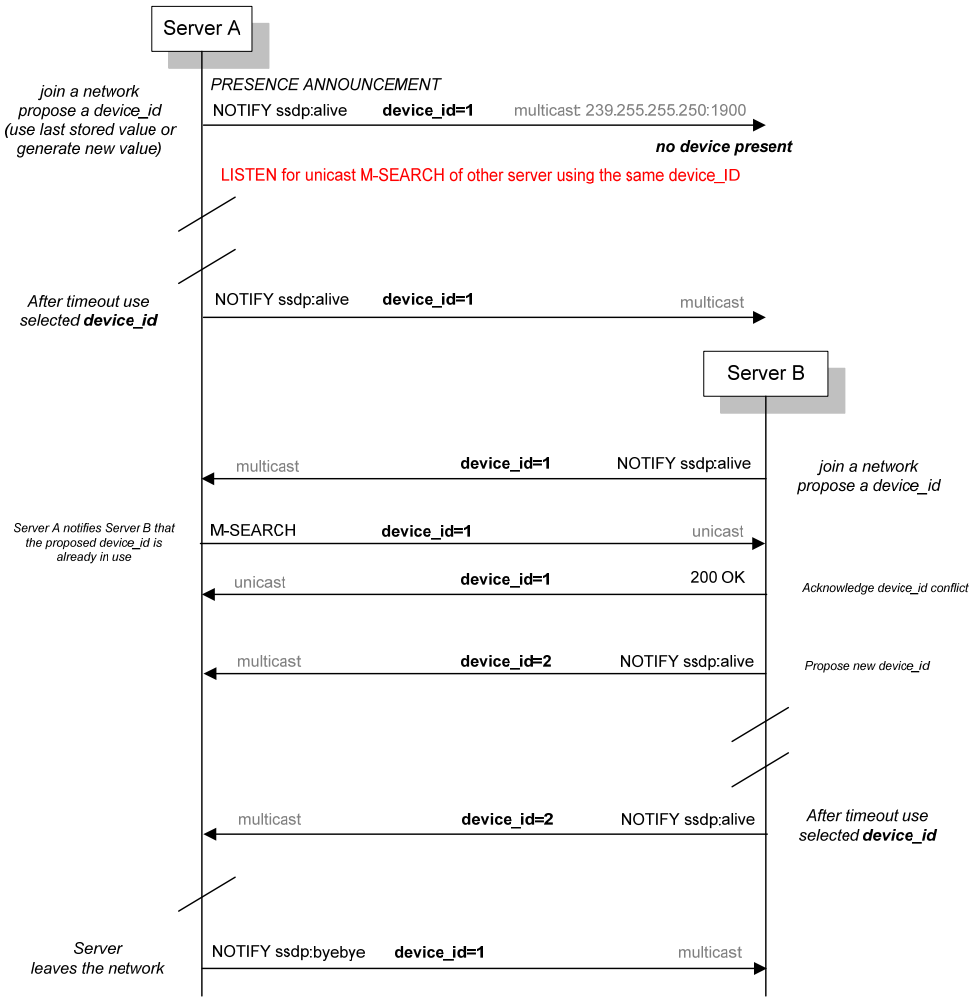
req. required field opt. optional field

[1] The header field names are case insensitive and the header field values are case sensitive. The header field order in a message is not significant.

[2] The "DEVICEID.SES.COM:" field name with the device_id value shall be returned only if the M-SEARCH request includes the "DEVICEID.SES.COM:" field name.

3.3.6 Sequence Diagram

The diagram below shows the sequencing of methods during the discovery phase:



3.4 Description

The Description phase allows SAT>IP devices to provide more information about themselves to control points on the network. The device description is done via an XML file.

The location of this file is acquired during the discovery phase: the LOCATION field in the SSDP NOTIFY message header contains the URL of the description file.

3.4.1 XML Device Description

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0" configId="configuration number">
 <specVersion>
  <major>1</major>
  <minor>1</minor>
 </specVersion>
 <device>
  <deviceType>urn:ses-com:device:SatIPServer:1</deviceType>
  <friendlyName>short user-friendly title</friendlyName>
  <manufacturer>manufacturer name</manufacturer>
  <manufacturerURL>URL to manufacturer site</manufacturerURL>
  <modelDescription>long user-friendly title</modelDescription>
  <modelName>model name</modelName>
  <modelNumber>model number</modelNumber>
  <modelURL>URL to model site</modelURL>
  <serialNumber>manufacturer's serial number</serialNumber>
  <UDN>uuid:UUID</UDN>
  <UPC>Universal Product Code</UPC>
  <iconList>
   <icon>
    <mimetype>image/format</mimetype>
    <width>horizontal pixels</width>
    <height>vertical pixels</height>
    <depth>color depth</depth>
    <url>URL to icon</url>
   </icon>
  </iconList>
  <presentationURL>URL for presentation</presentationURL>
 </device>
</root>
```

A SAT>IP device shall provide two JPEG icons and two PNG icons:

- the image width and height of the small JPEG and PNG icons must be 48 pixels,
- the image width and height of the large JPEG and PNG icons must be 120 pixels.

If the SAT>IP server integrates an HTTP server with an end-user interface the URL to the presentation root shall be provided in the <presentationURL> field.

3.5 Control

Control provides the functionality necessary for SAT>IP clients to request the delivery of media streams from SAT>IP servers. Device Control in SAT>IP can be handled through the use of RTSP or HTTP protocol mechanisms.

SAT>IP servers shall fully implement all protocol mechanisms specified in the current specification. Clients only need to implement those SAT>IP protocols important to their own proper operation.

3.5.1 RTSP

Clients can use RTSP to request the establishment of RTSP sessions with a server.

Clients and servers shall support RTSP version 1.0 as described by Appendix D of RFC 2326 [6].

The RTSP channel must be implemented using TCP. SAT>IP uses the standard RTSP server port number 554.

RTSP sessions are independent of the underlying TCP connections. An RTSP session generally spans multiple TCP connections (e.g. one TCP connection per request-response). An RTSP session may also consist of a single TCP connection (persistent mode) including several RTSP request-response pairs. RTSP sessions are always identified by an RTSP session number.

Streams requested via RTSP will be delivered using RTP/UDP. SAT>IP clients can negotiate unicast or multicast transport delivery.

3.5.1.1 Setting up a new stream

In order to set up a new TS media stream from a server through RTSP, a SAT>IP client issues an RTSP SETUP request to the server. The request contains:

- the parameters of the new TS media stream in an RTSP URI query string.
- the specification of the transport delivery method (unicast or multicast) in the header of the RTSP SETUP request.

The SETUP message starts an RTSP session with the server. The server creates a "non-playing" TS media stream identified by a streamID. This media stream is defined by the signal source, frontend selection, physical tuning and demultiplexing parameters.

The media stream is owned by the RTSP session initiator.

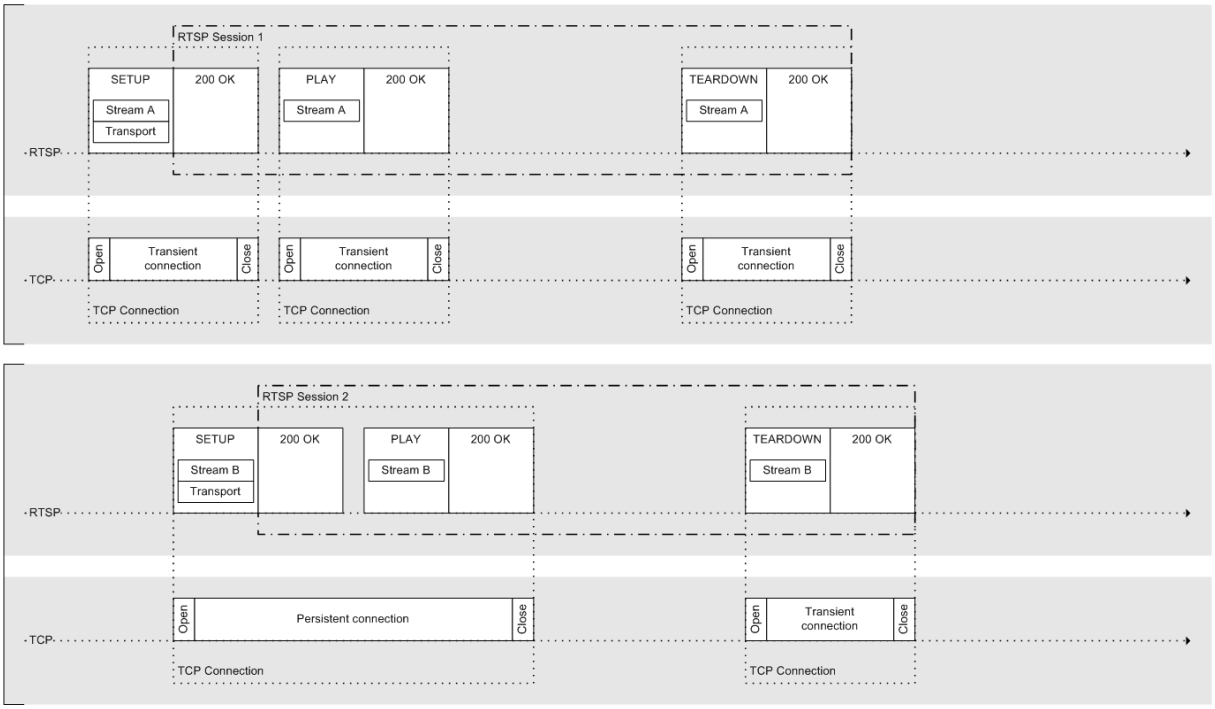
The server then responds to the client with a 200 OK message containing (in its header):

- the RTSP session number identifying the RTSP client-server session,
- an IP address (multicast or unicast) for the delivery of the media stream,
- a **streamID** identifying the TS media stream.

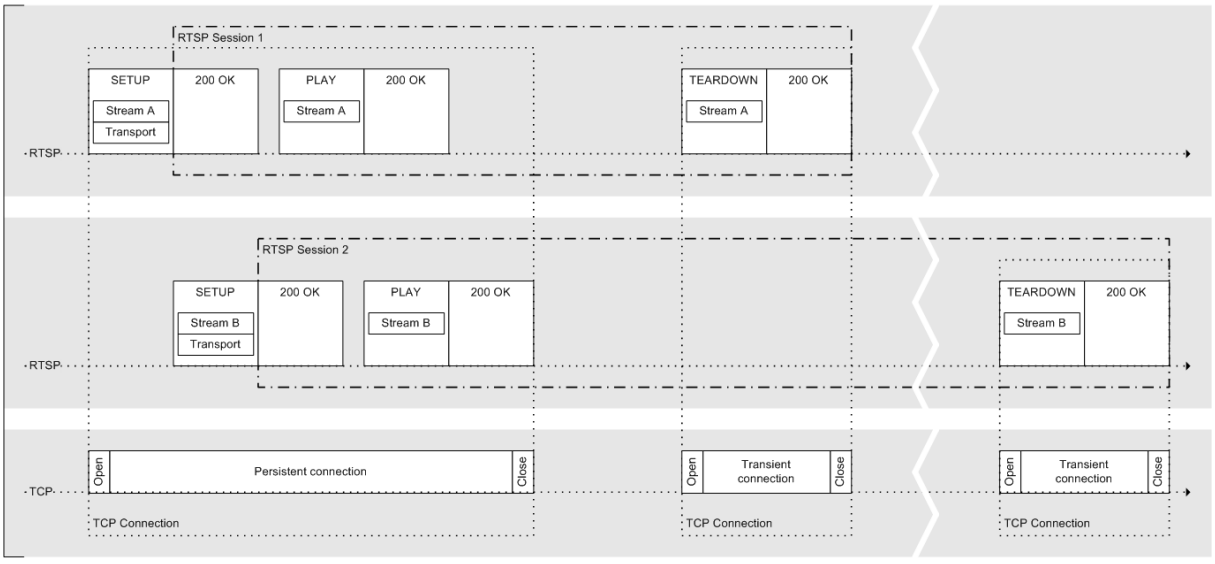
A given client can setup more than one RTSP session (e.g. to speed up channel changes).

Please note that the physical tuning may or may not intervene directly after the RTSP SETUP request message. Successful tuning is related to the availability of all the tuning parameters and the correct reception of the satellite signal. The creation of the media stream is not directly related to the actual tuning.

Example of a client with two RTSP sessions on the same SAT>IP server carried in different concurrent TCP connections:



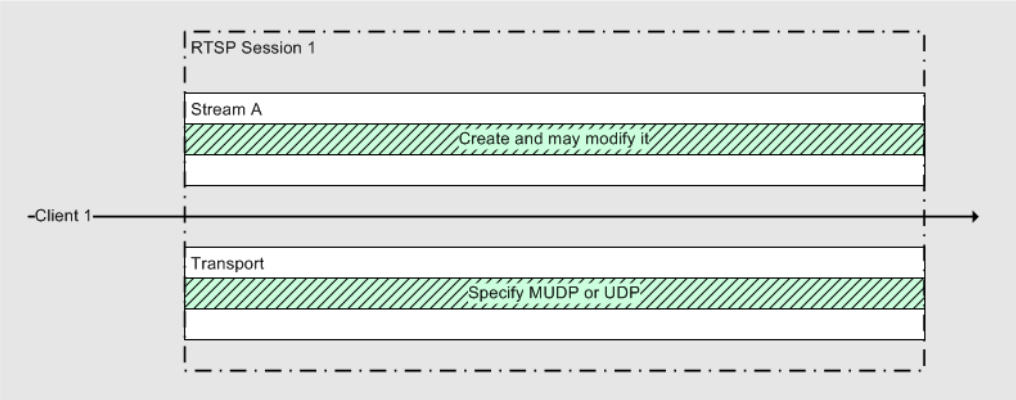
Example of a client with two RTSP sessions on the same SAT>IP server carried in one TCP connection at a given time:



Every RTSP session only contains a single TS media stream.

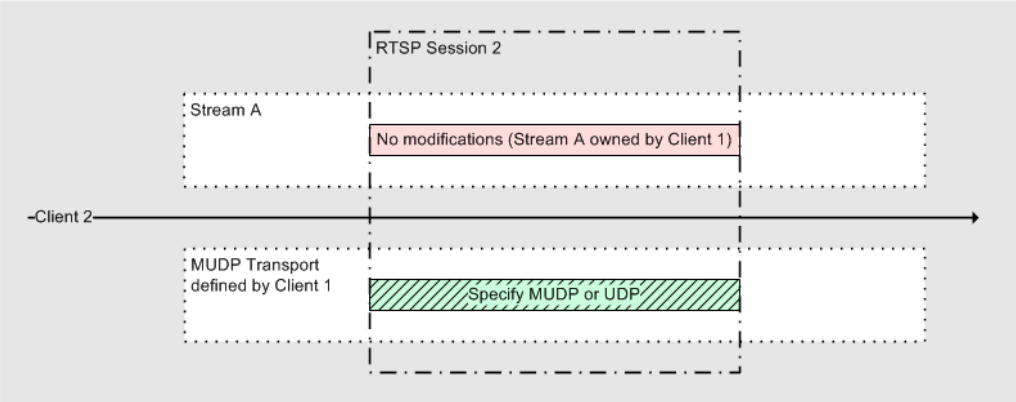
However a given media stream may be used in more than one RTSP session e.g. if a second client joins an existing media stream. Clients in these other sessions do not have the ownership of the media stream (see below).

Example of a Client 1 defining a TS media stream and its associated IP transport (unicast or multicast).

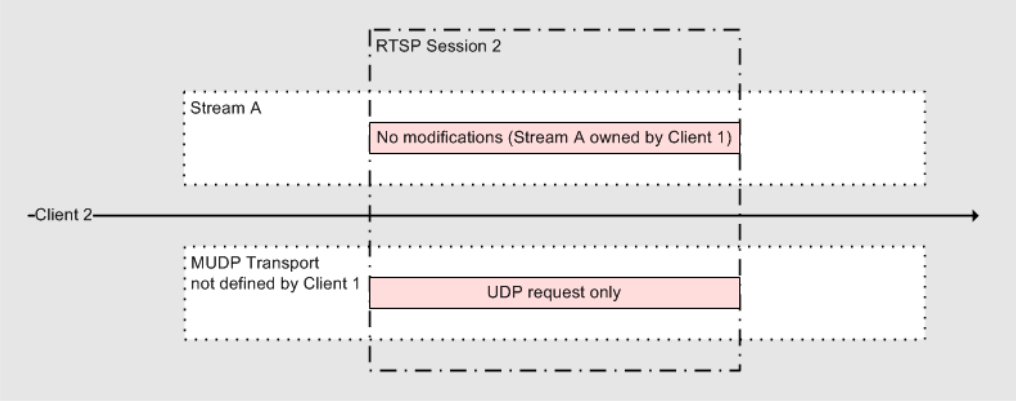


Client 2 may join the existing media stream, however cannot modify the stream itself.

If the original transport was multicast, Client 2 may join the original multicast or request an additional unicast transport for itself.



If the original transport was unicast, Client 2 may only request an additional unicast (not multicast) for itself.



3.5.1.2 Starting the playout of a media stream

In order to start the actual play-out of the TS media stream in **unicast**, the client sends:

- an RTSP PLAY message with the required streamID (or the URI query string).

In order to start the play-out of the TS media stream in **multicast**, the client sends:

- an RTSP PLAY message with the required streamID (or the URI query string),
- an IGMPv3 membership report [7] to the multicast group.

The server reacts on **either** message by playing/forwarding the stream.

The RTSP PLAY message can only be invoked by clients within an RTSP session (i.e. after having invoked the SETUP request which creates a new session).

3.5.1.3 Stopping the playout of a media stream

In order to stop the playout of a **unicast** TS media stream a client issues:

- an RTSP TEARDOWN message.

In order to stop the playout of a **multicast** TS media stream the client issues:

- an RTSP TEARDOWN message.
- an IGMP leave message.

An out-of-session client may leave a stream by only issuing an IGMP leave message.

The server stops the multicast TS media stream only when receiving the TEARDOWN message of the session owner. A server has to be able to handle the concurrent access to TS media streams from multiple clients.

An RTSP TEARDOWN message ends the RTSP session.

3.5.1.4 Modifying a media stream

A TS media stream may only be modified by its owner. The owner generally invokes the RTSP SETUP and/or PLAY methods for modifying the media stream.

The parameters of the new TS media stream are passed to the server in the RTSP URI query string.

A change in parameters shall not interrupt the playing stream and shall not lead to any packet loss within the stream.

3.5.1.5 Joining an existing stream

Clients may join an already defined stream (existing streamID) owned by another client:

- by setting up a new session themselves (invoking SETUP and PLAY on the streamID) or
- by purely joining an existing stream through an IGMPv3 membership report.

Please note that the joining client will not get the ownership of the media stream.

In this case, clients risk loosing access to their stream if the owner of the media stream does a TEARDOWN. Therefore this mainly applies in scenarios where streams have been statically joined and are therefore always present.

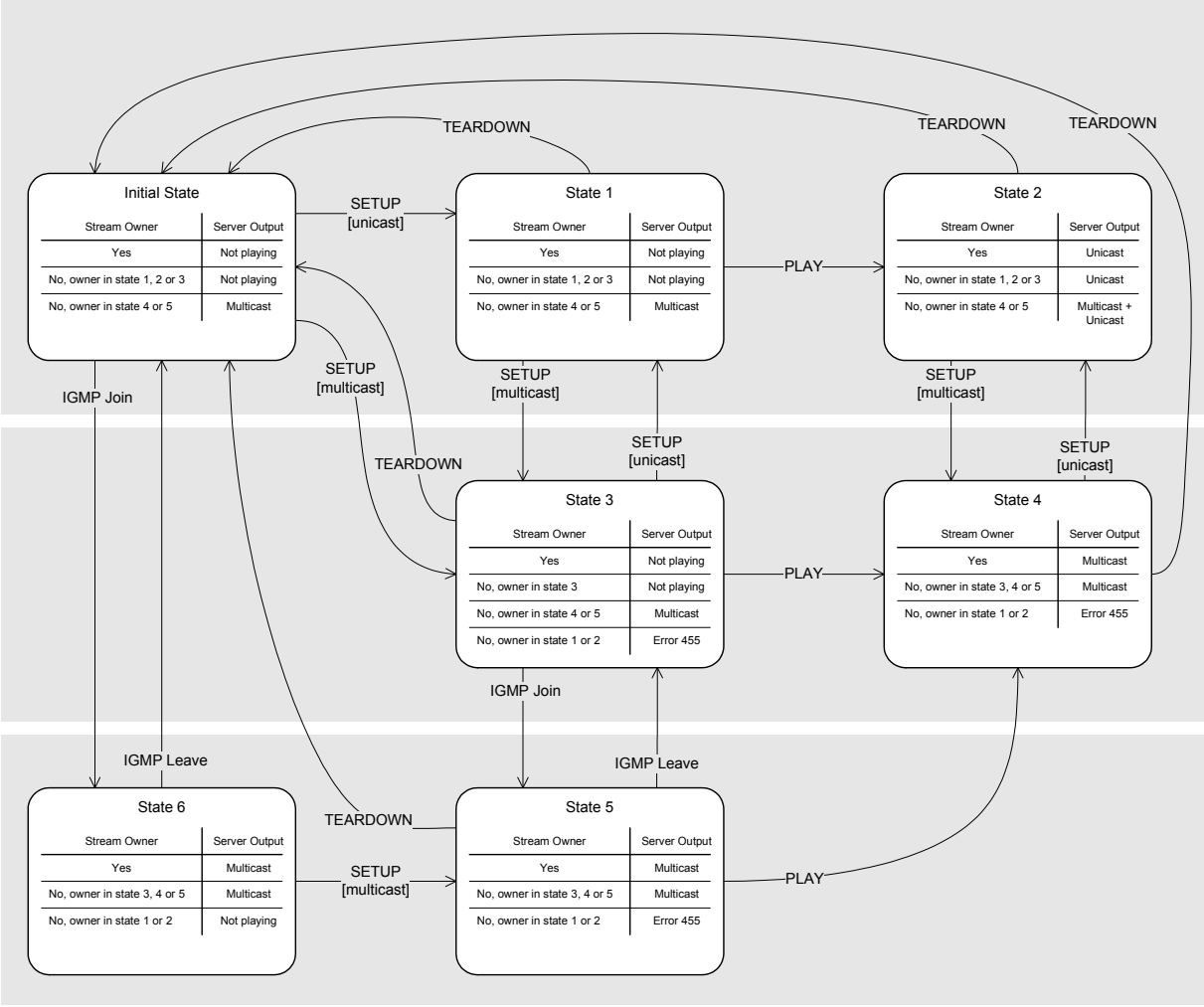
3.5.1.6 Server Stream Output

The SAT>IP server output depends on the server state (resulting from the sequence of RTSP and/or IGMP methods invoked by a client) and the state of the media stream owner.

If the client has the ownership of a media stream, the server output is given in line 1.

If the client does not have the ownership of the media stream, the server output also depends on the actual state of the media stream owner (lines 2, 3 or 4).

Server Stream Output State Machine



Any RTSP/IGMP method not shown in the diagram above will not modify the server output state.

3.5.1.7 Listing available media streams

A client can invoke the DESCRIBE request **anytime** (in or out of an RTSP session) to enquire about one particular media stream or all existing media streams configured on the SAT>IP server.

In response the SAT>IP server sends an SDP message providing the details of the media stream(s) including their multicast addresses.

3.5.2 Uniform Resource Identifier URI

The RTSP request consists of an RTSP method followed by an RTSP URI.

The URI is composed of the IP address followed by a stream identifier and an optional query.

RTSP_URI = "rtsp://" ip_address "/" ["stream="streamID] ["?"query]

URI Examples:

```
rtsp://192.168.128.1/?src=1&fe=1&freq=12402&pol=v&msys=dvbs&mtype=qpsk&sr=27500&fec=34&pids
=0,16,50,104,166,1707

rtsp://192.168.128.1/stream=0?freq=12402&pol=v&msys=dvbs&sr=27500&fec=34&pids=0,16,104,166,
1707

rtsp://192.168.128.1/stream=1?addpids=17,51&delpids=166,1707
```

3.5.3 Query

3.5.3.1 Description

The query specifies the signal source selection, frontend selection, physical tuning, and demultiplexing parameters of a TS media stream.

Signal Source Selection

Clients specify the source from which they intend to receive their signals. The source parameter communicates to the frontend the selection of a satellite orbital position.

The number of physical inputs corresponding to a given source may vary according to the actual LNB types supported e.g. Universal LNBs (Single, Quad, etc.) or Quatro, Single Cable. Please note that multiple LNBs may point to the same or different orbital positions.

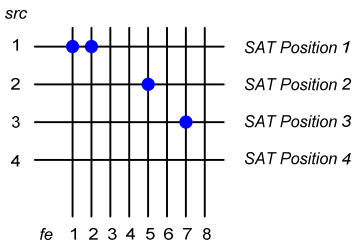
Every source may feed one or more frontends.

Frontend Selection

Clients may be configured statically by specifying one or more dedicated frontend(s) to use (fe=x). (static configuration). A static configuration guarantees that clients always get access to their tuning and demodulation resources. At the same time the static configuration limits the number of clients which may be active simultaneously.

If clients do not specify a particular frontend, the SAT>IP server will carry out the selection of the frontend automatically.

Example showing the source and frontend selection matrix:



Physical Tuning

Standard DVB-S/S2 tuning parameters are passed to the SAT>IP server frontend in order to tune to a given transponder.

Demultiplexing

The demux is informed about all the PIDs to be filtered through the "pids" comma separated values (CSV) list. Filtered PIDs may be changed, added or removed during normal operation via additional "pids", "addpids" or "delpids" queries.

Dynamic addition or removal of PIDs shall not interrupt continuous playback from the in-play demux and shall not lead to packet loss for any remaining PID values.

Clients are responsible for ignoring PIDs that they may have no interest in and which are present in the TS media stream.

3.5.3.2 Syntax

Queries are composed of a series of attribute value pairs separated by the "&" (ampersand) sign. The attribute value pairs are each separated by the "=" (equal) sign.

<attribute1>=<value1>&<attribute2>=<value2>&<attribute3>=<value3>...

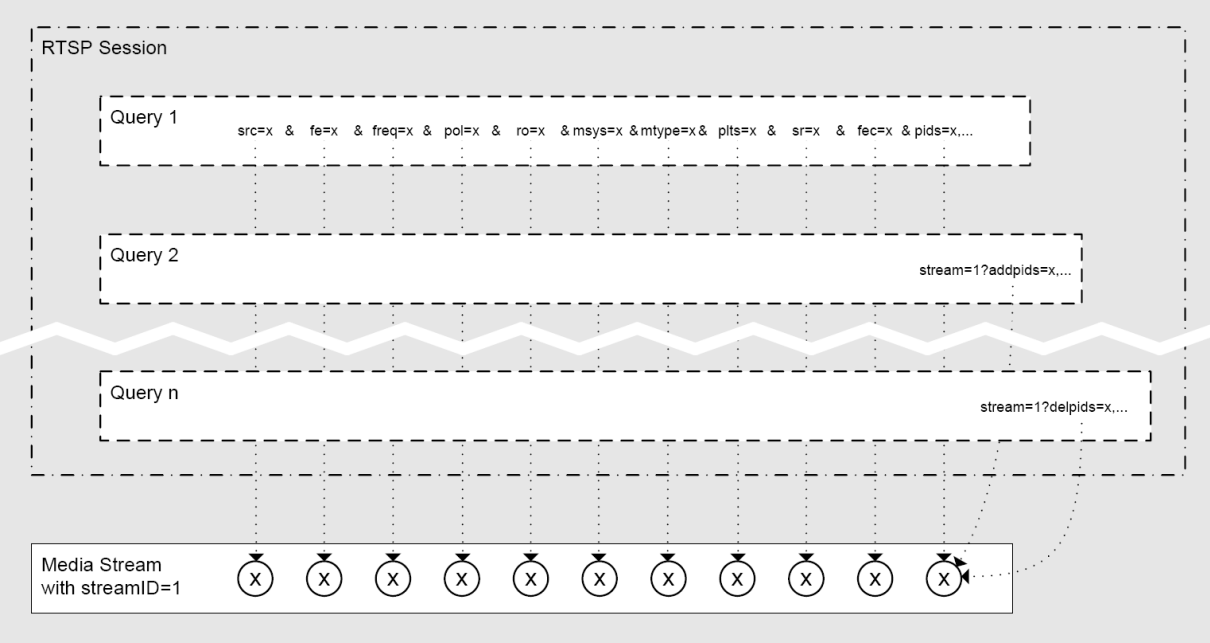
Query attribute value pairs may appear in any order. The parser on the SAT>IP server shall be sufficiently robust to correctly delineate values and identify the different parameters.

Name	Attribute	Value		Example
		Name	Range	
Signal source	src	srcID	Numerical value between 1 and 255. Default value is "1".	`src=1`
Frontend identifier	fe	feID	Numerical value between 1 and 65535.	`fe=1`
Frequency	freq	frequency	Transponder frequency expressed in MHz as fixed point type or integer value.	`freq=11361.75` `freq=11720`
Polarisation	pol	polarisation	Set to one of the following values: "h" horizontal linear, "v" vertical linear, "l" circular left, "r" circular right.	`pol=h`
Roll-Off	ro	roll_off	Set to one of the following values: "0.35", "0.25", "0.20". *Parameter not required for DVB-S.*	`ro=0.35`
Modulation system	msys	system	Set to one of the following values: "dvbs", "dvbs2".	`msys=dvbs2`
Modulation type	mtype	type	Set to one of the following values: "qpsk", "8psk". *Parameter not required for DVB-S.*	`mtype=8psk`
Pilot tones	plts	pilots	Set to one of the following values: "on", "off". *Parameter not required for DVB-S.*	`plts=off`
Symbol rate	sr	symbol_rate	Value in kSymb/s.	`sr=22000`
FEC inner	fec	fec_inner	Set to one of the following values: "12", "23", "34", "56", "78", "89", "35", "45", "910".	`fec=23`
List of PIDs	pids		CSV list of PIDs: (Num. values betw. 0 and 8191) for spts streams, "all" for mpts streams, "none" for no demux output.	`pids=0,16,201,302`
	addpids [1]		Opens new PID filters on the demux for streaming on the network. CSV list of PIDs.	`addpids=307,309`
	delpids [1]		Removes PID filters from the demux. CSV list of PIDs.	`delpids=201,302`
Multicast address	mcast [2]		Dot-decimal notation IP address.	`mcast=224.100.20.1`

[1] The "addpids" or "delpids" parameters shall not be used in combination with the "pids" parameter in the same RTSP query. The "addpids" and "delpids" parameters may be combined in the same RTSP query.

[2] Please note that in normal operation, multicast addresses are assigned automatically by the SAT>IP server. Usage of the mcast attribute is reserved for testing purposes only.

A media stream may be defined by a single RTSP query listing all the required parameters, or by a succession of RTSP methods with the succession of queries leading to its full definition.



The “addpids” or “delpids” parameters shall not be used in combination with the “pids” parameter in the same RTSP query.

The “addpids” and “delpids” parameters may be combined in the same RTSP query.

Query Examples:

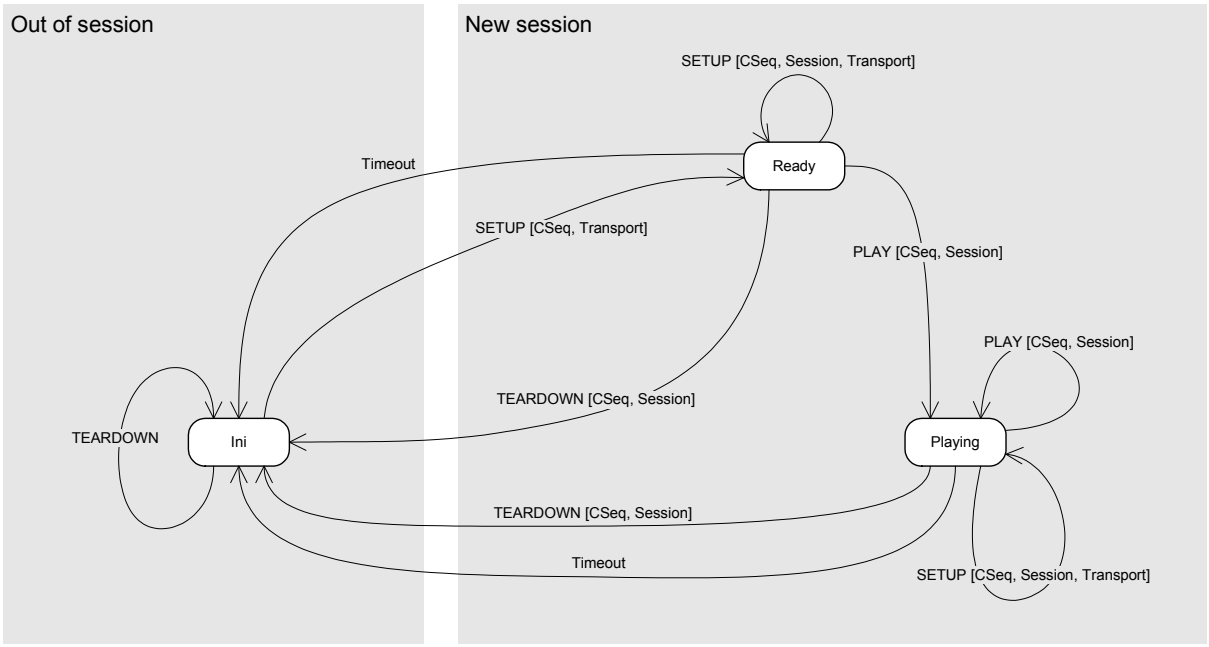
```
DVB-S

rtsp://192.168.128.5/?src=1&fe=1&freq=12402&pol=v&msys=dvbs&sr=27500&fec=34&pids=0,16,50,10
4,166,1707

DVB-S2

rtsp://192.168.128.5/?src=1&fe=1&freq=12720&pol=v&msys=dvbs2&mtype=8psk&ro=0.35&plts=on&sr=
30000&fec=910&pids=0,16,50,104,166,1707
```

3.5.4 RTSP Methods

Clients and servers shall support the following RTSP methods: OPTIONS, DESCRIBE, SETUP, PLAY and TEARDOWN.

Methods	RTSP URI	Usage
SETUP	rtsp://<ip_address>/?src=<srcID>&fe=<feID>&freq=<frequency>&…	Creating a new media stream
	rtsp://<ip_address>/stream=<streamID>	Joining an existing media stream
	rtsp://<ip_address>/stream=<streamID>?src=<srcID>&…	Modifying a media stream
PLAY	rtsp://<ip_address>/?src=<srcID>&fe=<feID>&freq=<frequency>&…	Playing with or without modifying a media stream
	rtsp://<ip_address>/stream=<streamID>	Playing a media stream
	rtsp://<ip_address>/stream=<streamID>?src=<srcID>&…	Playing with or without modifying a media stream
TEARDOWN	rtsp://<ip_address>/stream=<streamID>	Leaving of the session
OPTIONS	rtsp://<ip_address>/	Maintaining a session alive
DESCRIBE	rtsp://<ip_address>/	Description of all the media streams configured
	rtsp://<ip_address>/stream=<streamID>	Description of a particular media stream

RTSP State machine



RTSP Session Timeout

Once an RTSP session has been created (e.g. to start an RTP stream), the client needs to maintain the RTSP session by sending an RTSP request before the end of each timeout period (60 seconds by default). Typically clients send an RTSP OPTIONS method in order to keep the RTSP session alive. The server therefore knows about the presence or absence of each RTSP client.

3.5.4.1 Setup

The SETUP method is generally invoked by the client to create a new RTSP session and to setup a new media stream. It specifies the transport mechanism to be used for the streamed media. This includes information about whether the media transport shall use UDP **unicast** or UDP **multicast**.

In return to this method the SAT>IP server communicates the RTSP session number, the actual IP address used for media delivery and a **streamID** identifying the TS media stream.

Every RTSP session between a server and a client is identified by a randomly generated session number. This number is communicated by the server in its 200 OK response to the SETUP method.

The server allocates an IP multicast or unicast address for the delivery of the media stream based on the SETUP "Transport" header field request.

The streamID in SAT>IP uniquely identifies a TS media stream object. The streamID is a 16-bit numeric value. It is generated by the server and communicated to the client in the 200 OK response. The client should use the streamID in order to refer to streams which have been previously setup.

Method		**SETUP**
RTSP_URI		rtsp://<ip_address>/?src=<srcID>&fe=<feID>&freq=<frequency>&… rtsp://<ip_address>/stream=<streamID>?src=<srcID>&…
Request	**Headers**	CSeq, Session (1), Transport (2)
	Body	n/a
Response	**Headers**	CSeq, Session (3), Transport, com.ses.streamID
	Body	n/a
	Status Codes	400, 403, 404, 414, 454, 461, 500, 503, 505, 551
Remarks		1 This header field is only required if the client is in a session. 2 Only a single transport mode is allowed (no preference list). 3 The session timeout time value is specified by the "timeout" parameter in the "Session:" header field of the response. The default timeout value in RTSP is 60 seconds.
Example	**Unicast Transport Delivery**	
	Request	**SETUP rtsp://192.168.128.5/?src=1&fe=1&freq=12402&pol=v&msys=dvbs&sr=27500&fec=34&pids=0,16,50,104,166,1707 RTSP/1.0** **CSeq: 1** **Transport: RTP/AVP;unicast;client_port=1400-1401** **<CRLF>**
	Response	**RTSP/1.0 200 OK** **CSeq: 1** **Session: 12345678;timeout=60** **Transport: RTP/AVP;unicast;client_port=1400-1401;source=192.168.128.5;server_port=1528-1529** **com.ses.streamID: 1** **<CRLF>**
Example	**Multicast Transport Delivery**	
	Request	**SETUP rtsp://192.168.128.5/?src=1&fe=1&freq=12402&pol=v&msys=dvbs&sr=27500&fec=34&pids=0,16,50,104,166,1707 RTSP/1.0** **CSeq: 1** **Transport: RTP/AVP;multicast;port=5004-5005** **<CRLF>**
	Response	**RTSP/1.0 200 OK** **CSeq: 1** **Session: 12345678;timeout=60** **Transport: RTP/AVP;multicast;destination=239.0.0.9;port=5004-5005;ttl=5;source=192.168.128.5** **com.ses.streamID: 1** **<CRLF>**

Please note that an RTP delivered media stream is originated and received on an even port number and the associated RTCP announcement uses the next higher odd port number. The port names differ between UDP unicast transport and UDP multicast transport:

For unicast: client_port= <client RTP port> - <client RTCP port>
server_port= <server RTP port> - <server RTCP port>

For multicast: port= <RTP port> - <RTCP port>

3.5.4.2 Play

The PLAY method starts the delivery of the TS media stream to the SAT>IP client.

Method		**PLAY**
RTSP_URI		rtsp://<ip_address>/stream=<streamID>
Request	**Headers**	CSeq, Session
	Body	n/a
Response	**Headers**	CSeq, Session, RTP-Info
	Body	n/a
	Status Codes	400, 403, 404, 408, 453, 454, 500, 505, 551
Example	**Request**	**PLAY rtsp://192.168.128.5/stream=1 RTSP/1.0** **CSeq: 2** **Session: 12345678** **<CRLF>**
	Response	**RTSP/1.0 200 OK** **CSeq: 2** **Session: 12345678** **RTP-Info: url=rtsp://192.168.128.5/stream=1** **<CRLF>**

3.5.4.3 Teardown

The TEARDOWN method is used to terminate the RTSP session.

Method		**TEARDOWN**
RTSP_URI		rtsp://<ip_address>/stream=<streamID>
Request	**Headers**	CSeq, Session
	Body	n/a
Response	**Headers**	CSeq, Session
	Body	n/a
	Status Codes	400, 403, 404, 408, 454, 500, 505, 551
Example	**Request**	**TEARDOWN rtsp://192.168.128.5/stream=1 RTSP/1.0** **CSeq: 3** **Session: 12345678** **<CRLF>**
	Response	**RTSP/1.0 200 OK** **CSeq: 3** **Session: 12345678** **<CRLF>**

3.5.4.4 Options

The OPTIONS method may be invoked by clients to query servers on supported RTSP methods. Clients also invoke the OPTIONS method in order to keep the RTSP session alive within the timeout period.

Method		**OPTIONS**
RSTP_URI		rtsp://<ip_address>/
Request	**Headers**	CSeq, Session [(1)]
	Body	n/a
Response	**Headers**	CSeq, Session [(1)], Public
	Body	n/a
	Status Codes	400, 403, 454, 500, 505, 551
Remarks		This method may be issued at any time. [1] This header field is only required if the client is in a session (e.g. for a keep-alive).
Example	**Request**	**OPTIONS rtsp://192.168.128.5/ RTSP/1.0** **CSeq: 4** **<CRLF>**
	Response	**RTSP/1.0 200 OK** **CSeq: 4** **Public: OPTIONS, DESCRIBE, SETUP, PLAY, TEARDOWN** **<CRLF>**

3.5.4.5 Describe

SAT>IP also provides the possibility for clients to enquire servers about the TS media streams currently configured. SAT>IP servers provide an SDP (Session Description Protocol) [10] formatted description file in response to an RTSP DESCRIBE request.

The RTSP request may be done on the IP address of the SAT>IP server.

RTSP_URI = "rtsp://" ip_address "/"

In this case the SDP describes all TS streams available from the server.

The RTSP request may also be done on a particular streamID.

RTSP_URI = "rtsp://" ip_address "/" "stream="streamID

In this case the SDP describes only that particular stream.

The SDP implementation is based on RFC 4566 [10] complemented with the following attribute syntax:

```
Session Level:

s=SatIPServer:1 <frontends>

Media level:

a=control:stream=<streamID>

a=fmtp:33 ver=<major>.<minor>;src=<srcID>;tuner=<feID>,<level>,<lock>,<quality>,<frequency>
,<polarisation>,<system>,<type>,<pilots>,<roll_off>,<symbol_rate>,<fec_inner>;pids=<pid0>,…
,<pidn>
```

Most label fields have been specified already as part of the query syntax. Additional fields are specified below:

Name	Value		Example
	Name	**Range**	
No. of frontends	frontends	Contains the total number of frontends available on the server	`4`
Major version	major	Contains the major version number of the fmtp string syntax	`1.0`
Minor version	minor	Contains the minor version number of the fmtp string syntax	
Signal level	level	Numerical value between 0 and 255 No signal corresponds to 0	`240`
Frontend lock	lock	Set to one of the following values: "0" the frontend is not locked "1" the frontend is locked	`1`
Signal quality	quality	Numerical value between 0 and 15 Lowest value corresponds to highest BER	`7`

Method		DESCRIBE
RTSP_URI		rtsp://<ip_address>/ rtsp://<ip_address>/stream=<streamID>
Request	Headers	CSeq, Session [1], Accept
	Body	n/a
Response	Headers	CSeq, Session [1], Content-Type: application/sdp, Content-Base, Content-Length
	Body	application/sdp
	Status Codes	400, 404, 406, 500, 505, 551
Remarks		This method may be issued at any time. [1] This header field is only required if the client is in a session.
Example	Request	**DESCRIBE rtsp://192.168.128.5/** **CSeq: 5** **Accept: application/sdp** **<CRLF>**
	Response	**RTSP/1.0 200 OK** **CSeq: 5** **Content-Type: application/sdp** **Content-Base: rtsp://192.168.128.5/** **Content-Length: 724** **<CRLF>** **v=0** **o=- 5678901234 7890123456 IN IP4 192.168.128.5** **s=SatIPServer:1 4** **t=0 0** **m=video 5004 RTP/AVP 33** **c=IN IP4 239.0.0.8/5** **a=control:stream=0** **a=fmtp:33 ver=1.0;src=1;tuner=1,240,1,7,12402,v,dvbs,,,,27500,34;pids=0,16,56,112,168** **,1709** **a=inactive** **m=video 5006 RTP/AVP 33** **c=IN IP4 239.0.0.9/5** **a=control:stream=1** **a=fmtp:33 ver=1.0;src=1;tuner=1,240,1,7,12402,v,dvbs,,,,27500,34;pids=0,16,50,104,166** **,1707** **a=sendonly** **m=video 5008 RTP/AVP 33** **c=IN IP4 0.0.0.0** **a=control:stream=2** **a=fmtp:33 ver=1.0;src=1;tuner=1,240,1,7,12402,v,dvbs,,,,27500,34;pids=all** **a=inactive** **m=video 5010 RTP/AVP 33** **c=IN IP4 239.0.0.11/5** **a=control:stream=3** **a=fmtp:33 ver=1.0;src=2;tuner=2,221,1,6,11758,h,dvbs2,8psk,off,25,27500,56;pids=all** **a=sendonly**

The example above shows an SDP listing all the streams currently available from the SAT>IP server.

TS media streams in non-playing mode are tagged with the a=inactive attribute. Media streams in playing mode are tagged with a=sendonly.

Multicast stream addresses are signalled by the "c=" connection field. For unicast streams the address is set to 0.0.0.0.

3.5.5 Status Code Definitions

Status code messages are sent in reply to RTSP requests. The following table lists the status code messages that are used in the SAT>IP protocol.

Status code	Reason Phrase
100	Continue
200	OK
400	Bad Request
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
408	Request Timeout
414	Request-URI Too Long
453	Not Enough Bandwidth
454	Session Not Found
455	Method Not Valid in This State
461	Unsupported Transport
500	Internal Server Error
501	Not Implemented
503	Service Unavailable
505	RTSP Version Not Supported
551	Option Not Supported

3.5.5.1 Client Error Response Messages (4xx)

Status code	**400** – Bad Request	
Description	The request could not be understood by the server due to a malformed syntax. Returned when passing an undefined attribute in a query, missing a character, inconsistent request (duplicate attributes), etc.	
Response	**Headers**	CSeq, Content-Type: text/parameters, Content-Length
	Body	The message body of the response may contain the "**Check-Syntax:**" parameter followed by the malformed syntax.
Example	**Request**	**PLAY rtsp://192.168.128.5/strem=1 RTSP/1.0** **CSeq: 6** **Session: 12345678** **<CRLF>**
	Response	**RTSP/1.0 400 Bad Request** **CSeq: 6** **Content-Type: text/parameters** **Content-Length: 19** **<CRLF>** **Check-Syntax: strem**

Status code	**403** – Forbidden	
Description	The server understood the request, but is refusing to fulfil it. Returned when passing an attribute value not understood by the server in a query, or an out-of-range value.	
Response	**Headers**	CSeq, Content-Type: text/parameters, Content-Length
	Body	The message body of the response may contain the "**Out-of-Range:**" parameter followed by a space-separated list of the attributes that are not understood: "src" "fe" "freq" "pol" "msys" "mtype" "plts" "ro" "sr" "fec" "pids" "addpids" "delpids" "mcast"
Example	**Request**	**SETUP rtsp://192.168.128.5/?src=1&fe=1&freq=22402&pol=v&msys=dvbs&sr=27500&fec=34** **&pids=0,16,50,104,166,1707,8192 RTSP/1.0** **Cseq: 7** **<CRLF>**
	Response	**RTSP/1.0 403 Forbidden** **Cseq: 7** **Content-Type: text/parameters** **Content-Length: 23** **<CRLF>** **Out-of-Range: freq pids**

Status code	**404** – Not Found	
Description	The server has not found anything matching the Request-URI. Returned when requesting a stream with a streamID that does not exist.	
Response	**Headers**	CSeq
	Body	n/a
Example	**Request**	**PLAY rtsp://192.168.128.5/stream=7 RTSP/1.0** **CSeq: 8** **Session: 12345678** **<CRLF>**
	Response	**RTSP/1.0 404 Not Found** **CSeq: 8** **<CRLF>**

Status code	**405** – Method Not Allowed	
Description	The method specified in the request is not allowed for the resource identified by the Request-URI. Returned when applying a SETUP, PLAY or TEARDOWN method on an RTSP URI identifying the server.	
Response	**Headers**	CSeq, Allow
	Body	n/a
Example	**Request**	**PLAY rtsp://192.168.128.5/ RTSP/1.0** **CSeq: 9** **Session: 12345678** **<CRLF>**
	Response	**RTSP/1.0 405 Method Not Allowed** **CSeq: 9** **Allow: OPTIONS, DESCRIBE** **<CRLF>**

Status code	**406** – Not Acceptable	
Description	The resource identified by the request is only capable of generating response message bodies which have content characteristics not acceptable according to the accept headers sent in the request. Issuing a DESCRIBE request with an accept header different from "application/sdp".	
Response	**Headers**	CSeq
	Body	n/a
Example	**Request**	**DESCRIBE rtsp://192.168.128.5/ RTSP/1.0** **Cseq: 10** **Accept: text/plain** **<CRLF>**
	Response	**RTSP/1.0 406 Not Acceptable** **Cseq: 10** **<CRLF>**

Status code	**408** – Request Timeout	
Description	The client did not produce a request within the time that the server was prepared to wait. The client may repeat the request without modifications at any later time. E.g. issuing a PLAY request after the communication link had been idle for a period of time. The time interval has exceeded the value specified by the "timeout" parameter in the "Session:" header field of a SETUP response.	
Response	**Headers**	CSeq
	Body	n/a
Example	**Request**	**PLAY rtsp://192.168.128.5/stream=1 RTSP/1.0** **CSeq: 11** **Session: 12345678** **<CRLF>**
	Response	**RTSP/1.0 408 Request Timeout** **CSeq: 11** **<CRLF>**

Status code	**414** – Request-URI Too Long	
Description	The server is refusing to service the request because the Request-URI is longer than the server is willing to interpret. The RTSP protocol does not place any limit on the length of a URI. Servers should be able to handle URIs of unbounded length.	
Response	**Headers**	CSeq
	Body	n/a
Example	**Request**	**SETUP rtsp://192.168.128.5/?src=1&fe=1&freq=12402&pol=v&msys=dvbs&sr=27500&fec=34** **&pids=0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25** **RTSP/1.0** **Cseq: 12** **<CRLF>**
	Response	**RTSP/1.0 414 Request-URI Too Long** **Cseq: 12** **<CRLF>**

Status code	**453** – Not Enough Bandwidth	
Description	The request was refused because there is insufficient bandwidth on the in-home LAN. Returned when clients are requesting more streams than the network can carry.	
Response	**Headers**	CSeq
	Body	n/a
Example	**Request**	**PLAY rtsp://192.168.128.5/stream=25 RTSP/1.0** **CSeq:13** **Session: 12345678** **<CRLF>**
	Response	**RTSP/1.0 453 Not Enough Bandwidth** **CSeq: 13** **<CRLF>**

Status code	**454** – Session Not Found	
Description	The RTSP session identifier value in the "Session:" header field of the request is missing, invalid, or has timed out. Returned when issuing the wrong session identifier value in a request.	
Response	**Headers**	CSeq
	Body	n/a
Example	**Request**	**PLAY rtsp://192.168.128.5/stream=1 RTSP/1.0** **Cseq: 14** **Session: 0** **<CRLF>**
	Response	**RTSP/1.0 454 Session Not Found** **Cseq: 14** **<CRLF>**

Status code	**455** – Method Not Valid in This State	
Description	The client or server cannot process this request in its current state. Returned e.g. when trying to change transport parameters while the server is in the play state (e.g. change of port values, etc.).	
Response	**Headers**	CSeq
	Body	n/a
Example	**Request**	**SETUP rtsp://192.168.128.5/stream=1 RTSP/1.0** **CSeq: 15** **Transport: RTP/AVP;multicast;port=1402-1403** **<CRLF>**
	Response	**RTSP/1.0 455 Method Not Valid in This State** **CSeq: 15** **<CRLF>**

Status code	**461** – Unsupported Transport	
Description	The "Transport:" header field of the request did not contain a supported transport specification. Returned e.g. when issuing a profile that is different from "RTP/AVP".	
Response	**Headers**	CSeq
	Body	n/a
Example	**Request**	**SETUP rtsp://192.168.128.5/?src=1&fe=1&freq=12402&pol=v&msys=dvbs&sr=27500&fec=34** **&pids=0,16,50,104,166,1707 RTSP/1.0** **CSeq: 16** **Transport: RTP/SAVP;multicast;port=1400-1401** **<CRLF>**
	Response	**RTSP/1.0 461 Unsupported Transport** **CSeq: 16** **<CRLF>**

3.5.5.2 Server Error Response Messages (5xx)

Status code	**500** – Internal Server Error	
Description	The server encountered an error condition preventing it to fulfil the request. Returned when the server is not functioning correctly due to a hardware failure or a software bug or anything else that can go wrong.	
Response	**Headers**	CSeq
	Body	n/a
Example	**Request**	**DESCRIBE rtsp://192.168.128.5/** **CSeq: 17** **Accept: application/sdp** **<CRLF>**
	Response	**RTSP/1.0 500 Internal Server Error** **CSeq: 17** **<CRLF>**

Status code	**501** – Not Implemented	
Description	The server does not support the functionality required to fulfil the request. Issuing a request that is not implemented (e.g. PAUSE).	
Response	**Headers**	CSeq, Public
Example	**Request**	**PAUSE rtsp://192.168.128.5/stream=1 RTSP/1.0** **CSeq: 18** **Session: 12345678** **<CRLF>**
	Response	**RTSP/1.0 501 Not Implemented** **CSeq: 18** **Public: OPTIONS, DESCRIBE, SETUP, PLAY, TEARDOWN** **<CRLF>**

Status code	**503** – Service Unavailable	
Description	The server is currently unable to handle the request due to a temporary overloading or maintenance of the server. Returned when reaching the maximum number of hardware and software resources, the maximum number of sessions.	
Response	**Headers**	CSeq
	Body	The message body of the response may contain the "**No-More:**" parameter followed by a space-separated list of the missing ressources: “sessions” "frontends" "pids"
Example	**Request**	**SETUP rtsp://192.168.128.5/?src=2&freq=12402&pol=v&msys=dvbs&sr=27500&fec=34&pids=0,16,50,104,166,1707 RTSP/1.0** **CSeq: 19** **Transport: RTP/AVP;multicast;port=1400-1401** **<CRLF>**
	Response	**RTSP/1.0 503 Service Unavailable** **CSeq: 19** **<CRLF>** **No-More: frontends**

Status code	**505** – RTSP Version Not Supported	
Description	The server does not support the RTSP protocol version that was used in the request message.	
Response	**Headers**	CSeq
	Body	n/a
Example	**Request**	**OPTIONS rtsp://192.168.128.5/ RTSP/2.0** **CSeq: 20** **<CRLF>**
	Response	**RTSP/1.0 505 RTSP Version Not Supported** **CSeq: 20** **<CRLF>**

Status code	**551** – Option Not Supported	
Description	A feature-tag given in the "Require:" header field of the request was not supported. Issuing a request with a "Require:" header field.	
Response	**Headers**	CSeq, Unsupported
	Body	n/a
Example	**Request**	**PLAY rtsp://192.168.128.5/stream=1 RTSP/1.0** **CSeq: 21** **Require: specific-feature** **Specific-parameter: parameter** **<CRLF>**
	Response	**RTSP/1.0 551 Option Not Supported** **CSeq: 21** **Unsupported: specific-feature** **<CRLF>**

3.5.5.3 RTSP Headers in Request and Response Messages

Header	Message	Methods					
		SETUP	PLAY	TEARDOWN	OPTIONS	DESCRIBE	NOT IMPLEMENTED
CSeq	Request	req.	req.	req.	req.	req.	req.
	Response	all	all	all	all	all	501
Session	Request	opt.(1)	req.	req.	opt.(2)	opt.(3)	
	Response	200	200	200	200(2)	200(3)	
Transport	Request	req.					
	Response	200					
com.ses.streamID	Response	200					
RTP-Info	Response		200				
Accept	Request					req.	
Content-Base	Response					200	
Content-Type(4)	Response	400, 403, 503	400, 403, 503	400, 403		200	
Content-Length	Response	400, 403, 503	400, 403, 503	400, 403		200	
Public	Response				200		501
Allow	Response	405	405	405			
Require	Request	opt.	opt.	opt.	opt.	opt.	
Unsupported	Response	551	551	551	551	551	

req. required header opt. optional header all apply to all responses

[1] A SETUP request that references an existing RTSP session shall include the "Session:" header field with the session value of that existing session.

[2] When using the OPTIONS request as a session keep alive mechanism, the request shall include a "Session:" header field with the session value of the session that is being kept alive. The response carries also the "Session:" header field with the same session value.

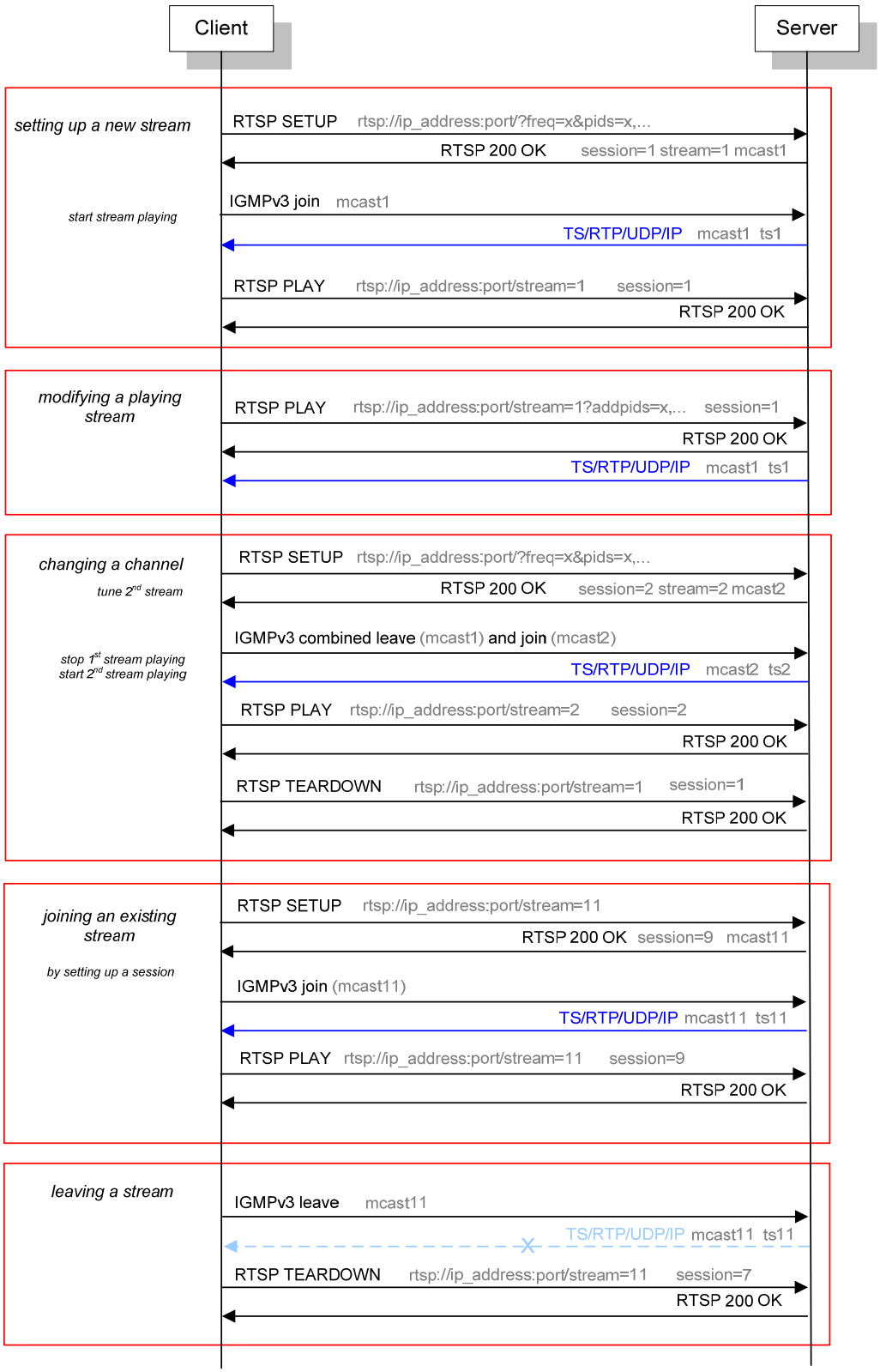
[3] A DESCRIBE request that references an existing RTSP session shall include the "Session:" header field with the session value of that existing session. The response carries also the "Session:" header field with the same session value.

[4] The Content-Type field takes the value:
- "application/sdp" in response to a successful DESCRIBE request containing an SDP description,
- "text/parameters" when the message body of the 400 response contains the "Check-Syntax:" parameter followed by the malformed syntax, when the message body of the 403 response contains the "Out-of-Range:" parameter followed by a space separated list of the attributes that are not understood or when the message body of the 503 response contains the "No-More:" parameter followed by a space separated list of the missing resources.

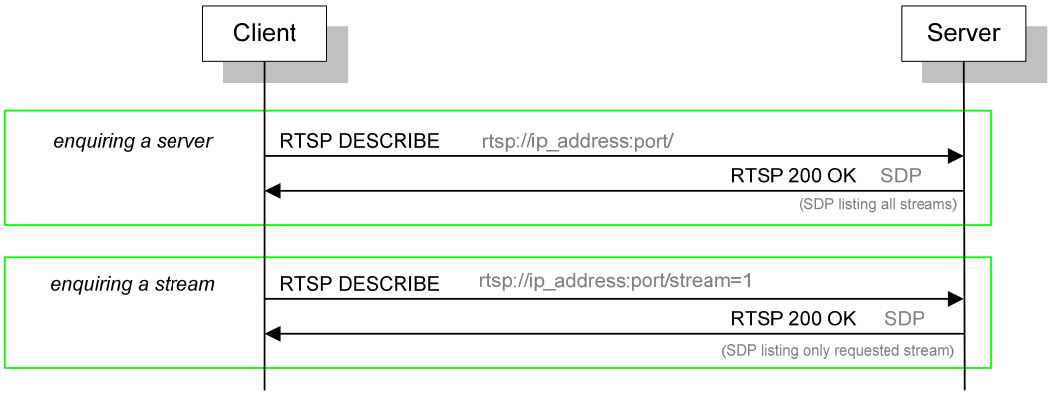
Please note that linear white spaces between header field names and header field values shall be ignored.

3.5.6 Sequence Diagrams

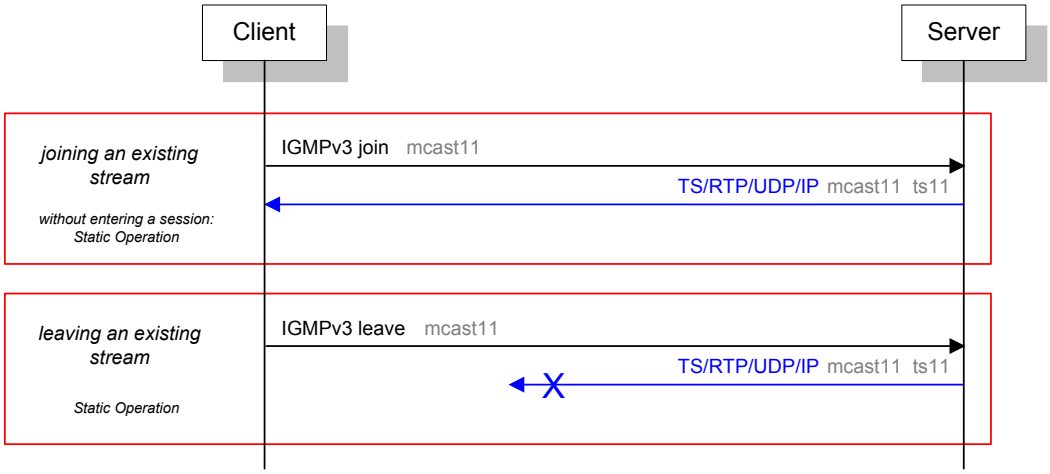
RTSP Operation Example



Requesting server / stream information through the RTSP DESCRIBE request:



Static Operation (see below):



3.5.7 RTCP Announcements

3.5.7.1 Description

Every RTP delivered media stream is accompanied by an RTCP announcement stream. The RTCP stream becomes available as soon as the RTP stream starts playing. This announcement stream carries information about the SAT>IP configuration for that particular TS media stream plus information about the real-time reception conditions.

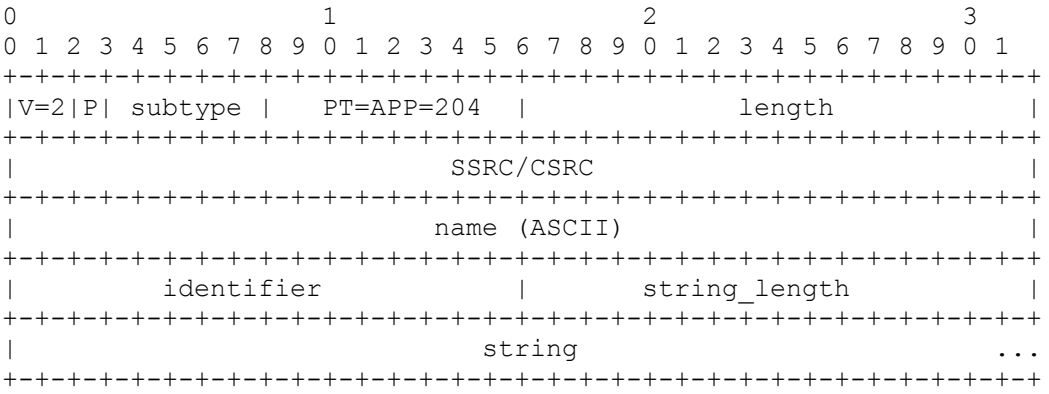
These announcements use an RTCP (RFC 3550) [8] control channel that is delivered on the same multicast address as the corresponding RTP stream. However the port number of the RTCP stream is the port number of RTP + 1.

The RTCP stream carries information about the signal level, tuner lock, signal quality and configured DVB reception parameters of the corresponding RTP stream. RTCP should provide 5 updates of the reception parameters per second.

In the context of this specification clients are required not to send any Receiver Reports back to the SAT>IP server.

3.5.7.2 Syntax

RTCP announcements are carried in RTCP Application-Defined APP packets as specified in RFC 3550 [8]. The APP packets have the following syntax:



The APP packet fields are specified below:

Field	Value	Example
name	Set to "SES1".	**SES1**
identifier	A 16-bit version field to allow multiple interoperating packet formats. Set to "0000".	**0000**
string_length	A 16-bit field specifying in hexadecimal the number of bytes of the string field.	**0048**
string	Field carrying the payload format specified below.	**See below.**

String Payload Format:

```
ver=<major>.<minor>;src=<srcID>;tuner=<feID>,<level>,<lock>,<quality>,<frequency>,<polarisa
tion>,<system>,<type>,<pilots>,<roll_off>,<symbol_rate>,<fec_inner>;pids=<pid0>,…,<pidn>
```

The string payload format is identical to the format following the "a=fmtp:33" attribute defined in the SDP file.

The text is encoded using UTF-8 encoding as specified in RFC 2279 [5].

- If the text string fills the packet to the next 32-bit boundary, the string is not null terminated.
- If not, the APP packet shall be padded with null bytes to the next 32-bit boundary (This padding is separate from that indicated by the P bit in the RTCP header).

Example Text String:

```
ver=1.0;src=1;tuner=1,240,1,7,12402,v,dvbs,,,,27500,34;pids=0,16,56,112,168,1709
```

Please note:

The APP packet is always part of a compound RTCP packet. According to the rules of RFC 3550 [8], a compound RTCP packet must contain, in addition to the APP packet, a Sender Report (SR) packet plus a Source Description Items (SDES) packet.

3.5.8 HTTP

SAT>IP Media servers shall also be controllable via HTTP based requests.

HTTP protocol support shall rely on HTTP version 1.1.

If a SAT>IP client would like to get a stream delivered via the HTTP transport protocol it shall use the HTTP GET method to initiate a stream playout.

The HTTP URI syntax is as follows:

HTTP_URI = "http://" ip_address [":"port] "/" ["?"query]

The HTTP query syntax is identical to the RTSP query syntax specified in the table under 3.5.3.2.

A streaming HTTP client should implement the "stop" media operation by disconnecting the TCP connection of the HTTP transaction.

Standard HTTP status code messages are returned in reply to HTTP requests.

Status code messages 400, 403 and 503 can make use of the syntactical extensions ("Check-Syntax", "Out-of-Range", "No-More") described for RTSP.

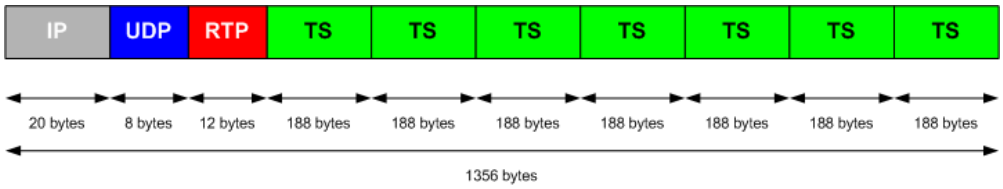
Example http get message:

```
http://192.168.128.1/?src=1&fe=1&freq=12402&pol=v&msys=dvbs&mtype=qpsk&sr=27500&fec=34&pids
=0,16,50,104,166,1707
```

3.6 Media Transport

3.6.1 RTP Transport

SAT>IP uses the industry standard RFC 2250 [4] for carrying transport streams. Generally, respecting the overall MTU size, 7 TS packets can be carried per IP datagram. However an IP packet may also contain less than 7 TS packets. RTP encapsulation is mandatory.



SAT>IP devices can carry MPTS or SPTS streams according to the session setup parameters. MPTS streams contain the entire original transport stream as received off the satellite. Partial Transport Streams or Single Program Transport Streams only carry a selection of the PIDs making up the original transport stream.

The SAT>IP receiver requests the PIDs required in order to present a requested program to the end-user. The SAT>IP receiver implements a DVB service logic and thus is capable of parsing DVB Service Information and Program Specific Information in order to correctly tune to DVB services.

Please note that when no signal is being received, the signal is being lost, or no PID is available, the SAT>IP server shall nevertheless issue an empty RTP packet at least every 100ms.

3.6.2 HTTP Streaming

The SAT>IP server shall also implement HTTP as a mandatory media transport protocol.

The HTTP header shall contain the following MIME type: "video/MP2T".

3.7 Media Formats

SAT>IP devices shall support the carriage of A/V content and related information in an MPEG-2 Transport Stream as specified in clause 4 of TS 101 154 [13].

The SAT>IP architecture does not require server-side A/V format transcoding.

4 Dynamic vs Static Server Operation

4.1 Dynamic Operation (default)

In normal operation SAT>IP clients request the delivery of TS media streams from SAT>IP servers. SAT>IP servers dynamically tune to satellite transponders as requested. This mode of operation is called dynamic.

Each time that a client requests the setup of a stream it creates a session with the server. Within the session the client can request modifications to the streams as needed.

Tuner resources are dynamically allocated by the server based on these requests.

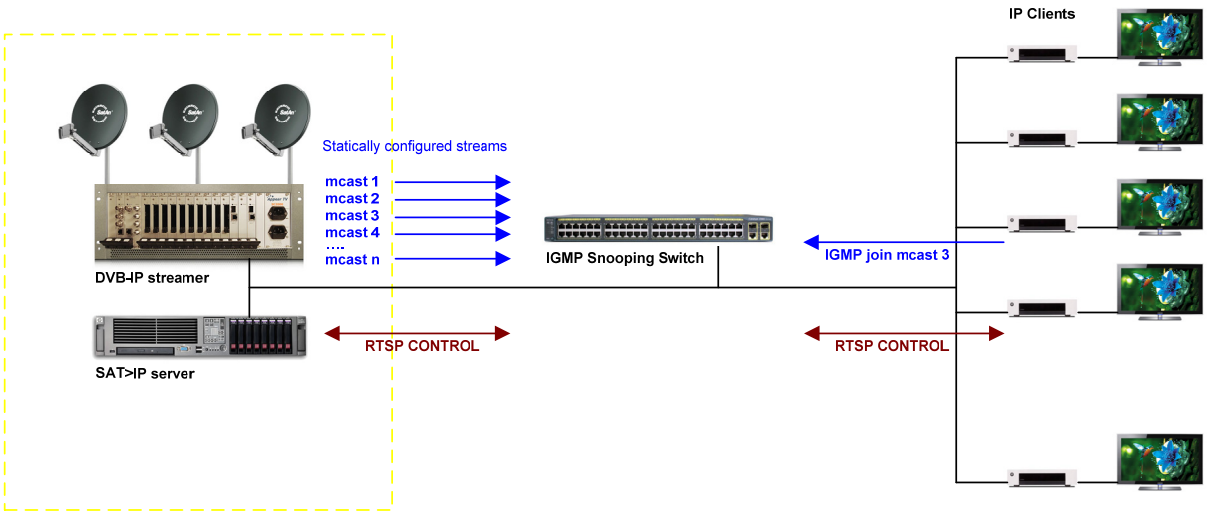
4.2 Static Operation

The SAT>IP protocol also supports a so-called static operation. This type of operation is suitable for very large installations with many tens, hundreds or even more clients. Examples for such deployments are hospitals, large hotels, new build areas where satellite signals are distributed via fiber.

In these installations a configuration terminal sets up the streams through RTSP requests and creates a session for each stream with the server. The configuration terminal as the virtual master of the sessions has total control over all the TS media streams.

Clients join and leave the streams without entering into an RTSP session by simply issuing IGMP join and leave messages as required.

The diagram below shows an example of such a system:



4.3 Mixed Operation

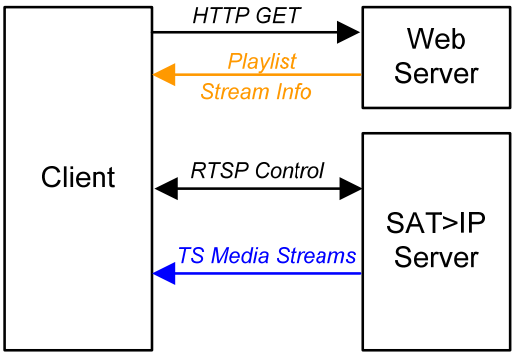
In mixed operation, SAT>IP servers have certain streams permanently configured (monopolising a part of the server's tuning resources) and another part of the resources available for dynamic tuning.

Appendix A (Informative): Client Autoconfiguration

Clients may optionally obtain stream information, not only by parsing DVB service information, but also directly from configuration files that can be downloaded from (local or remote) Autoconfiguration Servers.

This mechanism facilitates client implementation and allows clients to obtain information about the streams available to them without running a full DVB network scan and without implementing the full DVB service logic.

Autoconfiguration Servers are standard Web Servers holding information about SAT>IP streams.



SAT>IP clients according to this Appendix A, feature an option in their settings menu to specify a URL to an autoconfiguration server.

During startup, in normal operation, the client accesses the autoconfiguration server to get a file listing all the streams available on a given network.

The configuration file provides a name for each stream and a direct link (URI) to access the stream.

Initially it is recommended that clients according to this Appendix implement the format of standard M3U playlists. Further formats may be added later on for specific applications.

M3U playlist format

```
#EXTM3U
#EXTINF:0,Channel 1
rtsp://192.168.1.105/stream=1
#EXTINF:0,Channel 2
igmp://239.1.12.10:10000
#EXTINF:0,Channel 3
rtp://239.35.140.11:10000
#EXTINF:0,Das Erste
rtsp://192.168.1.105/stream=12
#EXTINF:0,ZDF
rtsp://192.168.1.105/stream=13
#EXTINF:0,3sat
http://192.168.128.1/?src=1&fe=1&freq=12402&pol=v&msys=dvbs&mtype=qpsk&sr=27500&fec=34&pids=0,16,50,104,166,1707
```