comcast®

Next Generation On Demand (NGOD)

RTSP Usage Specification 1.1

June 15, 2005

Document Status Sheet

Document Title:	RTSP Usage 1.1
Revision History:	ECR # 3 – 3/24/05 – Incorporate support for selections of stream control protocols
	ECR # 5 – 4/18/05 – Added fields to QAM transport header
	ECR # 7 – 3/22/05 – Clarify SDP playlist definition
	ECR # 10 – 3/22/05 – Correct playlist example based on ADI provider id and asset id format
	ECR #19 – 4/8/05 – Clarify RTSP headers are case sensitive
	ECR # 21 – 4/6/05 – Added reference to the InBandMarker extension header for SETUP request message. Added reference to InBandMarker extension header in setup message from SM.
	ECR # 49 – Added clarification on the representation of data within header fields
	ECR # 51 – Added explanation of abs_path component of RTSP URL
	ECR # 54 – 3/21/05 – Add Announce Codes used for R6
	ECR # 58 – 4/11/05 – Added indent to continuation lines of RTSP header fields
	ECR # 59 - 6/2/05 - Modified Require: header to use com.comcast.ngod.xx. Note that changes were already present in document, but the ECR was not shown in this Change List.
	ECR # 60 – 3/06/05 – Modified example QAM Transport headers to include mandatory “client” field
	ECR # 61 - Added Reason Text to Reason: header examples. Note that changes were already present in document, but the ECR was not shown in this Change List.
	ECR # 63 – 4/11/05 – Renamed SDP extension “playlist-item” to “X-playlist-item”
	ECR # 65 – 4/7/05 – Change reference numbers for Normative and Informative References (Section 2.1 and 2.2) to include respective N and I.
	ECR # 69 – 4/17/05 – Added announce_destination SET_PARAMETER
	ECR # 94 – 4/28/05 – Modified definition of ClientSessionId
	ECR # 95 – 5/2/05 – Added Session Groups to announce_destination SET_PARAMETER message
	ECR #100 – 5/2/05 – Added additional reason code 550 for “session time-out” and response code 551 for “option not

	supported", changed the position of presentation state to use the same definition of LSCP 1.0 NPT. 5/25/05 - Various modifications after initial 1.1 pre-release review 6/2/05 - Complete ECR 58 to make indentation on all messages consistent. 6/8/05 – Added clarification of session timeout behavior.			
Date:	June 15, 2005			
Status:	Work in Progress	Draft	Issued	Released
Distribution Restrictions:	VE Authors Only	VE & PE members only	Suppliers	None

Key to Document Status Codes:

Work in Progress (W)	An incomplete document, designed to guide discussion and generate feedback that may include several alternative requirements for consideration.
Draft (D)	A document in specification format considered largely complete, but lacking review by other VE and PE vendors. Drafts are susceptible to substantial change during the review process.
Issued (I)	A stable document, which has undergone rigorous VE & PE vendor review and is suitable for product design and development, cross-vendor interoperability, and for certification testing.
Released (R)	A stable document, reviewed, tested and validated, suitable to enable cross-vendor interoperability.

Contents

List of Tables

SCOPE

1.1 Introduction and Overview

The Comcast Next Generation On-Demand architecture leverages the RTSP protocol for Release 1 across several interfaces including S1, C1, S6, S3, and R2. It is necessary to make the RTSP syntax behavior as consistent as possible across those interfaces. This document attempts to unify the usage of RTSP across those interfaces.

1.2 Purpose of document

This document defines the subset of, and extensions to, RTSP in order to be compliant with the Comcast Next Generation On Demand architecture.

1.3 Scope

The following information will be included in this document:

- RTSP syntax across interfaces
- RTSP behavior across interfaces

Each respective interface document within the NGOD architecture that utilizes RTSP will include the following.

- Specific constraints and requirements of that interface
- Definition of socket-level transport (TCP only)
- Examples of each message supported

1.4 Requirements (Conformance Notation)

Throughout this document, the words that are used to define the significance of particular requirements are capitalized. These words are:

"MUST"	This word or the adjective "REQUIRED" means that the item is an absolute requirement of this specification. The word "MANDATORY" may be used in lieu of "MUST" in certain circumstances.
"MUST NOT"	This phrase means that the item is an absolute prohibition of this specification.
"SHOULD"	This word or the adjective "RECOMMENDED" means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.

“SHOULD NOT”	This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
“MAY”	This word or the adjective “OPTIONAL” means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

2 REFERENCES

2.1 Normative References

The following documents may be of use in understanding this interface.

[N-1] H. Schulzrinne; et al, Real Time Streaming Protocol (RTSP), RFC 2326, April 1998

[N-2] R. Fielding; et al, Hypertext Transfer Protocol -- HTTP/1.1, RFC 2068, January 1997

[N-3] Berners-Lee, T., and D. Connolly, "HyperText Markup Language Specification - 2.0", RFC 1866, November 1995

[N-4] M. Handley; et al, SDP: Session Description Protocol, RFC 2327, April 1998

[N-5] H. Schulzrinne; et al, Real Time Streaming Protocol (RTSP), RFC 2326bis-07, July 19, 2004

[N-6] Time Warner Pegasus Light Stream Control Protocol, V1.0 June 10, 1999

[N-7] The Leach, Mealling and Salz Draft for the Specification of UUIDs, IETF (standards body)

2.2 Informative References

N/A

3 TERMS AND DEFINITIONS

This specification defines the following terms:

N/A

4 ABBREVIATIONS AND ACRONYMS

This specification uses the following abbreviations and acronyms:

Table 1 - Abbreviations and Acronyms

ES	Elementary Stream
NGOD	Next Generation On Demand
NPT	Normal Play Time – format as defined in the RTSP RFC 2326
PID	Program Identifier
PMT	Program Map Table
RTSP	Real-time Streaming Protocol

TCP	Transmission Control Protocol
UDP	Universal Datagram Protocol
UUID	Universally Unique Identifier

5 INTERFACE SPECIFICATION

5.1 Key Decision Drivers, Scope, and Constraints

The following key decisions and constraints bound the architecture and associated interfaces.

5.2 Overall Architecture

This document is based on the overall architecture defined in RFC 2326 unless otherwise noted. The base of the RTSP defines the interaction between RTSP client and server. Each NGOD interface that utilizes RTSP will define which component will play the role of client and which will play the role of the server. Please refer to the respective NGOD specification for usage constraints within that interface.

5.3 Protocol Interfaces

The protocol chosen for these interfaces is based on the Real-time Streaming Protocol (RTSP). The RTSP protocol has various options that, if not implemented uniformly, could lead to a lack of interoperability. This specification will define a unifying minimal subset of RTSP messaging in order to be compliant with the Comcast Next Generation On Demand (NGOD) Architecture. This specification utilizes RFC 2326 as the base RTSP specification for all syntax and behavior unless otherwise noted.

The architecture shall support TCP socket-level transport. RTSP over UDP is not required in NGOD. The type of socket-level transport used for a particular interface shall be defined in the respective interface document.

6 RTSP METHODS

6.1 Introduction

This section defines the RTSP methods that will be required within the NGOD architecture.

6.2 RFC 2326

The method types supported in the NGOD use of RTSP are the same as RFC 2326 with the exception of the PING method, which is described below.

6.3 PING

The syntax and usage of the PING method within the NGOD architecture will be as described the “draft-ietf-mmusic-rfc2326bis-07” draft of the RTSP specification.

The following is an example use of the PING method.

Client to Server request:

```
PING rtsp://sessionmanager2.comcast.com:554 RTSP/1.0
CSeq: 123
Session:12345678
Require: com.comcast.ngod.s1
```

Server to Client response:

```
RTSP/1.0 200 OK
CSeq: 123
Session:12345678
```

6.4 Required Support of Methods

The minimal subset of RTSP methods required by an NGOD interface is as follows.

- ANNOUNCE
- SET_PARAMETER
- GET_PARAMETER
- PING

Whether or not a particular method is required to be supported within a particular NGOD interface will be specified within that interface’s specification.

7 RTSP URL EXTENSION

7.1 ABS_PATH

The RSTP URL can be extended to include an <abs_path> component. In NGOD this can be used for a client to request a particular resource. For example, over the R2 interface, an ODRM may request a particular logical SOP from the streaming server:

```
rtsp://<streaming-server-path>:<streaming-server-port>/<sop> RTSP/1.0
```

8 RTSP HEADERS

8.1 Introduction

This section describes the following:

- Required and optional headers that are required for each method
- The definition of each extension header, including purpose and syntax
- Clarification on each modified header
- All of the RTSP headers shall be case insensitive

8.2 Data Representation

RTSP Generic Specification: RTSP format for the following fields should be:

- Client IDs – where the client ID is a MAC address, it will be represented as a 12-digit hex ASCII value, no “0x” prefix, no “:”, and with leading zeros. Where there is no valid client ID available, the value “FFFFFFFFFFFF” shall be used.
- Bandwidth – decimal ASCII representation of an integer in bits/second
- All RTSP session IDs – decimal ASCII representation of an integer
- All IP’s are in the form n.n.n.n where n is a decimal ASCII representation of an integer
- Ports are decimal ASCII representation of an integer

8.3 RTSP Header Extensions

The table below describes additional requirements beyond RFC2326 for the inclusion of headers within RTSP messages.

Type "RequestResponse" designates general request headers to be found in both requests and responses, type "Request" designates request headers, and type "Response" designates response headers. Fields marked with "Required" in the column labeled "support" MUST be implemented by the recipient for a particular method, while fields marked "Optional" are optional. Note that not all fields marked "Required" will be sent in every request of this type. The "Required" means only that client (for response headers) and server (for request headers) MUST implement the fields. The last column lists the method for which this header field is meaningful.

Table 2 – RTSP Header Extensions

Header	Type	Support	Methods
ClientSessionId	RequestResponse	Required for S1 only	SETUP
Notice	Request	Required	ANNOUNCE
OnDemandSessionId	RequestResponse	Required	SETUP, TEARDOWN ANNOUNCE
Reason	Request	Required	TEARDOWN
Require	Request	Required	SETUP, ANNOUNCE, SET_PARAMETER GET_PARAMETER, PING, PLAY, PAUSE
SessionGroup	Request	Optional	SETUP, GET_PARAMETER, SET_PARAMETER
Policy	Request	Optional	SETUP
StreamControlProto	Request	Optional	SETUP
InBandMarker	Request	Optional	SETUP
JitterBuffer	RequestResponse	Optional	SETUP (see R6)
ReportTrafficMismatch	Request	Optional	SETUP (see R6)
ProvisionPort	Request	Optional	SETUP (see R6)
StartChecking	Request	Optional	SETUP (see R6)
StopChecking	Request	Optional	SETUP (see R6)
Sop	Response	Optional	SETUP
SopGroup	Response	Optional	SETUP

8.4 Extension: ClientSessionId

The “ClientSessionId” header defines a unique session identifier created by an On Demand Client. This identifier is globally unique. ClientSessionId is used in the S1 session setup message as the client’s identifier of the session. It is analogous to the DSM-CC SessionID field.

The syntax of a ClientSessionId header is identical to the “session-id” defined in section 3.4 of RFC2326.

The syntax of a ClientSessionId is a 20 character ASCII representation of a 10 byte hexadecimal value. The most significant 6 bytes are the client ID, the least significant 4 bytes are a session ID unique to the client. The combination of the two provides a globally unique identifier.

An example of the ClientSessionId header follows.

```
ClientSessionId: 00AF123456DE00000001
```

This ClientSessionId represents a client with a MAC address of 00:AF:12:34:56:DE and a session ID of 00000001.

It should be noted that the ClientSessionId is different from the RTSP session field between the On Demand Client and Streaming Server. ClientSessionId is generated by the On Demand Client. The RTSP session field between the On Demand Client and Streaming Server is assigned by the Streaming Server and returned to the client over S3 for further stream control. This is equivalent to the function of Stream Handle in the DSM-CC S1/C1 specification.

8.5 Extension: Notice

The “Notice” header defines information sent from a RTSP server to a RTSP client within an ANNOUNCE message.

The syntax of the Notice header is as follows.

```
Notice: <Notice-code>  <text description>
  event-date=<datetime> npt=<npt-value> [,<Notice-code>
   <text description>
  event-date=<datetime> npt=<npt-value>]*
```

The NPT should be in milliseconds. In addition, the definitions of NPT should be consistent with those defined in LSC 1.0 [6]. It will be represented as an 8-digit hex ASCII value, no “0x” prefix and no leading zero. Where Notice is used but npt is not known, npt-value (but not the attribute) will be omitted.

Please see “12.3 ANNOUNCE Codes” below for a full list of the valid options for Notice-code.

An example of the Notice header follows.

```
Notice: 2101 "End-of-Stream Reached"
  event-date=19930316T064707.735Z npt=2314223
```

8.6 Extension: OnDemandSessionId

The “OnDemandSessionId” header defines the unique session identifier that the NGOD Session Manager assigns to a given session.

The syntax of the OnDemandSessionId header is as follows.

```
OnDemandSessionId: <id>
```

where <id> is a string representation of a 128-bit UUID as defined by IETF. The UUID will be constructed using the MAC address of the system on which the UUID is created to ensure global uniqueness.

An example of the OnDemandSessionId header follows.

```
OnDemandSessionId: be074250-cc5a-11d9-8cd5-0800200c9a66
```

8.7 Extension: Reason

The “Reason” header defines the reason for a particular TEARDOWN message.

The syntax for the Reason header is as follows.

```
Reason: <reason-code>  <reason-text>
```

where <reason-code> is an integer and <reason-text> is a textual description of the reason.

Please see “12.4 TEARDOWN Reason Codes” for a full list of the valid reason codes.

An example of the Reason header follows.

```
Reason: 200  user pressed stop
```

8.8 Extension: Require

The “Require” header will be implemented per the “draft-ietf-mmusic-rfc2326bis-07” draft of the RTSP specification.

The “Require” header defines a mechanism by which the client indicates in a request that certain extensions are required to fulfill this request.

The syntax of the Require header is as follows.

```
Require: com.comcast.ngod.<interface-id>
```

where <interface-id> is the NGOD identifier of the interface, e.g. “s1”, “c1”, s3”.

The specific NGOD interfaces that use this feature will define the exact usage.

An example of the Require header follows.

```
Require: com.comcast.ngod.s1
```

8.9 Extension: SessionGroup

The “SessionGroup” header defines a token passed on a SETUP request that is used to identify a group of sessions. This header then may be used in GET_PARAMETER requests to clarify the request.

The syntax of the SessionGroup header is as follows.

```
SessionGroup: <token>
```

where <token> is a string.

An example of the SessionGroup header follows.

```
SessionGroup: SM1
```

The SessionGroup header is used in the process of re-synchronizing session state between two NGOD components. The SessionGroup token is generated by an RTSP client component to group its established sessions together. A SessionGroup token is passed to a RTSP server within SETUP request messages. RTSP servers remember the SessionGroup token for each established session. RTSP clients may send a GET_PARAMETER request for the “session_list” parameter, passing the SessionGroup token. The RTSP server returns the list of session IDs for currently active sessions that have that SessionGroup token. Please see the “session_list” extension section for more details. Note that a session may only belong to a single Session Group.

For example, a given Session Manager (#1) may choose to represent its sessions with the SessionGroup token “SM1”. This value is configured to be unique. Within each SETUP request this Session Manager sends to an ODRM it will pass “SessionGroup: SM1”. When this Session Manager wants to synchronize sessions with the ODRM it will send a GET_PARAMETER of session_list, passing “SessionGroup: SM1”. The ODRM RTSP server will return the list of sessions with a SessionGroup of SM1. Note that there may be a second Session Manager establishing sessions on the same ODRM, but it could be configured to pass “SessionGroup: SM2”. In this manner each Session Manager may synchronize with the ODRM for solely the sessions that it established.

8.10 Extension: Policy

The “Policy” header defines a mechanism by which the client indicates in a request that certain policies are requested to be taken into consideration.

The syntax of the Policy header is as follows.

```
Policy: <policy-name>=<policy-value>
     [,<policy-name>=<policy-value>]*
```

Where

- <policy-name> is the name of a specific policy

- <policy-value> is the value for that policy represented by an integer number.

The specific NGOD interfaces that use this feature will define the exact usage and behavior. For priority policy, it uses an integer value starting, 1 being the highest priority.

Examples of the Policy header follows.

```
Policy: priority=1
```

8.11 Extension: InBandMarker

The "InBandMarker" header defines a mechanism by which Resource Managers can request that specific data be injected into the In-Band stream by an Edge Device. For example, this would allow the Edge Device to be instructed to install CA Descriptors to support pre-encryption.

In-Band data to be injected can be specified as an MPEG descriptor, an MPEG Table, or as an MPEG packet. This data can be injected into the In-Band stream in a number of ways. The following table defines the specific types of data which can be injected, along with the rules for how they will be injected.

Marker Type	**Description**	**Desired Device Behavior**	**Data Delivered over RTSP**
Type 1	Descriptor in PMT header section i.e. before ES List.	Device must modify the incoming PMT by placing the descriptor into the header and then recalculating the CRC.	Descriptor
Type 2	Separate table but in same transport packet as PMT	Device must append table to end of PMT packets if space is available. If not, device may discard after sending RTSP announce. Device may also convert to type 3 if capability exists.	Mpeg table.

Marker Type	Description	Desired Device Behavior	Data Delivered over RTSP
Type 3	Same PID as PMT but different transport packet	Device must ensure that CC stream remains continuous between marker packets and PMTs. If a table is passed, the device must build an mpeg packet. If packet is given, the device maybe required to replace the PID value.	Mpeg table, Mpeg Packet
Type 4	Tables carried in TS packets on separate data PID. Referred to in ES section of PMT	Device may be called upon to modify PMT in order to place ES record in it. Device must ensure that CC stream for PID is continuous. If a table is passed, the device must build an mpeg packet. If packet is given, the device maybe required to replace the PID value.	Mpeg table, Mpeg packet
Type 5	Descriptor in ES section. Specifically, following the ES List PID values. Can be on a per ES basis.	Device must rebuild the PMT and recalculate the CRC. Device may be called on to add the ES record if it doesn't exist.	Mpeg Descriptor

The syntax of the InBandMarker header is as follows.

```
InbandMarker:  <IBMarkerElement>
             [,<IBMarkerElement>]*
```

The IBMarkerElement is defined as follows:

```
type = (1|2|3|4|5);
[ESType=<elementary stream type>;]
[ESCount=<es type count>;]
[pidType=(A|R);]
[pidValue=<pid>;]
[insertRate=<insert Rate>;]
[insertDuration=<duration>;]
[dataType =(D|T|P);]
data= datum[datum]*
```

where

<elementary stream type> the stream's elementary stream type referenced in the PMT.

<Es type count> attempts to isolate which ES within the stream of a given type e.g. second audio

<pidType A> absolute. The PID value would be an absolute value.

<pidType R> relative. The PID value is a number that should be added to the current PMT PID to derive the PID in which the data should be injected.

<pidValue> must be an unsigned

<insertRate> Specified the number of milliseconds between insertion of this In-Band Marker.

<insertDuration> Specifies the length of time in milliseconds the data should be injected. 0 or missing means for the length of the session.

<Data type> indicates what is being passed in the data i.e. [D]escriptor, [T]able, or [P]acket.

<Data> ASCII representation of a hex value i.e. two ASCII chars per octet.

The fields required in a given InBandMarker header depend on the In-Band Marker Type. The following table specifies which fields are required for each In-Band Marker Type.

Field	**Type 1**	**Type 2**	**Type 3**	**Type 4**	**Type 5**
ES Type	NA	NA	NA	Optional: If this value is present, the device must add an ES record of this type to the ES section of the PMT.	Required

Field	Type 1	Type 2	Type 3	Type 4	Type 5
ES Count	NA	NA	NA	Optional: if ES type is not present, this value has no meaning. If there are already ES records of the same type present in the PMT, the ES descriptor should appear as the Nth descriptor of this type. If there are N+m ES values of this type in the PMT already, the device does not need to modify the PMT.	Optional: The Nth ES record of that ES type in the PMT should be modified to contain the following descriptor.
Pid Type	NA	NA	NA	Required:	NA
Pid Value	NA	NA	NA	Required:	NA
InsertRate	NA	Optional: Default value= every PMT	Optional: default value = 125ms	Optional: default = 125ms	NA
InsertDuration	Optional: Default = length of session	Optional: Default = length of session	Optional: Default = length of session	Optional: Default = length of session	Optional: Default = length of session

Field	Type 1	Type 2	Type 3	Type 4	Type 5
DataType	NA	P	T,P Note, if a packet is delivered and the PID is not the same value as the PMT pid, the device must restamp the PID for the packet to be that of the PMT pid.	T,P Note, if a packet is delivered and the PID is not the same value as the pid value parameter, the device must restamp the PID for the packet.	NA

Examples of the InBandMarker header follows.

```
InbandMarker: type=1;data=0906454701EE,
  type=4;pidType=A;pidValue=01EE;dataType=T;data=C0123457 ,
  type=4;pidType=A;pidValue=01EE;dataType=T;data=C1432000 ,
  type=4;pidType=A;pidValue=01EE;dataType=T;
  insertDuration=10000;data=4002003030
```

8.12 Extension: StreamControlProto

The “StreamControlProto” header defines a mechanism by which the client indicates in a request about the stream control capability for interface C1.

The syntax of the StreamControlProto header is as follows.

```
StreamControlProto: <type>[,<type>]*
```

- Where <type> is the string to indicate the type(s) of the stream control protocol supported over C1. The currently defined values are: rtsp (RTSP over TCP), lscp (LSCP over TCP), lscpu (LSCP over UDP).

Example:

```
StreamControlProto: lscp
```

- The StreamControlProto header can be applied to various NGOD interfaces based on the following rules:

- S3, S3', or R2: If the StreamControlProto header is missing over these interfaces, only type "lscp" is assumed. Otherwise, the StreamControlProto header(s) should be passed on (or filtered before passed on) by the SM from the interface S1 to S3, S3', and R2.

- S1-DSMCC: If the StreamControlProto header is missing over interface S1-DSMCC session setup request, only type "lscp" is assumed. In this case, it is optional for SM to add the StreamControlProto header with type "lscp" over interfaces S3, S3', or R2 since the default value is "lscp" over these interfaces. If a session setup request is received by the SM via S1-DSMCC with the StreamControlProto header(s), the header(s) is passed on (or filtered before passed on) by the SM to the interfaces S3, S3', and R2.

- S1-RTSP: If the StreamControlProto header is missing over interface S1-RTSP session setup request, only type "rtsp" is assumed. In this case, it is mandatory for SM to add the StreamControlProto header with type "rtsp" over interface S3, S3', and R2. If a session setup request is received by the SM via S1-RTSP with the StreamControlProto header(s), the header(s) is passed on (or filtered before passed on) to the interfaces S3, S3', and R2.

Based on the C1 capabilities received from the StreamControlProto header or inferred over interface R2, the Streaming Server must indicate available stream control port choices for the session within the SDP control parameters as defined bellows:

```
a=control:<protocol>://<host>:<port>/<streamhandle>
```

- <protocol> indicates the stream control protocol selected e.g. rtsp, lscp, lscpu.
- <host> is the IP address or fully qualified DNS name of the stream control port.
- <port> is the TCP or UDP port number of the stream control port. Available TCP and UDP choices may use identical port numbers.
- <streamhandle> is the RTSP session ID or LSCP stream handle to control this stream, depending on the <protocol> value. Available RTSP and LSCP choices may use identical values.

8.13 Transport Header Extensions

The "Transport" header defines the transport parameters acceptable to the client for data transmission.

8.13.1 Transport Header for QAM Transport

The syntax of the Transport header for QAM transport is as follows.

```
Transport:
  MP2T/DVBC/QAM;
  unicast;
  client=<client-id>;
```

```
        [destination=<destination>;]
        [bandwidth=<bandwidth>;]
        [qam_name=<qam_name>;]
        [qam_group=<qam_group>;]
        [annex=<annex>;]
        [channel_width=<channel_width>;]
        [modulation=<modulation>;]
        [interleaver=<interleaver>]

        [,
        MP2T/DVBC/QAM;
        unicast;
        client=<client-id>;
        [destination=<destination>;]
        [bandwidth=<bandwidth>;]
        [qam_name=<qam_name>]
        [qam_group=<qam_group>;]
        [annex=<annex>;]
        [channel_width=<channel_width>;]
        [modulation=<modulation>;]
        [interleaver=<interleaver>]
        ]*
```

where

- <client-id> is a unique identifier of the client
- <destination> describes the "<frequency>.<program-number>" where <frequency> describes the frequency in Hertz with which to tune to the request stream and <program-number> is the program number of the requested stream.
- <qam_name> is a string representation of a QAM name
- <qam_group> is a string representation of a QAM Group
- <bandwidth> is the bandwidth requirement in bits per second.
- <annex> possible values (A|B|C), if omitted defaults to B
- <channel_width> possible values (6|7|8), if omitted defaults to 6
- <modulation> possible values (qam64|qam256|qam1024), if omitted defaults to qam256
- <interleaver> possible values (I128J1 | 128J2 | I64J2 | I128J3 | I32J4 | I128J4 | I16J8 | I128J5 | I8J16 | I128J6 | I128J7 | I128J8), if omitted defaults to I128J4

8.13.2 Transport Header for UDP Transport

The syntax of the Transport header for UDP transport is as follows.

```
    Transport:
      MP2T/DVBC/UDP;
      unicast;
      client=<client-id>;
      [qam_destination=<qam_destination>;]
      [destination=<destination>;]
```

```
        [client_port=<client-port>;]
        [source=<source>;]
        [server_port=<server-port>;]
        [bandwidth=<bandwidth>;]
        [qam_name=<qam_name>;]
        [qam_group=<qam_group>;]
        [annex=<annex>;]
        [channel_width=<channel_width>;]
        [modulation=<modulation>;]
        [interleaver=<interleaver>]

        [,
        MP2T/DVBC/<lower-transport>;
        unicast;
        client=<client-id>;
        [qam_destination=<qam_destination>;]
        [destination=<destination>;]
        [client_port=<client-port>;]
        [source=<source>;]
        [server_port=<server-port>;]
        [bandwidth=<bandwidth>;]
        [qam_name=<qam_name>;]
        [qam_group=<qam_group>;]
        [annex=<annex>;]
        [channel_width=<channel_width>;]
        [modulation=<modulation>;]
        [interleaver=<interleaver>]
        ]*
```

where:

- <client-id> is a unique identifier of the client
- <qam_destination> describes the "<frequency>.<program-number>" where <frequency> describes the frequency in Hertz with which to tune to the request stream and <program-number> is the program number of the requested stream. Note that this is used in a special case whereby the Session Manager requests the UDP and QAM transport information from the ERM.
- <destination> describes the IP/host address to which data is streamed. E.g. IP of EdgeQAM device.
- <client-port> describes the port to which the data is streamed. E.g. an input port for an EdgeQAM device
- <source> is the IP/host address from which the data is streamed. E.g. the IP address of a streaming server
- <server_port> is the port from which the data is streamed. E.g. the output port of a streaming server
- <qam_name> is a string representation of a QAM name
- <qam_group> is a string representation of a QAM group

- <bandwidth> is the bandwidth requirement in bits per second.
- <annex> possible values (A|B|C), if omitted defaults to B
- <channel_width> possible values (6|7|8), if omitted defaults to 6
- <modulation> possible values (qam64|qam256|qam1024), if omitted defaults to qam256
- <interleaver> possible values (I128J1 | 128J2 | I64J2 | I128J3 | I32J4 | I128J4 | I16J8 | I128J5 | I8J16 | I128J6 | I128J7 | I128J8), if omitted defaults to I128J4

Note that multiple qam-names may be represented in a single Transport header.

Note that specific constraints on which parameters are required in the context of individual interfaces and messages will be defined in the specific interface specification.

8.13.3 Examples of Transport Header

The following examples are not intended as an exhaustive list of all possible syntax combinations. For the exact syntax of a given Transport header consult the NGOD interface specification for that interface.

An example of the SETUP request Transport header of On Demand Client to Session Manager follows.

```
Transport:
  MP2T/DVBC/QAM;unicast;client=00AF123456DE;
  qam_name=Chicago.Southbend.5,
  MP2T/DVBC/QAM;unicast;client=00AF123456DE;
  qam_name=Chicago.Southbend.10
```

An example of the SETUP response Transport header of Session Manager to On Demand Client follows.

```
Transport:
  MP2T/DVBC/QAM;unicast;client=00AF123456DE;
  destination=24000000.23
```

An example of the SETUP request Transport header of Session Manager to ERM follows.

```
Transport:
  MP2T/DVBC/QAM;unicast;bandwidth=3750000;
  qam_name=Chicago.Southbend.5;client=00AF123456DE,
  MP2T/DVBC/QAM;unicast;bandwidth=3750000;
  qam_name=Chicago.Southbend.10;client=00AF123456DE
```

An example of the SETUP response Transport header of ERM to Session Manager follows.

```
Transport:
  MP2T/DVBC/UDP;unicast;destination=192.26.13.1;
  client_port=4588;qam_destination=24000000.23;
  annex=B;modulation=qam256;interleaver=I128J4;
  client=00AF123456DE
```

An example of the SETUP request Transport header of Session Manager to ODRM follows.

```
Transport:
  MP2T/DVBC/UDP;unicast;destination=192.26.13.1;
  client_port=4588;client=00AF123456DE
```

9 SDP ATTRIBUTES

9.1 Introduction

This section defines clarifications and extensions to valid SDP attributes.

9.2 Extension: a=X-playlist-item

The NGOD implementation will utilize the “a=X-playlist-item” type to describe playlist elements to be played out within a session. Note that the “a=X-playlist-item” lines within the SDP section shall be in the order of desired playout.

The syntax of the playlist item definition is as follows.

```
a=X-playlist-item: <provider-id> <asset-id>[ <range>]
```

where

- “a=” is SDP syntax for a media attribute. For more information on SDP syntax please see RFC 2327.
- <provider-id> is the CableLabs provider_id for the content asset
- <asset-id> is the CableLabs asset_id for the content asset
- <range> is the playout range in the form: [<start-npt>]-[<stop-npt>]

The NPT should be in milliseconds. In addition, the definitions of NPT should be consistent with those defined in LSC 1.0 [6]. It will be represented as an 8-digit hex ASCII value, no “0x” prefix and no leading zero.

An example of the X-playlist-item type follows.

```
a=X-playlist-item: comcast.com abcd1234567890123456
1000-6000
```

9.3 Required Use of SDP

Use of SDP text is required in the following NGOD specifications.

- S1 – SETUP response
- S3 – For setups from the Session Manager to the ODRM - SETUP request and response
- R2 – SETUP request and response

9.4 SDP Example

For a comprehensive syntax definition of SDP, please see RFC 2327. For the reader’s convenience, an example of SDP text is included below.

The following is an example of the SDP SETUP Request text that applies to the NGOD R2 and S3 interfaces. Note that use of SDP is not required for S1 SETUP requests.

Within a RTSP R2 or S3 SETUP Request:

Table 3 – Example SDP SETUP Request

SDP Text Example	Description
v=0	This is the SDP protocol version number. For this NGOD release, this value is 0.
o=- 1234 2890842807 IN IP4 10.47.16.5	The fields: (1) Email address. For this release this will be “-“. (2) Session identifier. In requests, it is the OnDemandSessionId ID. (3) Session description version. It should be a Network Time Protocol (NTP) format timestamp. This should be the time that the session setup message was created. (4) Indicates address format. “IN” indicates internet format. (5) Indicates address format. “IP4” indicates internet protocol version 4. (6) IP address creating the session. In requests, it's the address of the server initiating the request (SM or ODRM).
s=	This is a text string description of the session. For this NGOD release this value is “ “.

SDP Text Example	Description
t=0 0	Describes the validity start/end times of the session. 0 indicates media is always available.
a=X-playlist-item: comcast.com abcd1234567890123456 1000-6000	See playlist-item description above.
c=IN IP4 0.0.0.0	Describes the media stream. This may be considered a fixed string for this NGOD release.
m=video 0 udp MP2T	This may be considered a fixed string for this NGOD release.

SDP text within a S1, S3 or R2 RTSP SETUP Response:

Table 4 – Example SDP SETUP Response

SDP Text Example	Description
v=0	This is the SDP protocol version number. For this NGOD release, this value is 0.
o=- 777 2890842817 IN IP4 1.2.3.4	The fields: (1) Email address. For this release this will be “-“. (2) Session identifier. In responses, it is always the RTSP session ID of this session on the Streaming Server. (Note on S1: The On Demand Client will use this session ID when sending RTSP stream control commands over C1.) (3) Session description version. It should be a Network Time Protocol (NTP) format timestamp. This should be the time that the session setup message was created. (4) Indicates address format. “IN” indicates internet format. (5) Indicates address format. “IP4” indicates internet protocol version 4. (6) IP address of the RTSP server on the Streaming Server.

SDP Text Example	Description
s=	This is a text string description of the session. For this NGOD release this value is " ".
t=0 0	Describes the validity start/end times of the session. 0 indicates media is always available.
a=control:lscp://videoserver234.comcast.com:554/9876	The syntax of the control location is as follows. `a=control:<protocol>://<host>:<port>/<streamhandle>` where • "a=" is SDP syntax for a media attribute. For more information on SDP syntax please see RFC 2327. <protocol> indicates the stream control protocol selected e.g. rtsp, lscp, lscpu. <host> is the IP address or fully qualified DNS name of the stream control port. <port> is the TCP or UDP port number of the stream control port. Available TCP and UDP choices may use identical port numbers. <streamhandle> is the RTSP session ID or LSCP stream handle to control this stream, depending on the <protocol> value. Available RTSP and LSCP choices may use identical values.
c=IN IP4 0.0.0.0	Describes the media stream. This may be considered a fixed string for this NGOD release.
m=video 0 udp MP2T	This may be considered a fixed string for this NGOD release.

10 GET_PARAMETER EXTENSIONS AND CLARIFICATIONS

10.1 Introduction

This section describes extensions and clarifications to the valid parameter options and responses for the GET_PARAMETER method.

10.2 Parameter Options

The complete list of parameter types are as follows.

Table 5 – Parameter Options

Parameter Type	Description
`connection_timeout`	Timeout setting for activity on a connection.
`session_list`	List of current active sessions.
`position`	The current stream position. The position uses the NPT format defined in the LSCP 1.0 protocol in 8-digit hex ASCII value, no “0x” prefix and no leading zero.
`presentation_state`	Current state of the stream
`scale`	The current play scale. E.g. 1.0, 7.0, -7.0

10.3 Clarification: presentation_state

The list of possible return values for presentation_state is as follows.

```
init
ready
play
pause
```

Fast forward will be represented with a presentation_state of “play” and a scale greater than 1.0 . Rewind will be represented with a presentation_state of “play” and a negative scale value.

10.4 Extension: connection_timeout

The RTSP server shall support a parameter of “connection_timeout”.

The return value from a get_parameter of connection_timeout shall be as follows.

```
     connection_timeout: <timeout>
```

where <timeout> is an integer representing seconds.

An example of a get_parameter connection_timeout response follows.

```
     connection_timeout: 300
```

10.5 Extension: session_list

The RTSP server shall support a parameter of “session_list”. As an optional message header to the GET_PARAMETER session_list request, the RTSP server shall support the SessionGroup header.

The return value conveys a list of sessions’ IDs for sessions that are active and were setup with the SessionGroup value. If the SessionGroup header is omitted, all sessions will be returned.

The syntax for the return value for a GET_PARAMETER session_list is as follows.

```
session_list: [<rtsp-session-id>:<on-demand-session-id>]
     [<rtsp-session-id>:<on-demand-session-id>]*
```

where <rtsp-session-id> is the RTSP server’s session ID and <on-demand-session-id> is the OnDemandSessionId generated by the Session Manager.

An example return value for a GET_PARAMETER session_list follows.

```
     session_list: 8543694731153482370:19006
          4980897343209869699:19007 854526448847611567:19005
          2977006442459148853:19008 949369239213932872:19004
          1788211202849211600:19009
```

For more information on how the session_list information is used by NGOD components, please see the “Extension: SessionGroup” section.

11 SET_PARAMETER EXTENSIONS AND CLARIFICATIONS

11.1 Introduction

This section describes extensions and clarifications to the valid parameter options and responses for the SET_PARAMETER method.

11.2 Parameter Options

The complete list of parameter types are as follows.

Table 6 – Parameter Options

Parameter Type	Description
`sessionGroups`	A list of session groups to be associated with the connection used for the SET_PARAMETER message.

11.3 Clarification: sessionGroups

Normally the connection between the RTSP client and server is persistent and announce messages will be sent across it. However, if the connection is broken due to an unexpected failure, there needs to be a way for the RTSP server to re-associate sessions with a particular client connection. When either the original RTSP client, or a designated replacement (vendor innovation) connects, it sends the list of session groups for which it is responsible. When the server requires to send an announce for a session that was created on a now broken connection, it matches the sessions' group to the group lists for all current connections. If a match is found, the announce message will be sent over that connection.

The format is as follows:

```
sessionGroups: <session_group>[ <session_group>]*
```

Example:

```
SET_PARAMETER rtsp://172.168.2.2/1234 RTSP/1.0
CSeq: 314
Require: com.comcast.ngod.r2
Content-Type: text/parameters
Content-Length: 40

sessionGroups: SM1.SG1 SM1.SG2 SM1.SG3
```

12 RTSP CONSTANTS

12.1 Introduction

This section defines constant value used in the NGOD implementations of RTSP. These include response status codes, ANNOUNCE codes, and TEARDOWN reason codes.

12.2 Response Status Codes

The following is the complete set of NGOD RTSP response codes. Specific descriptions or clarifications as to how these codes relate to specific NGOD interfaces may be found in the respective interface specifications.

Table 7 – Response Status Codes

Code	Message
200	OK
300	Redirect – Multiple Choices
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
408	Request Time Out
410	Gone
413	Request Entity Too Large
414	Request URI Too Long
415	Unsupported Media Type
451	Invalid Parameter
453	Not Enough Bandwidth
454	Session Not Found
455	Method Not Valid In This State
456	Header Field Not Valid

Code	Message
457	Invalid Range
458	Parameter Is Read-Only
459	Aggregate Operation Not Allowed
460	Only Aggregate Operation Allowed
461	Unsupported Transport
462	Destination Unreachable
499	No such content
500	Internal Server Error
501	Not implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout
505	RTSP Version Not Supported
551	Option Not Supported
599	No available streaming servers

12.3 ANNOUNCE Codes

The following announce codes shall be supported. These are the valid values for the "Notice:" header.

Table 8 – Announce Codes

Code	Message
2101	End-of-Stream Reached – sent when the end-of-stream has been reached
2104	Start-of-Stream Reached – sent when the start-of-stream has been reached, e.g. when in rewind mode
4400	Error Reading Content Data – sent when there is an issue reading the data to be streamed
5200	Server Resources Unavailable – sent when there is an issue with availability of server resources
5401	Downstream Failure - sent when there is a downstream failure, e.g., QAM failure

Code	Message
5402	Client Session Terminated - sent as a notification that for some reason the RTSP client session was terminated by the server
5502	Internal Server Error – sent as a notification that there was an issue with the server
5601	In Band Stream Marker Mismatch
5602	Bandwidth Exceeded Limit

12.4 TEARDOWN Reason Codes

The complete set of NGOD TEARDOWN Reason Codes is as follows.

Table 9 – TEARDOWN Reason Codes

Teardown Reason Code	Description
200	User stop
201	End of stream
202	Beginning of stream
203	Pause timeout
400	Fail to tune
401	Loss of tune
403	RTSP failure
404	Channel failure
405	No RTSP server
406	Trick-play failed
407	Internal ODA issue
408	Unknown
550	Session Time-out

13 TCP CONNECTION BEHAVIOR

13.1 Introduction

This section defines the required behavior of components with respect to TCP connections.

13.2 Establishing the socket

It will be the requirement of the RTSP client to establish the TCP socket and to re-establish it if the socket is ever disconnected. The RTSP client and server will use that socket for messages regarding multiple sessions with no need to establish a separate socket per session.

13.3 Connection timeout

In TCP mode it is required that upon establishing the socket connection a RTSP client send a GET_PARAMETER of connection_timeout to understand the connection timeout value.

Optionally, if a RTSP server does not receive any communication over a given RTSP TCP socket connection for the number of seconds specified by the connection_timeout value, it may close the TCP socket.

14 SESSION TIMEOUT

14.1 Introduction

This section defines behavior with respect to session timeouts. Unless otherwise specified the session timeout mechanism is as defined in RFC 2326.

14.2 Session Timeout Value

It is suggested that the session timeout value be set to three hours.

14.3 Session Timeout Behavior

In the event that an RTSP server detects a session timeout, the following will occur:

- RTSP server detects the session timeout
- RTSP server sends an announce with a notice code of 2102 “Session In Progress”
- The SM sends an announce response “OK” or “Session not found”. Note: this response renews the RTSP server session timer
- The SM may or may not send a teardown

Example:

```
ANNOUNCE * RTSP/1.0
CSeq: 1
Notice: 2102 "Session In Progress" event-date=20050606T102749.258Z npt=
OnDemandSessionId: 4b6727a2-d6b0-11d9-ae2e-112358132134
Require: com.comcast.ngod.s6
Session: 10000104

-----------------

RTSP/1.0 200 OK
CSeq: 1
OnDemandSessionId: 4b6727a2-d6b0-11d9-ae2e-112358132134
Session: 10000104

--- OR ---

RTSP/1.0 454 Session Not Found
CSeq: 1
OnDemandSessionId: 4b6727a2-d6b0-11d9-ae2e-112358132134
Session: 10000104
```

15 MESSAGE TIMEOUT

15.1 Introduction

This section defines a suggested interaction scenario for the timing out of message requests. This timeout mechanism is especially important in environments where the transport is UDP. Section 9.5 of draft-ietf-mmusic-rfc2326bis-07 provides guidance on message timeouts.

15.2 Interaction Scenario

15.2.1 Step 1 – Sender sends RTSP Request

To begin a communication the sender sends a RTSP request. After sending the message the sender starts a timer T.

If T expires prior to reception of a response the sender shall consider the message failed.

If after T has expired the sender receives a RTSP response for the message it shall execute appropriate cleanup business logic.

15.2.2 Step 2 – Receiver receives RTSP Request and responds

The Receiver receives RTSP requests from a sender.

If the Receiver has issues with the format or content of the RTSP request or its parameters, it shall respond to the sender with a RTSP response with “Response Code” of “400”.

Upon reception of a valid RTSP request the Receiver will execute the appropriate logic. When complete, the Receiver will send a RTSP response to the sender with the appropriate parameters.

15.2.3 Step 3 – Sender receives the RTSP Response

The Sender receives the RTSP response from the Receiver.

16 SESSION IDENTIFIERS

Within the NGOD architecture session ID’s will be implemented as follows.

- The client creates a ClientSessionId and sends it on the session setup request in the ClientSessionId RTSP header.
- The Session Manager creates a OnDemandSessionId on reception of a session setup. The OnDemandSessionId is passed to every server-side component that relates to this session in the OnDemandSessionId RTSP header.

- Each individual RTSP server (e.g. ERM, ODRM) creates its own specific RTSP session ID for its session per the RTSP RFC.
- In interactions with the RTSP servers relating to the existing session (e.g. teardown request) both the RTSP session ID and the OnDemandSessionId will be sent.