The Real Time Streaming Protocol (RTSP) Protocol Modules for TTCN-3 Toolset with TITAN, Function Specification

Endre Kulcsár

Version 155 17-CNL 113 588, Rev. A, 2012-06-14

Table of Contents

How to Read This Document

This is the Function Specification for the Real Time Streaming Protocol (RTSP) protocol modules. RTSP protocol modules are developed for the TTCN-3 Toolset with TITAN.

Scope

The purpose of this document is to specify the content of the Real Time Streaming Protocol (RTSP) protocol modules. Basic knowledge of TTCN-3 [2] and TITAN TTCN-3 Test Executor [3] is valuable when reading this document.

General

Protocol modules implement the message structures of the related protocol in a formalized way, using the standard specification language TTCN-3. This allows defining of test data (templates) in the TTCN-3 language [2] and correctly encoding/decoding messages when executing test suites using the TITAN TTCN-3 test environment [3].

NOTE	This version of the protocol module is not compatible with TITAN releases earlier than R8B.

Functional Specification

Protocol Version Implemented

This set of protocol modules implements protocol messages and constants of The Real Time Streaming Protocol (RTSP). The modules are based on RFC 2326 (see [1]) and Interface Description [5].

Implemented Messages

According to [1] both RTSP message types "request" and "response" are implemented. Additionally the message type "erroneous message" is introduced for not decodable messages.

Implemented Methods

All methods specified in Chapter 6.1 of [1] are implemented as follows:

- `"DESCRIBE"`
- `"ANNOUNCE"`
- `"GET_PARAMETER"`
- `"OPTIONS"`

- `"PAUSE"`
- `"PLAY"`
- `"RECORD"`
- `"REDIRECT"`
- `"SETUP"`
- `"SET_PARAMETER"`
- `"TEARDOWN"`

Supported Header Fields

All header field specified in Chapter 12 of [1] is supported as listed below. The fields in parentheses are not listed in Chapter 12 of [1] but listed in subchapters of chapter 12 of [1].

- ***Accept***
- ***Accept-Encoding***
- ***Accept-Language***
- ***Allow***
- ***Authorization***
- ***Bandwidth***
- ***Blocksize***
- ***Cache-Control***
- ***Conference***
- ***Connection***
- ***Content-Base***
- ***Content-Encoding***
- ***Content-Language***
- ***Content-Length***
- ***Content-Location***
- ***Content-Type***
- ***Content-Type***
- ***CSeq***
- ***Date***
- ***Expires***
- ***From***
- ***(Host)***
- ***(If-Match)***
- ***If-Modified-Since***
- ***Last-Modified***

- ***Proxy-Authenticate***
- ***Proxy-Require***
- ***Public***
- ***Range***
- ***Referer***
- ***Require***
- ***Retry-After***
- ***RTP-Info***
- ***Scale***
- ***Session***
- ***Server***
- ***Speed***
- ***(Time-Stamp)***
- ***Transport***
- ***Unsupported***
- ***User-Agent***
- ***(Vary)***
- ***Via***
- ***WWW-Authenticate***

Implemented But Not Specified Header Fields

The list of implemented header fields which are not specified in [1] is as follows. They are used in Ericsson proprietary solutions.

- ***RDTFeatureLevel***
- ***RealChallenge1***
- ***Reconnect***
- ***Rtcp-Interval***
- ***StatsMask***
- ***Vsrc***
- ***x-Real-usestrackid***
- ***x-Vig-Bno***
- ***x-Vig-MSISDN***
- ***x-retransmit***
- ***x-dynamic-rate***

- *x-transport-options*
- *x-prebuffer*

In addition the following headers specified in [5] are also implemented:

X-ActionX-EncodingFilesX-UdpPipe

X-MbmsSync

X-Bandwidth

X-Content

X-Fec

X-UserPlaneDest

X-FluteBitrate

X-TsiX-ContentFdtSendIntervalX-Reporting

Header Field Extensibility

Each header field listed in Supported Header Fields and Implemented But Not Specified Header Fields are available as optional fields having value of characterstring. To provide the extensibility for future development, extension header list is implemented. It is a list of name-value pairs where both names and values are arbitrary charstrings (see Header Implementation).

Header Implementation

According to Supported Header Fields, Implemented But Not Specified Header Fields and Header Field Extensibility, common header implemented for RTSP request and response to support positive and negative test as follows:

```
type set HeaderStruct {
  charstring    accept          optional, //12.1
  charstring    acceptEncoding  optional, //12.2
  charstring    acceptLanguage  optional, //12.3
  charstring    allow           optional, //12.4
  charstring    authorization   optional, //12.5
  charstring    bandwidth       optional, //12.6
  charstring    blocksize       optional, //12.7
  charstring    cacheControl    optional, //12.8
  charstring    conference      optional, //12.9
  charstring    connection      optional, //12.10
  charstring    contentBase     optional, //12.11
  charstring    contentEncoding optional, //12.12
  charstring    contentLanguage optional, //12.13
  charstring    contentLength   optional, //12.14
```

```
  charstring    contentLocation optional, //12.15
  charstring    contentType     optional, //12.16
  charstring    cSeq            optional, //12.17
  charstring    date            optional, //12.18
  charstring    expires         optional, //12.19
  charstring    fromField       optional, //12.20
  charstring    host            optional, //12.21
  charstring    ifMatch         optional, //12.22
  charstring    ifModifiedSince optional, //12.23
  charstring    lastModified    optional, //12.24
  charstring    location        optional, //12.25
  charstring    proxyAuth       optional, //12.26
  charstring    proxyRequire    optional, //12/27
  charstring    publicField     optional, //12.28
  charstring    range           optional, //12.29
  charstring    rdtFeatureLevel optional, //additional
  charstring    realChallenge1  optional, //additional
  charstring    reconnect       optional, //additional
  charstring    referer         optional, //12.30
  charstring    retryAfter      optional, //12.31
  charstring    require         optional, //12.32
  charstring    rtcpInterval    optional, //additional
  charstring    rtpInfo         optional, //12.33
  charstring    scale           optional, //12.34
  charstring    speed           optional, //12.35
  charstring    server          optional, //12.36
  charstring    session         optional, //12.37
  charstring    statsMask       optional, //additional
  charstring    timeStamp       optional, //12.38
  charstring    transport       optional, //12.39
  charstring    unsupported     optional, //12.40
  charstring    userAgent       optional, //12.41
  charstring    vary            optional, //12.42
  charstring    via             optional, //12.43
  charstring    vsrc            optional, //additional
  charstring    wwwAuth         optional, //12.44
  charstring    xRealUsestrackid optional,//additional
  charstring    xVigBno         optional, //additional
  charstring    xVigMsisdn      optional, //additional
  charstring    xRetransmit     optional, //additional
  charstring    xDynamicRate    optional, //additional
  charstring    xTransportOptions optional, //additional
  charstring    xPrebuffer      optional, //additional
  charstring    xAction         optional, // RTSPx
  charstring    xEncodingFiles  optional, // RTSPx
  charstring    xUdpPipe        optional, // RTSPx
  charstring    xMbmsSync       optional, // RTSPx
  charstring    xBandwidth      optional, // RTSPx
  charstring    xContent        optional, // RTSPx
  charstring    xFec            optional, // RTSPx
  charstring    xUserPlaneDest  optional, // RTSPx
```

```
  charstring    xFluteBitrate   optional, // RTSPx
  charstring    xTsi            optional, // RTSPx
  charstring    xContentFdtSendInterval optional,//RTSPx
  charstring    xReporting      optional, // RTSPx

//extensionHeaders:
  HeaderLines extensionHeaders optional
}
Where

type record HeaderLine {
  charstring header_name,
  charstring header_value
};
```

Protocol Modifications/Deviations

Relaxed Conditions

There is no constraint between received and sent messages. The constraints should be implemented in the user's test program.

URI in the request line is a simple charstring. Its correctness is not checked.

Reason Code can be any integer in the Status Line

Reason Phrase can be any charstring. There is no constraint between them for test purposes.

Restrictions

Octetestrings supported only.

Utf8text not supported.

The encoded message is octetstring. Within it the request line, the status line and the header shall be convertible for charstring, the body can be any octetstring.

Encoding/Decoding and Other Related Functions

This product also contains encoding/decoding functions that assure correct encoding of messages when sent from TITAN and correct decoding of messages when received by TITAN. Implemented encoding/decoding functions and the extra length calculator function are:

Name	Type of formal parameters	Type of return value	Description
`enc_PDU_RTSP`	in `PDU_RTSP` msg, in Boolean automaticContentLengthCalc:=true	Octetstring	Encodes the RTSP PDU into octetestring
`dec_PDU_RTSP`	in octetstring stream, inout `PDU_RTSP` msg, in boolean debugging := `tsp_RTSP_debugging`	integer	Decodes the message in octestring into `PDU_RTSP`
`f_RTSP_getMsgLen`	In octetstring stream	integer	Calculates the length of the message stream from the beginning of the message (especially from the field Content-Length).

Encoding/Decoding Logic

According to RFC2326, the following rules are followed in the decoding and encoding processes:

- The lines are finished by "\r\n". Message lines finished only by "\n" can be tolerated. The degree of tolerance is `ERROR`, `WARNING`, `WARNING_ONCE` or `ACCEPT`.
- If the message begins with "RTSP/" it is an RTSP response, otherwise it is an RTSP request.
- The RTSP message consists of **three** parts:
  1. The **first line** of the message **is the first part** of the message. It is the Status Line for message type of request otherwise the first line is the Request Line. They are split up according to RFC 2326.
  2. The **second part** of the message is **the header**. It consists of header fields. Details can be found in Supported Header Fields - Header Implementation.

     The header finished by an additional "\r\n" (i.e a sequence "r" is the end of the header).
  3. The **third field** of the message is **the body**. It can be any octetstring.
- The header field ***Content-Length*** is present (with correct value) in the encoded message if and only if the body length is greater than `"zero"` and the `automaticContentLengthCalc` parameter of the encoding function is `true`. If this parameter is set `false` then the ***Content-Length*** header field is encoded as it is in the `HeaderStruct` and its value does not depend on the length of the body so it's suitable for making negative tests.

Terminology

TITAN TTCN-3 Test Executor (see [3]).

Abbreviations

ETSI

European Telecommunications Standards Institute

IETF

Internet Engineering Task Force

RFC

Request for Comments

RTSP

Real Time Streaming Protocol

TTCN-3

Testing and Test Control Notation version 3

References

[1] IETF RFC 2326
Real Time Streaming Protocol (RTSP)

[2] ETSI ES 201 873-1 v.3.2.1 (02/2007)
The Testing and Test Control Notation version 3. Part 1: Core Language

[3] Programmer's Technical Reference for the TITAN TTCN-3 Test Executor

[4] CBC/XL-12:0167 Uen
Interface Description, RTSPx