Network Working Group
Request for Comments: 4145
Category: Standards Track

D. Yon
Tactical Software, LLC
G. Camarillo
Ericsson

September 2005

# TCP-Based Media Transport in the Session Description Protocol (SDP)

## Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

## Copyright Notice



## Abstract

This document describes how to express media transport over TCP using the Session Description Protocol (SDP). It defines the SDP 'TCP' protocol identifier, the SDP 'setup' attribute, which describes the connection setup procedure, and the SDP 'connection' attribute, which handles connection reestablishment.

Table of Contents

## 1. Introduction

The Session Description Protocol [4] provides a general-purpose format for describing multimedia sessions in announcements or invitations. SDP uses an entirely textual data format (the US-ASCII subset of UTF-8 [11]) to maximize portability among transports. SDP does not define a protocol; it defines the syntax to describe a multimedia session with sufficient information to participate in that session. Session descriptions may be sent using arbitrary existing application protocols for transport (e.g., SAP [9], SIP [10], RTSP [6], email, HTTP [8], etc.).

SDP [4] defines two protocol identifiers: RTP/AVP and UDP, both of which represent unreliable, connectionless protocols. While these transports are appropriate choices for multimedia streams, there are applications for which TCP is more appropriate. This document defines a new protocol identifier, 'TCP', to describe TCP connections in SDP.

TCP introduces two new factors when describing a session: how and when should endpoints perform the TCP connection setup procedure. This document defines two new attributes to describe TCP connection setups: 'setup' and 'connection'.

## 2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [3], and they indicate requirement levels for compliant implementations.

## 3. Protocol Identifier

The following is the ABNF for an 'm' line, as specified by RFC 2327 [4].

```
media-field =         "m=" media space port ["/" integer]
                      space proto 1*(space fmt) CRLF
```

This document defines a new value for the proto field: 'TCP'.

The 'TCP' protocol identifier is similar to the 'UDP' protocol identifier in that it only describes the transport protocol, and not the upper-layer protocol. An 'm' line that specifies 'TCP' MUST

further qualify the application-layer protocol using an fmt identifier. Media described using an 'm' line containing the 'TCP' protocol identifier are carried using TCP [1].

## 4. Setup Attribute

The 'setup' attribute indicates which of the end points should initiate the TCP connection establishment (i.e., send the initial TCP SYN). The 'setup' attribute is charset-independent and can be a session-level or a media-level attribute. The following is the ABNF of the 'setup' attribute:

```
setup-attr           =  "a=setup:" role
role                 =  "active" / "passive" / "actpass"
                        / "holdconn"
```

'active': The endpoint will initiate an outgoing connection.

'passive': The endpoint will accept an incoming connection.

'actpass': The endpoint is willing to accept an incoming connection or to initiate an outgoing connection.

'holdconn': The endpoint does not want the connection to be established for the time being.

### 4.1. The Setup Attribute in the Offer/Answer Model

The offer/answer model, defined in RFC 3264 [5], provides endpoints with a means to obtain shared view of a session. Some session parameters are negotiated (e.g., codecs to use), while others are simply communicated from one endpoint to the other (e.g., IP addresses). The value of the 'setup' attribute falls into the first category. That is, both endpoints negotiate its value using the offer/answer model.

The negotiation of the value of the 'setup' attribute takes places as follows. The offerer states which role or roles it is willing to perform; and the answerer, taking the offerer's willingness into consideration, chooses which roles both endpoints will actually perform during connection establishment. The following are the values that the 'setup' attribute can take in an offer/answer exchange:

| Offer | Answer |
|---|---|
| active | passive / holdconn |
| passive | active / holdconn |
| actpass | active / passive / holdconn |
| holdconn | holdconn |

The active endpoint SHOULD initiate a connection to the port number on the 'm' line of the other endpoint. The port number on its own 'm' line is irrelevant, and the opposite endpoint MUST NOT attempt to initiate a connection to the port number specified there. Nevertheless, since the 'm' line must contain a valid port number, the endpoint using the value 'active' SHOULD specify a port number of 9 (the discard port) on its 'm' line. The endpoint MUST NOT specify a port number of zero, except to denote an 'm' line that has been or is being refused.

The passive endpoint SHOULD be ready to accept a connection on the port number specified in the 'm' line.

A value of 'actpass' indicates that the offerer can either initiate a connection to the port number on the 'm' line in the answer, or accept a connection on the port number specified in the 'm' line in the offer. That is, the offerer has no preference as to whether it accepts or initiates the connection and, so, is letting the answerer choose.

A value of 'holdconn' indicates that the connection should not be established for the time being.

The default value of the setup attribute in an offer/answer exchange is 'active' in the offer and 'passive' in the answer.

## 5. The Connection Attribute

The preceding description of the 'setup' attribute is placed in the context of using SDP to initiate a session. Still, SDP may be exchanged between endpoints at various stages of a session to accomplish tasks such as terminating a session, redirecting media to a new endpoint, or renegotiating the media parameters for a session. After the initial session has been established, it may be ambiguous whether a subsequent SDP exchange represents a confirmation that the endpoint is to continue using the current TCP connection unchanged, or is a request to make a new TCP connection. The media-level 'connection' attribute, which is charset-independent, is used to disambiguate these two scenarios. The following is the ABNF of the connection attribute:

```
connection-attr        = "a=connection:" conn-value
conn-value             = "new" / "existing"
```

## 5.1. Offerer Behaviour

Offerers and answerers use the 'connection' attribute to decide whether a new transport connection needs to be established or, on the other hand, the existing TCP connection should still be used. When an offerer generates an 'm' line that uses TCP, it SHOULD provide a connection attribute for the 'm' line unless the application using the 'm' line has other means to deal with connection reestablishment.

After the initial offer/answer exchange, any of the endpoints can generate a new offer to change some characteristics of the session (e.g., the direction attribute). If such an offerer wants to continue using the previously-established transport-layer connection for the 'm' line, the offerer MUST use a connection value of 'existing' for the 'm' line. If, on the other hand, the offerer wants to establish a new transport-layer connection for the 'm' line, it MUST use a connection value of 'new'.

> Note that, according to the rules in this section, an offer that changes the transport address (IP address or port number) of an 'm' line will have a connection value of 'new'. Similarly, the 'connection' attribute in an initial offer (i.e., no transport connection has been established yet) takes the value of 'new'.

The 'connection' value resulting from an offer/answer exchange is the 'connection' value in the answer. If the 'connection' value in the answer is 'new', the end-points SHOULD establish a new connection. If the connection value in the answer is 'existing', the end-points SHOULD continue using the exiting connection.

Taking into consideration the rules in Section 5.2, the following are the values that the 'connection' attribute can take in an offer/answer exchange:

| Offer | Answer |
|---|---|
| new | new |
| existing | existing / new |

If the connection value resulting from an offer/answer exchange is 'existing', the end-points continue using the existing connection. Consequently, the port numbers, IP addresses, and 'setup' attributes negotiated in the offer/answer exchange are ignored because there is no need to establish a new connection.

The previous rule implies that an offerer generating an offer with a connection value of 'existing' and a setup value of 'passive' needs to be ready (i.e., needs to allocate resources) to receive a connection request from the answerer just in case the answerer chooses a connection value of 'new' for the answer. However, if the answerer uses a connection value of 'existing' in the answer, the offerer would need to deallocate the previously allocated resources that were never used because no connection request was received.

To avoid allocating resources unnecessarily, offerers using a connection value of 'existing' in their offers may choose to use a setup value of 'holdconn'. Nevertheless, offerers using this strategy should be aware that if the answerer chooses a connection value of 'new', a new offer/answer exchange (typically initiated by the previous offerer) with setup value different than 'holdconn' will be needed to establish the new connection. This may, of course, cause delays in the application using the TCP connection.

The default value of the connection attribute in both offers and answers is 'new'.

## 5.2. Answerer Behaviour

The connection value for an 'm' line is negotiated using the offer/answer model. The resulting connection value after an offer/answer exchange is the connection value in the answer. If the connection value in the offer is 'new', the answerer MUST also use a value of 'new' in the answer. If the connection value in the offer is 'existing', the answerer uses a value of 'existing' in the answer if it wishes to continue using the existing connection and a value of 'new' if it wants a new connection to be established.

> In some scenarios where third party call control [12] is used, an endpoint may receive an initial offer with a connection value of 'existing'. Following the previous rules, such an answerer would use a connection value of 'new' in the answer.

If the connection value for an 'm' line resulting from an offer/answer exchange is 'new', the endpoints SHOULD establish a new TCP connection as indicated by the 'setup' attribute. If a previous TCP connection is still up, the endpoints SHOULD close it as soon as the offer/answer exchange is completed. It is up to the application to ensure proper data synchronization between the two TCP connections.

If the connection value for an 'm' line resulting from an offer/answer exchange is 'existing', the endpoints SHOULD continue using the existing TCP connection.

## 6. Connection Management

This section addresses connection establishment, connection reestablishment, and connection termination.

### 6.1. Connection Establishment

An endpoint that according to an offer/answer exchange is supposed to initiate a new TCP connection SHOULD initiate it as soon as it is able to, even if the endpoint does not intend to immediately begin sending media to the remote endpoint. This allows media to flow from the remote endpoint if needed.

> Note that some endpoints need to wait for some event to happen before being able to establish the connection. For example, a wireless terminal may need to set up a radio bearer before being able to initiate a TCP connection.

### 6.2. Connection Reestablishment

If an endpoint determines that the TCP for an 'm' line has been closed and should be reestablished, it SHOULD perform a new offer/answer exchange using a connection value of 'new' for this 'm' line.

> Note that the SDP direction attribute (e.g., 'a=sendonly') deals with the media sent over the TCP connection, but has no impact on the TCP connection itself.

### 6.3. Connection Termination

Typically, endpoints do not close the TCP connection until the session has expired, been explicitly terminated, or a new connection value has been provided for the 'm' line. Additionally, specific applications can describe further scenarios where an end-point may close a given TCP connection (e.g., whenever a connection is in the half-close state). As soon as an end-point notices that it needs to terminate a TCP connection, it SHOULD do so.

In any case, individual applications may provide further considerations on how to achieve a graceful connection termination. For example, a file application using TCP to receive a FIN from the remote endpoint may need to finish the ongoing transmission of a file before sending its own FIN.

## 7. Examples

The following examples show the most common usage of the 'setup' attribute combined with TCP-based media descriptions. For the purpose of brevity, the main portion of the session description is omitted in the examples, which only show 'm' lines and their attributes (including 'c' lines).

### 7.1. Passive/Active

An offerer at 192.0.2.2 signals its availability for a T.38 fax session at port 54111:

```
m=image 54111 TCP t38
c=IN IP4 192.0.2.2
a=setup:passive
a=connection:new
```

An answerer at 192.0.2.1 receiving this offer responds with the following answer:

```
m=image 9 TCP t38
c=IN IP4 192.0.2.1
a=setup:active
a=connection:new
```

The endpoint at 192.0.2.1 then initiates the TCP connection to port 54111 at 192.0.2.2.

### 7.2. Actpass/Passive

In another example, an offerer at 192.0.2.2 signals its availability for a T.38 fax session at TCP port 54111. Additionally, this offerer is also willing to set up the media stream by initiating the TCP connection:

```
m=image 54111 TCP t38
c=IN IP4 192.0.2.2
a=setup:actpass
a=connection:new
```

The endpoint at 192.0.2.1 responds with the following description:

```
m=image 54321 TCP t38
c=IN IP4 192.0.2.1
a=setup:passive
a=connection:new
```

This will cause the offerer (at 192.0.2.2) to initiate a connection to port 54321 at 192.0.2.1.

### 7.3. Existing Connection Reuse

Subsequent to the exchange in Section 7.2, another offer/answer exchange is initiated in the opposite direction. The endpoint at 192.0.2.1 wishes to continue using the existing connection:

```
m=image 54321 TCP t38
c=IN IP4 192.0.2.1
a=setup:passive
a=connection:existing
```

The endpoint at 192.0.2.2 also wishes to use the existing connection and responds with the following description:

```
m=image 9 TCP t38
c=IN IP4 192.0.2.2
a=setup:active
a=connection:existing
```

The existing connection from 192.0.2.2 to 192.0.2.1 will be reused.

Note that the endpoint at 192.0.2.2 uses 'setup:active' in response to the offer of 'setup:passive', and uses port 9 because it is active.

### 7.4. Existing Connection Refusal

Subsequent to the exchange in Section 7.3, another offer/answer exchange is initiated by the endpoint at 192.0.2.2, again wishing to reuse the existing connection:

```
m=image 54111 TCP t38
c=IN IP4 192.0.2.2
a=setup:passive
a=connection:existing
```

However, this time the answerer is unaware of the old connection and thus wishes to establish a new one. (This could be the result of a transfer via third-party call control.) It is unable to act in the 'passive' mode and thus responds as 'active':

```
m=image 9 TCP t38
c=IN IP4 192.0.2.3
a=setup:active
a=connection:new
```

The endpoint at 192.0.2.3 then initiates the TCP connection to port 54111 at 192.0.2.2, and the endpoint at 192.0.2.2 closes the old connection.

Note that the endpoint at 192.0.2.2, while using a connection value of 'existing', has used a setup value of 'passive'. Had it not done this and instead used a setup value of 'holdconn' (probably to avoid allocating resources as described in Section 5.1), a new offer/answer exchange would have been needed in order to establish the new connection.

## 8. Other Connection-Oriented Transport Protocols

This document specifies how to describe TCP-based media streams using SDP. Still, some of the attributes defined here could possibly be used to describe media streams based on other connection-oriented transport protocols as well. This section provides advice to authors of specifications of SDP extensions that deal with connection-oriented transport protocols other than TCP.

It is recommended that documents defining new SDP protocol identifiers that involve extra protocol layers between TCP and the media itself (e.g., TLS [7] over TCP) start with the string 'TCP/' (e.g., 'TCP/TLS').

The 'setup' and the 'connection' attributes are specified in Section 4 and Section 5 respectively. While both attributes are applicable to 'm' lines that use the 'TCP' protocol identifier, they are general enough to be reused in 'm' lines with other connection-oriented transport protocols. Therefore, it is recommended that the 'setup' and 'connection' attributes are reused, as long as it is possible, for new proto values associated with connection-oriented transport protocols.

Section 6 deals with TCP connection management. It should be noted that while in TCP both end-points need to close a connection, other connection-oriented transport protocols may not have the concept of half-close connections. In such a case, a connection would be terminated as soon as one of the end-points closed it, making it unnecessary for the other end-point to perform any further action to terminate the connection. So, specifications dealing with such transport protocols may need to specify slightly different procedures regarding connection termination.

## 9. Security Considerations

See RFC 2327 [4] for security and other considerations specific to the Session Description Protocol in general.

An attacker may attempt to modify the values of the connection and setup attributes in order to have endpoints reestablish connections unnecessarily or to keep them from establishing a connection. So, it is strongly RECOMMENDED that integrity protection be applied to the SDP session descriptions. For session descriptions carried in SIP [10], S/MIME is the natural choice to provide such end-to-end integrity protection, as described in RFC 3261 [10]. Other applications MAY use a different form of integrity protection.

## 10. IANA Considerations

This document defines two session- and media-level SDP attributes: setup and connection. Their formats are defined in Section 4 and Section 5, respectively. These two attributes should be registered by the IANA under "Session Description Protocol (SDP) Parameters" under "att-field (both session and media level)".

This document defines a proto value: TCP. Its format is defined in Section 3. This proto value should be registered by the IANA under "Session Description Protocol (SDP) Parameters" under "proto".

The SDP specification, RFC2327, states that specifications defining new proto values, like the TCP proto value defined in this RFC, must define the rules by which their media format (fmt) namespace is managed. For the TCP protocol, new formats SHOULD have an associated MIME registration. Use of an existing MIME subtype for the format is encouraged. If no MIME subtype exists, it is RECOMMENDED that a suitable one is registered through the IETF process [2] by production of, or reference to, a standards-track RFC that defines the transport protocol for the format.

## 11. Acknowledgements

Jonathan Rosenberg, Rohan Mahy, Anders Kristensen, Joerg Ott, Paul Kyzivat, Robert Fairlie-Cuninghame, Colin Perkins, and Christer Holmberg provided valuable insights and contributions.

## 12. References

### 12.1. Normative References

[1] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[2] Freed, N., Klensin, J., and J. Postel, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", BCP 13, RFC 2048, November 1996.

[3] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[4] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

[5] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

### 12.2. Informative References

[6] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[7] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[8] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[9] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.

[10] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[11] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[12] Rosenberg, J., Peterson, J., Schulzrinne, H., and G. Camarillo, "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)", BCP 85, RFC 3725, April 2004.

## Authors' Addresses

David Yon
Tactical Software, LLC
1750 Elm St., Suite 803
Manchester, NH 03104
USA

EMail: yon-comedia@rfdsoftware.com

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: Gonzalo.Camarillo@ericsson.com

## Full Copyright Statement



## Intellectual Property



## Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.