[MS-SDP]:
Session Description Protocol (SDP) Extensions

Revision Summary

Date	Revision History	Revision Class	Comments
05/11/2007	0.1		MCPP Milestone 4 Initial Availability
08/10/2007	0.2	Minor	Updated the technical content.
09/28/2007	0.3	Minor	Updated the technical content.
10/23/2007	0.4	Minor	Updated the technical content.
11/30/2007	0.5	Minor	Updated the technical content.

Date	Revision History	Revision Class	Comments
01/25/2008	0.5.1	Editorial	Revised and edited the technical content.

Table of Contents

1 Introduction

This document describes a Microsoft extension protocol, Session Description Protocol (SDP) Extensions. The base protocol, Session Description Protocol (SDP), is specified in [RFC4566]. This document describes the **session description** used to negotiate **instant messaging**, **audio/video**, and **data collaboration sessions** noting the extensions used. This document also describes how encryption for audio/video and data collaboration sessions is negotiated.

Protecting data against security threats related to the privacy of RTC media communications between **clients** is of paramount importance. Microsoft has extended the Session Description Protocol to meet this challenge by providing encryption of data collaboration (DC) and audio/visual (A/V).<1>

This encryption functionality is for the **Session Initiation Protocol (SIP)** (as specified in [RFC3261]) service provider only and does not extend to other kinds of traffic. Microsoft strongly recommends that these extensions be used with Transport Layer Security (TLS) to protect the encryption key when it is passed in the SIP/SDP signaling.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

Client
Server

The following terms are specific to this document:

200 OK: A **response** to indicate that the **request** has succeeded.

Audio/Video (AV) Session: A **session** involving exchange of audio and video data between participants in real time.

Data Collaboration (DC) Session: A **session** involving sharing of applications and/or whiteboard between participants in real time.

Data Encryption Standard (DES): An encryption standard that specifies a FIPS approved cryptographic algorithm as specified in [FIPS140].

Instant Messaging (IM) Session: A **session** involving exchange of text-based instant messages between participants in real time.

INVITE: An **SIP Method** used to invite a user or a service to participate in a **session**.

Message Digest 5 (MD5): A hashing algorithm as specified in [RFC1321] that can also be used for encryption key generation.

Real-Time Transport Protocol (RTP): A network protocol as specified in [RFC3550] that provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio and video.

Real-Time Transport Control Protocol (RTCP): A network protocol as specified in [RFC3550] that allows monitoring of the **RTP** data delivery in a scalable manner and provides minimal control and identification functionality.

Session: A multimedia **session** is a set of multimedia senders and receivers and the data streams flowing from senders to receivers. A multimedia conference is an example of a multimedia **session**.

Session Description: A well-defined format for conveying sufficient information to discover and participate in a multimedia **session**.

Session Initiation Protocol (SIP): An application-layer control (signaling) protocol for creating, modifying, and terminating **sessions** with one or more participants. **SIP** is specified in [RFC3261].

SIP Method: The method is the primary function that an **SIP request** is meant to invoke on a **server**. The method is carried in the **request** message itself. Example methods are **INVITE** and BYE.

SIP Request (Request): An **SIP** message sent from a **client** to a **server** for the purpose of invoking a particular operation.

SIP Response (Response): An **SIP** message sent from a **server** to a **client**, indicating the status of a **request** sent from the **client** to the **server**.

T.120: An ITU standard for real-time data conferencing, including application sharing and whiteboarding.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[FIPS46-3] National Institute of Standards and Technology, "Federal Information Processing Standards Publication 46-3: Data Encryption Standard (DES)", October 1999, http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[FIPS140] National Institute of Standards and Technology, "Federal Information Processing Standards Publication 140-2: Security Requirements for Cryptographic Modules", December 2002, http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MS-SIP] Microsoft Corporation, "Session Initiation Protocol Extensions", August 2007.

[RFC1321] Rivest, R. "The MD5 Message-Digest Algorithm", RFC 1321, April 1992, http://www.ietf.org/rfc/rfc1321.txt

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[RFC3261] Rosenberg, J., et al., "SIP: Session Initiation Protocol", RFC 3261, June 2002, http://www.ietf.org/rfc/rfc3261.txt

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V., "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, July 2003, http://www.ietf.org/rfc/rfc3550.txt

[RFC3605] Huitema, C., "Real Time Control Protocol (RTCP) Attribute in Session Description Protocol (SDP)", RFC 3605, October 2003, http://www.ietf.org/rfc/rfc3605.txt

[RFC3611] Friedman, T., Ed., Caceres, R., Ed., and Clark, A., Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003, http://www.ietf.org/rfc/rfc3611.txt

[RFC4566] Handley, M., Jacobson, V., and Perkins, C., "SDP: Session Description Protocol", RFC 4566, July 2006, http://www.ietf.org/rfc/rfc4566.txt

1.2.2 Informative References

There are no informative references for Session Description Protocol (SDP) Extensions.

1.3 Protocol Overview (Synopsis)

The Session Description Protocol (SDP), as specified in [RFC4566], describes multimedia **sessions** for a variety of purposes associated with sessions.

Only the Microsoft extensions to this protocol are documented in this document. SDP Extensions define additional SDP primitives to negotiate various types of sessions and also to negotiate encryption. These extensions include:

- Extensions to negotiate data collaboration sessions (application sharing and whiteboarding): SDP Extensions defines a new value for the **fmt** field of **msdata** to indicate data collaboration. The **msdata** field can have a subfield **wb** to indicate whiteboarding, or a subfield **appsharing** to indicate application sharing.
- Extensions to negotiate an instant messaging session: SDP Extensions defines a new media name x-ms-message used to indicate instant messaging media.
- Extensions to negotiate audio/video encryption: SDP Extensions defines a new SDP attribute a=encryption that can be used to negotiate whether encryption is required, optional, or rejected for an audio or video media stream. This attribute is used in conjunction with k= SDP field, which can be used to carry the encryption key if one is needed.
- Extensions to negotiate data collaboration encryption: SDP Extensions that support encryption for data collaboration are similar to the extensions for negotiating audio/video encryption. The a=encryption attribute is used to indicate whether encryption is required, optional, or rejected. However, no k= field is used by SDP Extensions for data collaboration encryption, because the key is exchanged by a separate mechanism using the **T.120** stream for data collaboration.

These extensions are described in detail in section 3.

1.4 Relationship to Other Protocols

Session Description Protocol (SDP) Extensions depends on SDP, as specified in [RFC4566], and upon the Session Initiation Protocol (SIP), as specified in [RFC3261]. SDP Extensions defines additional SDP primitives needed to negotiate encryption parameters for data collaboration and audio/video sessions.

1.5 Prerequisites/Preconditions

This protocol assumes that both clients support SDP [RFC4566] and Session Initiation Protocol (SIP) [RFC3261].

1.6 Applicability Statement

Session Description Protocol (SDP) Extensions allows negotiation of whether the media should be encrypted and if so, what encryption key should be used.

The encryption negotiation enhancements to SDP may be used when the application wants to encrypt audio/video and data collaboration media on the wire. These enhancements SHOULD be used in conjunction with Transport Layer Security (TLS) to protect the encryption key if the key is passed in the SIP/SDP signaling.

1.7 Versioning and Capability Negotiation

Session Description Protocol (SDP) Extensions does not have protocol versioning. Instead, explicit capability negotiation is performed as specified in this section. The encryption negotiation enhancements to SDP may be used when the application wants to encrypt audio/video and data collaboration media on the wire. These enhancements SHOULD be used in conjunction with Transport Layer Security (TLS) to protect the encryption key if the encryption key is passed in the SIP/SDP signaling.

1.7.1 Encryption Capability Negotiation

Encryption for audio/video can be negotiated by using the SDP Extensions attribute "**a=encryption**" and the SDP field "**k=**". The "**k=**" field is specified in [RFC4566] and is used to carry the encryption key. The following table illustrates how a client can negotiate its support for audio/video encryption:

Protocol Element	Description
Include the **a=encryption:required** attribute and a **k=** field.	Supports encryption and requires it due to local policy.
Include the **a=encryption:optional** attribute and a **k=** field.	Supports encryption but does not require it.
Include the **a=encryption:rejected** attribute.	Encryption is not allowed due to local policy.
No **a=encryption** attribute.	Does not support encryption.

Encryption for a data collaboration can be negotiated using the a=encryption attribute. Note that no k= field is needed in this case because key negotiation for data collaboration is handled inside the T.120 stream. The following table illustrates how a client can negotiate its support for data collaboration encryption:

Protocol Element	Description
Include the **a=encryption:required** attribute.	Supports encryption and requires it due to local policy.
Include the **a=encryption:optional** attribute.	Supports encryption but does not require it.
Include the **a=encryption:rejected** attribute.	Encryption is not allowed due to local policy.
No **a=encryption** attribute.	Supports encryption but does not require it.

1.7.2 Interaction between Audio/Video and Data Encryption Negotiation

Data collaboration and audio/video encryption are negotiated separately and controlled through separate registry settings at the client.

Even though data collaboration and audio/video encryption are negotiated separately and controlled through separate registry settings at the client, it is possible that in a session involving both DC and audio/video, both parties may be able to negotiate encryption for one type but not for the other.

The following example shows how different encryption settings could be requested for audio/video and data collaboration.

```
m=audio 49170 RTP/AVP 0
a=encryption:required
k=base64:bhdsfsd78f7dssdfssfsd7sdfssa

m=video 49171 RTP/AVP 25
a=encryption:required
k=base64:bhdsfsd78f7dssdfssfsd7sdfssa

m=application 1503 tcp msdata
a=encryption:rejected
```

In this situation, the session should fail. Encryption must be successfully negotiated for all media types in a session for that session to succeed.

For reINVITEs, failure to negotiate encryption does not result in failure of the existing session. As with any 4xx or 5xx **response** to a reINVITE, the client reverts to the previous session description, including any encryption parameters that had been negotiated at that time.

1.7.3 Interaction between Audio and Video Encryption Negotiation

Since the audio and video components of an audio/video call are represented as separate streams in SDP, it is technically possible for the audio and video components to request different encryption settings.

The following example shows how different encryption settings could be requested for audio and video components.

```
m=audio 49170 RTP/AVP 0
a=encryption:required
k=base64:bhdsfsd78f7dssdfssfsd7sdfssa

m=video 49171 RTP/AVP 25
a=encryption:optional
k=base64:bhdsfsd78f7dssdfssfsd7sdfssa
```

While this is technically possible, it is not supported. The audio and video streams must specify the same encryption attribute.

If a conflict occurs, a 488 error response with an appropriate Warning: header should be returned, as shown in the following example:

```
SIP/2.0 488 Encryption Levels not compatible

Warning: 308 ms.com "Encryption Levels not compatible"
```

1.7.4 Renegotiation of Encryption

It is theoretically possible to renegotiate the encryption level in a session by sending a reINVITE message with a different encryption level than that negotiated at the time of session creation. However, this is not supported by Session Initiation Protocol (SIP) Extensions, and any such reINVITE SHOULD be rejected.

1.8 Vendor-Extensible Fields

There are no vendor-extensible fields specific to Session Description Protocol (SDP) Extensions. Session Initiation Protocol (SIP) [RFC3261] and SDP [RFC4566] may be used by vendors as needed.

1.9 Standards Assignments

The attributes and fields defined by SDP Extensions have NOT been registered with IANA. SDP by itself is specified in [RFC4566].

2 Messages

The following sections specify how Session Description Protocol (SDP) Extensions messages are transported and message syntax.

2.1 Transport

Microsoft extensions to SDP do not introduce a new transport to exchange messages. SDP messages are carried inside SIP messages. The content type MUST be specified as application/sdp for SIP messages carrying SDP, and the session description should be included in the body of the SIP message.

SIP messages MAY be transported over UDP, TCP, or TLS. Microsoft strongly recommends that these enhancements SHOULD be used in conjunction with Transport Layer Security (TLS) to protect the encryption key if the key is passed in the SIP/SDP signaling.

2.2 Message Syntax

Session Description Protocol (SDP) Extensions does not introduce a new message format and relies on SIP and SDP message formats. SIP message format is specified in [RFC3261] section 7. The SDP format is specified in [RFC4566]. The extensions are defined as custom SDP fields and attributes.

The following example shows a standard SDP packet and the extensions.

```
v=0
o=username 0 0 IN IP4 157.59.134.89
s=session
c=IN IP4 157.59.134.89
b=CT:1000
t=0 0
m=audio 56472 RTP/AVP 97 111 112 6 0 8 4 5 3 101
a=rtpmap:97 red/8000
a=rtpmap:111 SIREN/16000
a=fmtp:111 bitrate=16000
a=rtpmap:112 G7221/16000
a=fmtp:112 bitrate=24000
a=rtpmap:6 DVI4/16000
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:4 G723/8000
a=rtpmap:5 DVI4/8000
a=rtpmap:3 GSM/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=encryption:required
k=base64:bhdsfsd78f7dssdfssfsd7sdfssa
m=video 33792 RTP/AVP 34 31
a=rtcp:33801
a=rtpmap:34 H263/90000
a=rtpmap:31 H261/90000
a=encryption:required
k=base64:bhdsfsd78f7dssdfssfsd7sdfssa
m=application 1503 tcp msdata
a=sendonly
a=encryption:required
```

For all media description blocks, notice the a=encryption:required attribute, which is an SDP Extensions extension to indicate that encryption is required for these media. Other possible values for the a=encryption attribute are a=encryption:optional and a=encryption:rejected. The a=encryption extension is described in detail in section 3.

Another SDP Extensions extension shown in the example above is the msdata for the **fmt** field for the m=application media type. Two parameter values are defined for the msdata: *wb* and *appsharing*. The *wb* parameter is used to indicate whiteboarding, as specified in section 3.1.5.2.2, whereas *appsharing* is used to indicate application sharing as specified in section 3.1.5.2.1.

The example below shows SDP for an instant messaging session.

```
v=0
o=- 0 0 IN IP4 172.27.174.212
s=session
c=IN IP4 172.27.174.212
t=0 0
m=x-ms-message 5060 sip null
```

Notice the m=x-ms-message line which shows the SDP Extensions extension for a new media name x-ms-message used to negotiate instant messaging.

Apart from the noted extensions in the two examples, the rest of the session description is standard SDP.

3 Protocol Details

3.1 User Agent Details

SDP is used to negotiate session description between user agents. There is no **server** role that is relevant for Session Description Protocol (SDP) Extensions.

3.1.1 Abstract Data Model

Not applicable.

3.1.2 Timers

There are no timers required by Session Description Protocol (SDP) Extensions beyond what are required by Session Initiation Protocol Extensions, as specified by [MS-SIP].

3.1.3 Initialization

Not applicable.

3.1.4 Higher-Layer Triggered Events

There are no new higher-layer events beyond what would be triggered for SIP sessions. The higher-layer should receive an event for an incoming session. For SDP Extensions this event should also include the encryption setting requested by the caller. When a session is established successfully, the higher layer should receive an event indicating this success. If session negotiation fails, the higher layer should receive an event indicating the failure. If the session negotiation failed due to incompatible encryption settings, the event should indicate the incompatibility.

3.1.5 Message Processing Events and Sequencing Rules

There are no new message processing events or sequencing rules defined by SDP Extensions beyond those defined by SIP/SDP.

3.1.5.1 Audio and Video

The Session Description Protocol (SDP) for an audio/video session is standard (according to RFC) with one exception: negotiation of encryption of the audio/video session uses additional extensions, which are specified in section 3.1.5.5.

3.1.5.2 Data Collaboration

3.1.5.2.1 Application Sharing

An application-sharing session is identified by the following media information in SDP:

```
m=application 1503 tcp msdata
a=encryption:required
a=fmtp:msdata appsharing
```

The **media** field extension conveys capability for application sharing or use of a whiteboard. The value "tcp" is valid for the **proto** subfield, and the value "msdata" is valid for the **fmt** subfield. a=fmtp:msdata appsharing indicates that the format being negotiated is application sharing.

The port (1503) is the standard port used for T.120 and is implemented by the underlying NetMeeting client. See section 3.1.5.4 for additional details about encryption of application sharing.

3.1.5.2.2 Whiteboard

A whiteboard session is identified by the following media information in SDP:

```
m=application 1503 tcp msdata
a=encryption:required
a=fmtp:msdata wb
```

The port (1503) is the standard port used for T.120 and is implemented by the underlying NetMeeting client. a=fmtp:msdata wb indicates that the format being negotiated is whiteboarding.

3.1.5.3 Instant Messaging

The SDP signaling for an instant messaging session using the previous RTC 1.2 APIs is shown in the following:

```
v=0
o=- 0 0 IN IP4 172.27.174.212
s=session
c=IN IP4 172.27.174.212
t=0 0
m=x-ms-message 5060 sip null
```

Note The target of the instant messaging session in this case is carried in the To: header of the **INVITE request**.

The SDP signaling for an instant messaging session using the RTC 1.3 APIs is shown in the following:

```
v=0
o=- 0 0 IN IP4 157.56.66.71
s=session
c=IN IP4 157.56.66.71
t=0 0
m=message 5060 sip sip:peer@tradewind.com
```

The SIP URI present in the SDP signaling is the SIP URI of the peer user with whom a user wants to establish an IM session. This should match the SIP URI in the To header of the INVITE request. The port indicates the listening port of the client for direct connections if that is supported. The port SHOULD be ignored if the clients are connected through a server. The 1.3 client can also accept the older 1.2 format (which has null for the SIP URI and uses x-ms-message as the media type).

3.1.5.4 Data Collaboration Encryption

For data collaboration sessions, the initiating and receiving parties must negotiate support for encryption by using a new SDP attribute.<2>

The new SDP encryption attribute can be implemented in any of the following ways:

- If encryption is required, the SDP signaling carries a media-level attribute as shown in the following:

```
m=application 1503 tcp msdata
a=encryption:required
```

- If encryption is supported but not required, the SDP signaling carries a media-level attribute as shown in the following:<3>

```
m=application 1503 tcp msdata
a=encryption:optional
```

- If encryption is not allowed, the SDP signaling carries a media-level attribute as shown in the following:

```
m=application 1503 tcp msdata
a=encryption:rejected
```

3.1.5.4.1 Application Behavior

If data collaboration (DC) encryption is required at the initiating side, the application calls the application API (such as NetMeeting) to create a secured T.120 session. If encryption is not required, the initiating application can simply call the application API to create an unsecured T.120 session.<4>

In the absence of an application-supplied policy, the default policy is "allowed," meaning encryption is allowed but not required.

The logic for negotiation of support for encryption of T.120 DC sessions can be described in terms of application-level policy. Behavior is determined by the setting (required, optional, or not allowed) on both the initiator and receiver sides.

Initiator setting	Receiver setting	Result
Required	Required	DC encrypted
Required	Allowed	DC encrypted
Required	Not allowed	Failure
Allowed	Required	DC encrypted
Allowed	Allowed	DC not encrypted
Allowed	Not allowed	DC not encrypted

Initiator setting	Receiver setting	Result
Not allowed	Required	Failure
Not allowed	Allowed	DC not encrypted
Not allowed	Not allowed	DC not encrypted

In the preceding table, two circumstances yield a result of Failure:

- In the first case, the session initiator sends encryption:required.

  If the receiver understands the encryption attribute, it responds with a 488 response (described below) and the session setup fails. The initiator should display an error message to the user stating that encryption could not be negotiated.

  If the receiver does not understand the encryption attribute, the receiver responds with a **200 OK** (with no encryption attribute). The initiator acknowledges (ACK) the response. The T.120 session setup, however, fails and the initiator should display an error message to the user and immediately send a BYE to tear down the SIP signaling session.

- In the second case, the session initiator sends encryption:rejected.

  The receiver responds with a 488 error response and the session setup fails. The initiator should display an error message to the user stating that encryption could not be negotiated.

3.1.5.4.2 Negotiation Failure and Error Messages

If the initiator requires encryption and then receives a 488 error response to INVITE, the application should display a new error message in the user interface (UI) explaining to the user that the encryption could not be negotiated.

The 488 error response should contain a Warning: header to indicate that encryption levels are incompatible. The following is an example 488 error response:

```
SIP/2.0 488 Encryption Levels Incompatible
Warning: 308 ms.com "Encryption Levels Incompatible"
```

3.1.5.5 Audio/Video Encryption

For audio/video sessions, the initiating and receiving parties must negotiate support for encryption by using a new SDP attribute.

To implement the new SDP encryption attribute,<5> two pieces of information are required for audio/video encryption. The first is to determine whether encryption is required and the second is to determine what the encryption key should be. This can be implemented in any of the following ways:

- If encryption is required, the SDP signaling carries a media-level attribute as shown in the following:

```
m=audio 49170 RTP/AVP 0
a=encryption:required
```

- If encryption is supported but not required, the SDP signaling carries a media-level attribute as shown in the following:

```
m=audio 49170 RTP/AVP 0
a=encryption:optional
```

- If encryption is not allowed, the SDP signaling carries a media-level attribute as shown in the following:

```
m=audio 49170 RTP/AVP 0
a=encryption:rejected
```

The encryption key to use is carried in a media-level key parameter as shown in the following example:

```
m=audio 49170 RTP/AVP 0
a=encryption:required
k=base64:bhdsfsd78f7dssdfssfsd7sdfssa
```

The encryption key is not encrypted. The only transformation of the key is the base64 encoding for transport in the textual session description (SDP). The clients MUST send the key in base 64 encoded form and MUST decode the key received from the other endpoint before using it to decrypt audio/video. Protection of the encryption key is provided by the TLS transport over which the Session Initiation Protocol (SIP) signaling is carried. The encryption key is used to encrypt traffic sent by the supplier of the key as well as to decrypt traffic at the recipient. The keys used in each direction are independent and are typically different.

If SDP indicates that encryption is required or optional, an encryption key must also be supplied. In the case of optional encryption, the encryption key may go unused. A new encryption key should be generated and used for every new INVITE request. ReINVITEs will typically use the same encryption key.

For information about SIP, see [RFC3261].

3.1.5.5.1 Encryption Algorithms

- The RTCv1 application uses **MD5** as the default key generation method.

There are also several possible encryption algorithms.

- **DES**
- 3DES
- RC2
- RC4
- AES

The RTCv1 application uses DES as the default encryption algorithm, but it is recommended that a more secure algorithm, such as AES, be used as recommended in [RFC3550].

The RTCv1 application uses MD5 as the default key generation method, but any suitable technique may be used for this purpose.

Encryption of an **RTP** stream may involve encryption of just the RTP packets or encryption of both the RTP and **Real-Time Transport Control Protocol (RTCP)** packets. The RTCv1 application only supports encryption of both RTP and RTCP (as specified in [RFC3605] and [RFC3611]) packets.

3.1.5.5.2 Application Behavior

The following table summarizes the application behavior based on the audio/video encryption policy at both the initiator and the receiver. This behavior applies to both an initial INVITE (as specified in [RFC3261]) and a reINVITE. Failure of a reINVITE does not result in the existing session being torn down.

The default policy in the absence of an application-supplied policy is "allowed," meaning encryption is allowed but not required.

Initiator Setting	Receiver Setting	Result
Required	Required	A/V encrypted
Required	Allowed	A/V encrypted
Required	Not allowed	Failure
Allowed	Required	A/V encrypted
Allowed	Allowed	A/V not encrypted
Allowed	Not allowed	A/V not encrypted
Not allowed	Required	Failure
Not allowed	Allowed	A/V not encrypted
Not allowed	Not allowed	A/V not encrypted

In the preceding table, two circumstances yield a result of failure:

- In the first case, the session initiator requires encryption, but the receiver setting is to not allow encryption. As a result, the call setup fails.

  The receiver responds with a 488 error response and the session setup fails. The initiator should display an error message to the user stating that encryption could not be negotiated. Note that the user at the receiving client may not get an indication of the call because the call is automatically rejected.

- In the second case, the initiator does not allow encryption, but the receiver requires encryption due to policy. Again, the call fails because of incompatible encryption policies at the two endpoints.

  The receiver responds with a 488 error response and the session setup fails. The initiator should display an error message to the user stating that encryption could not be negotiated. Note that the user at the receiving client may not get an indication of the call because the call is automatically rejected.

3.1.5.5.3 Negotiation Failure and Error Messages

If the caller requires encryption but receives a 488 error response from the callee, the application should display a new error message in the user interface (UI) explaining to the user that the encryption could not be negotiated.

The 488 error response should contain a Warning: header to indicate that encryption is required. The following is an example 488 error response:

```
SIP/2.0 488 Encryption Levels Incompatible
Warning: 308 ms.com "Encryption Levels Incompatible"
```

3.1.6 Timer Events

There are no new timer events defined by Session Description Protocol (SDP) Extensions beyond those defined by SIP/SDP.

3.1.7 Other Local Events

There are no new local events defined by Session Description Protocol (SDP) Extensions beyond those defined by SIP/SDP.

4 Protocol Examples

The following sections describe several operations as used in common scenarios to illustrate the function of Session Description Protocol (SDP) Extensions.

4.1 Peer Clients Require Encryption

Alice sends Bob an INVITE request for an audio call requesting encryption by including the a=encryption:required attribute and the SDP field k= with a suitable key.

Bob accepts the call with a 200 OK with the a=encryption:required attribute and the k= field.

Alice sends an ACK.

Alice and Bob start sending media encrypted with the keys negotiated using the k= field.

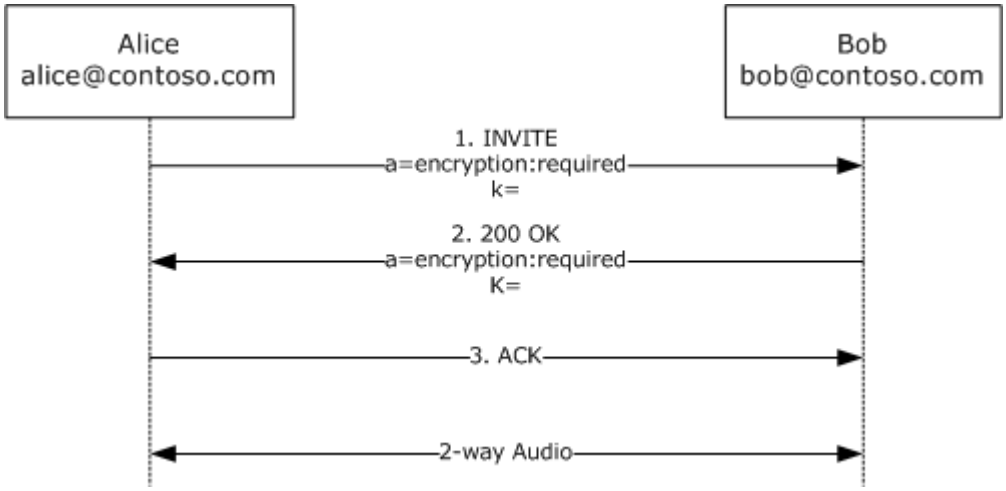

Figure 1: Audio call requesting encryption

An example of SDP signaling in a SIP INVITE request that is sent across the wire is as follows:

```
v=0
o=- 0 0 IN IP4 11.22.33.44
s=session
c=IN IP4 11.22.33.44
b=CT:1000
t=0 0
m=audio 5050 RTP/AVP 97 111 112 6 0 8 4 5 3 101
k=base64: Y6UN8SAFnqNTI61uN+1II3dqRk0spxbtqfuv5EYSYSM
a=rtpmap:97 red/8000
a=rtpmap:111 SIREN/16000
a=fmtp:111 bitrate=16000
a=rtpmap:112 G7221/16000
a=fmtp:112 bitrate=24000
a=rtpmap:6 DVI4/16000
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:4 G723/8000
a=rtpmap:5 DVI4/8000
```

```
a=rtpmap:3 GSM/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=encryption:required
m=video 5036 RTP/AVP 34 31
k=base64: Y6UN8SAFnqNTI61uN+1II3dqRk0spxbtqfuv5EYSYSM
a=rtpmap:34 H263/90000
a=rtpmap:31 H261/90000
a=encryption:required
```

An example of SDP signaling in the 200 OK response that is received across the wire in response to the INVITE is as follows:

```
v=0
o=- 0 0 IN IP4 11.22.33.55
s=session
c=IN IP4 11.22.33.55
b=CT:1000
t=0 0
m=audio 5050 RTP/AVP 97 111 112 6 0 8 4 5 3 101
k=base64: h6KmqIeIFj3rz4iiMAsYQju5EEitwAiveVu7RuHhev8
a=rtpmap:97 red/8000
a=rtpmap:111 SIREN/16000
a=fmtp:111 bitrate=16000
a=rtpmap:112 G7221/16000
a=fmtp:112 bitrate=24000
a=rtpmap:6 DVI4/16000
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:4 G723/8000
a=rtpmap:5 DVI4/8000
a=rtpmap:3 GSM/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=encryption:required
m=video 5036 RTP/AVP 34 31
k=base64: h6KmqIeIFj3rz4iiMAsYQju5EEitwAiveVu7RuHhev8
a=rtpmap:34 H263/90000
a=rtpmap:31 H261/90000
a=encryption:required
```

4.2 Client Requires Encryption but Peer Does Not Allow It

Alice sends Bob an INVITE for an audio/video call requesting encryption by including an a=encryption:required attribute and the SDP k= field with a suitable key.

- Since Bob does not allow encryption, he rejects the INVITE with a "488 Encryption Levels not compatible" response.

- Alice sends an ACK. The call is not established.

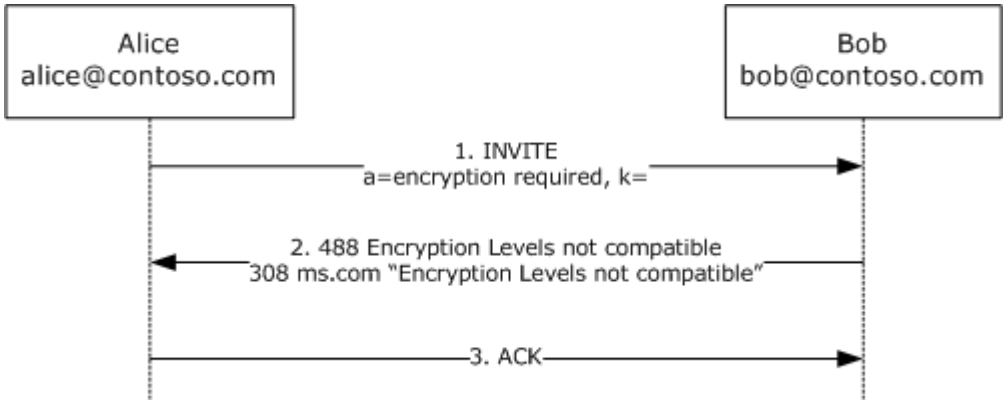

Figure 2: Alice requires encryption

An example of SDP signaling in the INVITE that is sent across the wire is as follows:

```
v=0
o=- 0 0 IN IP4 11.22.33.44
s=session
c=IN IP4 11.22.33.44
b=CT:1000
t=0 0
m=audio 5050 RTP/AVP 97 111 112 6 0 8 4 5 3 101
k=base64: Y6UN8SAFnqNTI61uN+1II3dqRk0spxbtqfuv5EYSYSM
a=rtpmap:97 red/8000
a=rtpmap:111 SIREN/16000
a=fmtp:111 bitrate=16000
a=rtpmap:112 G7221/16000
a=fmtp:112 bitrate=24000
a=rtpmap:6 DVI4/16000
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:4 G723/8000
a=rtpmap:5 DVI4/8000
a=rtpmap:3 GSM/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=encryption:required
m=video 5036 RTP/AVP 34 31
k=base64: Y6UN8SAFnqNTI61uN+1II3dqRk0spxbtqfuv5EYSYSM
a=rtpmap:34 H263/90000
a=rtpmap:31 H261/90000
a=encryption:required
```

An example of the SIP error response is as follows:

```
SIP/2.0 488 Encryption Levels not compatible
Via: SIP/2.0/TCP 11.22.33.44:4934
From: "Alice" <sip:alice@contoso.com>;tag=b8c543e1-ffb3-4b5a-880c-4d78f37643bb
To: <sip:bob@contoso.com>;tag=27221d0e-26d4-410f-9363-6ad3337cf152
```

```
Call-ID: 01a465dd-dafd-461d-a6ef-99c2c7df0e27@11.22.33.44
CSeq: 2 INVITE
Warning: 308 ms.com "Encryption Levels not compatible"
Content-Length: 0
```

5 Security

The following sections specify security considerations for implementers of Session Description Protocol (SDP) Extensions.

5.1 Security Considerations for Implementers

RTC APIs use the following security algorithms:

- MD5 is used for key generation.
- DES is used to encrypt the audio/video traffic.

Session Description Protocol (SDP) Extensions do not require the above algorithms to be used; other key generation and encryption algorithms may be used. However, if interoperability with RTC API clients is required, DES MUST be used for encryption.

5.2 Index of Security Parameters

Security Parameter	Section
Encryption algorithms	3.1.5.5.1

6 Appendix A: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows 2000 SP4
- Windows XP
- Windows Server 2003

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

<1> Section 1: Encryption is not supported on Windows 2000 and the associated protocol extensions are not used.

<2> Section 3.1.5.4: In Windows, the encryption key to use is actually generated by the underlying NetMeeting application and, as a result, does not need to be exchanged between the RTC clients through the SIP/SDP signaling. This is handled within the T.120 stream itself so no k= (key) line for the T.120 media will be present.

<3> Section 3.1.5.4: If the encryption attribute is not present in the SDP signaling, as will be the case for older Windows XP clients, encryption is considered to be supported but is not required.

<4> Section 3.1.5.4.1: When interoperating with a Windows XP client, the policy of the client can be considered as allowed. The Windows XP client does not send any encryption attribute in the SDP signaling of its INVITE or INVITE response, as described in [RFC3261].

<5> Section 3.1.5.5: If the encryption attribute is not present in the SDP signaling, as will be the case for older Windows XP clients, encryption is not supported. This is different from the data collaboration (DC) case because audio/video encryption requires a key exchange through the SDP signaling.

7 Index

T

U

V

W