[MS-SIPRE]:

Session Initiation Protocol (SIP) Routing Extensions

Revision Summary

Date	Revision History	Revision Class	Comments
4/4/2008	0.1	New	Initial Availability
4/25/2008	0.2	Major	Updated based on feedback
6/27/2008	1.0	Major	Updated and revised the technical content.
8/15/2008	1.01	Major	Revised and edited the technical content.
9/12/2008	1.02	Major	Revised and edited the technical content.
12/12/2008	2.0	Major	Updated and revised the technical content.
2/13/2009	2.01	Minor	Revised and edited the technical content.
3/13/2009	2.02	Minor	Revised and edited the technical content.
7/13/2009	2.03	Major	Revised and edited the technical content
8/28/2009	2.04	Editorial	Revised and edited the technical content
11/6/2009	2.05	Minor	Revised and edited the technical content
2/19/2010	2.06	Editorial	Revised and edited the technical content
3/31/2010	2.07	Major	Updated and revised the technical content
4/30/2010	2.08	Editorial	Revised and edited the technical content
6/7/2010	2.09	Editorial	Revised and edited the technical content
6/29/2010	2.10	Editorial	Changed language and formatting in the technical content.
7/23/2010	2.10	None	No changes to the meaning, language, or formatting of the technical content.
9/27/2010	3.0	Major	Significantly changed the technical content.
11/15/2010	3.0	None	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	3.0	None	No changes to the meaning, language, or formatting of the technical content.
3/18/2011	3.0	None	No changes to the meaning, language, or formatting of the technical content.
6/10/2011	3.0	None	No changes to the meaning, language, or formatting of the technical content.
1/20/2012	4.0	Major	Significantly changed the technical content.
4/11/2012	4.0	None	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	4.0	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	5.0	Major	Significantly changed the technical content.
2/11/2013	5.0	None	No changes to the meaning, language, or formatting of the

Date	Revision History	Revision Class	Comments
			technical content.
7/30/2013	5.0	None	No changes to the meaning, language, or formatting of the technical content.
11/18/2013	5.0	None	No changes to the meaning, language, or formatting of the technical content.
2/10/2014	5.0	None	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	5.1	Minor	Clarified the meaning of the technical content.
7/31/2014	5.2	Minor	Clarified the meaning of the technical content.
10/30/2014	5.2	None	No changes to the meaning, language, or formatting of the technical content.
3/30/2015	6.0	Major	Significantly changed the technical content.
9/4/2015	7.0	Major	Significantly changed the technical content.
7/15/2016	7.0.1	Editorial	Changed language and formatting in the technical content.

Table of Contents

1 Introduction

This document specifies proprietary software application extensions for implementing call routing functionality to the Session Initiation Protocol (SIP). SIP is used by applications to establish, modify, and terminate multimedia sessions or calls.

The extensions discussed in this protocol are used by SIP clients, proxies, and servers.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

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

Active Directory: A general-purpose network directory service. **Active Directory** also refers to the Windows implementation of a directory service. **Active Directory** stores information about a variety of objects in the network. Importantly, user accounts, computer accounts, groups, and all related credential information used by the Windows implementation of Kerberos are stored in **Active Directory**. **Active Directory** is either deployed as Active Directory Domain Services (AD DS) or Active Directory Lightweight Directory Services (AD LDS). [MS-ADTS] describes both forms. For more information, see [MS-AUTHSOD] section 1.1.1.5.2, Lightweight Directory Access Protocol (LDAP) versions 2 and 3, Kerberos, and DNS.

address-of-record: A **Session Initiation Protocol (SIP) URI** that specifies a domain with a location service that can map the URI to another URI for a user, as described in [RFC3261].

Augmented Backus-Naur Form (ABNF): A modified version of Backus-Naur Form (BNF), commonly used by Internet specifications. ABNF notation balances compactness and simplicity with reasonable representational power. ABNF differs from standard BNF in its definitions and uses of naming rules, repetition, alternatives, order-independence, and value ranges. For more information, see [RFC5234].

authentication: The act of proving an identity to a server while providing key material that binds the identity to subsequent communications.

call: A communication between peers that is configured for a multimedia conversation.

callee: An **endpoint** to which a **call** is initiated by a **caller**.

caller: An **endpoint** that initiates a **call** to establish a media session.

conference: A Real-Time Transport Protocol (RTP) session that includes more than one **participant**.

content type: A named and uniquely identifiable collection of settings and fields that store metadata for individual items in a SharePoint list. One or more content types can be associated with a list, which restricts the contents to items of those types.

Coordinated Universal Time (UTC): A high-precision atomic time standard that approximately tracks Universal Time (UT). It is the basis for legal, civil time all over the Earth. Time zones around the world are expressed as positive and negative offsets from UTC. In this role, it is also referred to as Zulu time (Z) and Greenwich Mean Time (GMT). In these specifications, all references to UTC refer to the time at UTC-0 (or GMT).

delegate: A user or resource that has permissions to act on behalf of another user or resource.

dialog: A peer-to-peer **Session Initiation Protocol (SIP)** relationship that exists between two **user agents** and persists for a period of time. A dialog is established by **SIP messages**, such as a 2xx response to an INVITE request, and is identified by a call identifier, a local tag, and a remote tag.

domain: A set of users and computers sharing a common namespace and management infrastructure. At least one computer member of the set must act as a domain controller (DC) and host a member list that identifies all members of the domain, as well as optionally hosting the **Active Directory** service. The domain controller provides **authentication** of members, creating a unit of trust for its members. Each domain has an identifier that is shared among its members. For more information, see [MS-AUTHSOD] section 1.1.1.5 and [MS-ADTS].

endpoint: A device that is connected to a computer network.

endpoint identifier (EPID): A unique identifier of a Session Initiation Protocol (SIP) **endpoint**. It is formed by combining the value of an epid parameter in a From or To header field with the **address-of-record** in the corresponding header field.

external user: Any user who is located outside the enterprise network boundary, including **remote users**, federated users, and public instant messaging (IM) users.

federated partner: An enterprise that is trusted for **federation**.

federated user: An external user who possesses valid credentials with a federated partner and who therefore is treated as authenticated by a protocol server.

federation: The ability of a server deployment to interoperate with other servers that were deployed by other enterprises.

fully qualified domain name (FQDN): An unambiguous domain name (2) that gives an absolute location in the Domain Name System's (DNS) hierarchy tree, as defined in [RFC1035] section 3.1 and [RFC2181] section 11.

Globally Routable User Agent URI (GRUU): A **URI** that identifies a **user agent** and is globally routable. A URI possesses a GRUU property if it is useable by any **user agent client (UAC)** that is connected to the Internet, routable to a specific user agent instance, and long-lived.

globally unique identifier (GUID): A term used interchangeably with **universally unique identifier (UUID)** in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the **GUID**. See also **universally unique identifier (UUID)**.

hash: A fixed-size result that is obtained by applying a one-way mathematical function, which is sometimes referred to as a hash algorithm, to an arbitrary amount of data. If the input data changes, the hash also changes. The hash can be used in many operations, including **authentication** and digital signing.

Hash-based Message Authentication Code (HMAC): A mechanism for message **authentication** using cryptographic hash functions. HMAC can be used with any iterative cryptographic hash function (for example, MD5 and **SHA-1**) in combination with a secret shared key. The cryptographic strength of HMAC depends on the properties of the underlying hash function.

header field: A component of a Session Initiation Protocol (SIP) message header, as described in [RFC3261].

in-band provisioning: A process in which a protocol client obtains configuration information from a protocol server.

Interactive Connectivity Establishment (ICE): A methodology that was established by the Internet Engineering Task Force (IETF) to facilitate the traversal of network address translation (NAT) by media.

INVITE: A **Session Initiation Protocol (SIP)** method that is used to invite a user or a service to participate in a session.

location profile: A definition of an environment where local numbers can be resolved to identifiers that either route to unique enterprise users or form unique numbers in a public telephone network, as defined by the International Telecommunications Union (ITU) recommendation.

location profile description: An **XML document** that contains the name of a **location profile** and a set of **translation rules** that are associated with that profile.

Media Access Control (MAC) address: A hardware address provided by the network interface vendor that uniquely identifies each interface on a physical network for communication with other interfaces, as specified in [IEEE802.3]. It is used by the media access control sublayer of the data link layer of a network connection.

Multipurpose Internet Mail Extensions (MIME): A set of extensions that redefines and expands support for various types of content in email messages, as described in [RFC2045], [RFC2046], and [RFC2047].

network address translation (NAT): The process of converting between IP addresses used within an intranet, or other private network, and Internet IP addresses.

NOTIFY: A method that is used to notify a **Session Initiation Protocol (SIP)** client that an event requested by an earlier SUBSCRIBE method has occurred. The notification optionally provides details about the event.

optimized dialing: A client-side optimization that occurs when users start dialing a phone number. The protocol client compares the collected digit sequence with the translation rules in the location profile and, when a match is detected, applies the rule and sends an INVITE request to the protocol server.

participant: A user who is participating in a **conference** or peer-to-peer **call**, or the object that is used to represent that user.

Presence Information Data Format (PIDF): A common data format defined in [RFC3863] to exchange presence information.

private line: A feature that can be enabled for a voice account and provides an additional, unpublished phone number for a user. A user can choose to disclose the phone number for a private line.

proxy: A computer, or the software that runs on it, that acts as a barrier between a network and the Internet by presenting only a single network address to external sites. By acting as a go-between that represents all internal computers, the proxy helps protects network identities while also providing access to the Internet.

public IM connectivity: The ability of a protocol server deployment to interoperate with a public instant messaging (IM) provider.

public IM provider: A provider of a public instant messaging (IM) service.

public IM user: An external user who belongs to a public instant messaging (IM) provider.

public switched telephone network (PSTN): Public switched telephone network is the voice-oriented public switched telephone network. It is circuit-switched, as opposed to the packet-switched networks.

REGISTER: A **Session Initiation Protocol (SIP)** method that is used by an SIP client to register the client address with an SIP server.

remote user: A user who has a persistent identity within an enterprise and is connected from outside the enterprise network boundary.

Request-URI: A **URI** in an HTTP request message, as described in [RFC2616].

security association (SA): A simplex "connection" that provides security services to the traffic carried by it. See [RFC4301] for more information.

server: A replicating machine that sends replicated files to a partner (client). The term "server" refers to the machine acting in response to requests from partners that want to receive replicated files.

SERVICE: A method that is defined by **Session Initiation Protocol (SIP)** extensions and is used by an SIP client to request a service from a server.

Session Description Protocol (SDP): A protocol that is used for session announcement, session invitation, and other forms of multimedia session initiation. For more information see [MS-SDP] and [RFC3264].

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

SHA-1: An algorithm that generates a 160-bit hash value from an arbitrary amount of input data, as described in [RFC3174]. SHA-1 is used with the Digital Signature Algorithm (DSA) in the Digital Signature Standard (DSS), in addition to other algorithms and standards.

SHA-1 hash: A hashing algorithm as specified in [FIPS180-2] that was developed by the National Institute of Standards and Technology (NIST) and the National Security Agency (NSA).

SIP element: An entity that understands the **Session Initiation Protocol (SIP)**.

SIP message: The data that is exchanged between **Session Initiation Protocol (SIP)** elements as part of the protocol. An SIP message is either a request or a response.

SIP protocol client: A network client that sends **Session Initiation Protocol (SIP)** requests and receives SIP responses. An SIP client does not necessarily interact directly with a human user. **User agent clients (UACs)** and proxies are SIP clients.

SIP registrar: A **Session Initiation Protocol (SIP)** server that accepts REGISTER requests and places the information that it receives from those requests into the location service for the domain that it handles.

SIP request: A **Session Initiation Protocol (SIP)** message that is sent from a **user agent client (UAC)** to a **user agent server (UAS)** to call a specific operation.

SIP response: A **Session Initiation Protocol (SIP)** message that is sent from a **user agent server (UAS)** to a **user agent client (UAC)** to indicate the status of a request from the UAC to the UAS.

SIP transaction: A **SIP transaction** occurs between a **UAC** and a **UAS**. The **SIP transaction** comprises all messages from the first request sent from the **UAC** to the **UAS** up to a final response (non-1xx) sent from the **UAS** to the **UAC**. If the request is **INVITE**, and the final response is a non-2xx, the **SIP transaction** also includes an ACK to the response. The ACK for a 2xx response to an **INVITE** request is a separate **SIP transaction**.

SUBSCRIBE: A **Session Initiation Protocol (SIP)** method that is used to request asynchronous notification of an event or a set of events at a later time.

token: A word in an item or a search query that translates into a meaningful word or number in written text. A token is the smallest textual unit that can be matched in a search query. Examples include "cat", "AB14", or "42".

transaction: The process of opening or creating an object on a server, and the subsequent committing of changes to the object by calling the required save function, at which time all changes to that instance of the object are either saved to the server, or discarded if a failure occurs before saving is finished successfully. Until successfully saved, changes are invisible to any other instances of the object.

translation rule: A tuple that consists of a regular expression that matches a subset of local numbers and a replacement pattern for it.

Transmission Control Protocol (TCP): A protocol used with the Internet Protocol (IP) to send data in the form of message units between computers over the Internet. TCP handles keeping track of the individual units of data (called packets) that a message is divided into for efficient routing through the Internet.

tuple: An ordered grouping of members from different dimensions or hierarchies. A single member is a special case of a tuple and can be used as an expression. Every hierarchy does not have to be represented in a tuple.

Uniform Resource Identifier (URI): A string that identifies a resource. The URI is an addressing mechanism defined in Internet Engineering Task Force (IETF) Uniform Resource Identifier (URI): Generic Syntax [RFC3986].

Uniform Resource Locator (URL): A string of characters in a standardized format that identifies a document or resource on the World Wide Web. The format is as specified in [RFC1738].

Uniform Resource Name (URN): A string that identifies a persistent Internet resource, as described in [RFC2141]. A URN can provide a mechanism for locating and retrieving a schema file that defines a specific namespace. Although a URL can provide similar functionality, a URN can refer to more than one URL and is not location-dependent.

universally unique identifier (UUID): A 128-bit value. UUIDs can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects in cross-process communication such as client and server interfaces, manager entry-point vectors, and RPC objects. UUIDs are highly likely to be unique. UUIDs are also known as **globally unique identifiers (GUIDs)** and these terms are used interchangeably in the Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the UUID. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the UUID.

user agent: An HTTP user agent, as specified in [RFC2616].

user agent client (UAC): A logical entity that creates a new request, and then uses the client transaction state machinery to send it. The role of **UAC** lasts only for the duration of that transaction. In other words, if a piece of software initiates a request, it acts as a **UAC** for the duration of that transaction. If it receives a request later, it assumes the role of a **user agent server (UAS)** for the processing of that transaction.

user agent server (UAS): A logical entity that generates a response to a **Session Initiation Protocol (SIP)** request. The response either accepts, rejects, or redirects the request. The role of the UAS lasts only for the duration of that transaction. If a process responds to a request, it acts as a UAS for that transaction. If it initiates a request later, it assumes the role of a **user agent client (UAC)** for that transaction.

web service: A unit of application logic that provides data and services to other applications and can be called by using standard Internet transport protocols such as HTTP, Simple Mail Transfer

Protocol (SMTP), or File Transfer Protocol (FTP). Web services can perform functions that range from simple requests to complicated business processes.

XML attribute: A name/value pair, separated by an equal sign (=) and included in a tagged element, that modifies features of an element. All XML attribute values are stored as strings enclosed in quotation marks.

XML document: A document object that is well formed, as described in [XML10/5], and might be valid. An XML document has a logical structure that is composed of declarations, elements, comments, character references, and processing instructions. It also has a physical structure that is composed of entities, starting with the root, or document, entity.

XML element: An XML structure that typically consists of a start tag, an end tag, and the information between those tags. Elements can have attributes (1) and can contain other elements.

XML namespace: A collection of names that is used to identify elements, types, and attributes in XML documents identified in a URI reference [RFC3986]. A combination of XML namespace and local name allows XML documents to use elements, types, and attributes that have the same names but come from different sources. For more information, see [XMLNS-2ED].

XML namespace prefix: An abbreviated form of an **XML namespace**, as described in [XML].

XML schema: A description of a type of **XML document** that is typically expressed in terms of constraints on the structure and content of documents of that type, in addition to the basic syntax constraints that are imposed by XML itself. An XML schema provides a view of a document type at a relatively high level of abstraction.

XML schema definition (XSD): The World Wide Web Consortium (W3C) standard language that is used in defining XML schemas. Schemas are useful for enforcing structure and constraining the types of data that can be used validly within other XML documents. XML schema definition refers to the fully specified and currently recommended standard for use in authoring **XML schemas**.

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

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.

[E164] ITU-T, "The International Public Telecommunication Numbering Plan", Recommendation E.164, February 2005, http://www.itu.int/rec/T-REC-E.164/e

Note There is a charge to download the specification.

[FIPS180] FIPS PUBS, "Secure Hash Standard", FIPS PUB 180-1, April 1995, http://niatec.info/GetFile.aspx?pid=63

[FIPS198a] National Institute of Standards and Technology, "The Keyed-Hash Message Authentication Code (HMAC)", FIPS PUB 198, March 2002, http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf

[IETFDRAFT-ICENAT-06] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", draft-ietf-mmusic-ice-06, October 2005, http://tools.ietf.org/html/draft-ietf-mmusic-ice-06

[IETFDRAFT-ICENAT-19] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", draft-ietf-mmusic-ice-19, October 2007, http://tools.ietf.org/html/draft-ietf-mmusic-ice-19

[IETFDRAFT-MCICSIP-11] Jennings, C., Ed. and Mahy, R., Ed., "Managing Client Initiated Connections in the Session Initiation Protocol (SIP)", draft-ietf-sip-outbound-11, November 2007, http://tools.ietf.org/id/draft-ietf-sip-outbound-11.txt

[IETFDRAFT-OUGRUAUSIP-10] Rosenberg, J., "Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)", draft-ietf-sip-gruu-10, July 2006, http://tools.ietf.org/id/draft-ietf-sip-gruu-10.txt

[IETFDRAFT-RCDPR-303-01] Ramanathan, R., Parameswar, S., and Vakil, M., "Response Code for Dynamic Proxy Redirect", draft-rajesh-sipping-303-01, February 2007, http://tools.ietf.org/id/draft-rajesh-sipping-303-01.txt

[IETFDRAFT-RCITD-199-01] Holmberg, C., "Response Code for Indication of Terminated Dialog", draft-ietf-sip-199-01.txt, August 2008, http://tools.ietf.org/id/draft-ietf-sip-199-01.txt

[IETFDRAFT-SF-605-01] Ramanathan, R., Vakil, M., and Parameswar, S., "Serial Forking and 605", draft-rajesh-sipping-605-01, March 2007, http://tools.ietf.org/id/draft-rajesh-sipping-605-01.txt

[IETFDRAFT-SIPSOAP-00] Deason, N., "SIP and SOAP", draft-deason-sip-soap-00, June 30 2000, http://www.softarmor.com/wgdb/docs/draft-deason-sip-soap-00.txt

[MC-RegEx] Microsoft Corporation, "Regular Expression Language Elements", http://msdn.microsoft.com/en-us/library/az24scfc(VS.80).aspx

[MS-CONFBAS] Microsoft Corporation, "Centralized Conference Control Protocol: Basic Architecture and Signaling".

[MS-CONFPRO] Microsoft Corporation, "Centralized Conference Control Protocol: Provisioning".

[MS-E911WS] Microsoft Corporation, "Web Service for E911 Support Protocol".

[MS-PRES] Microsoft Corporation, "Presence Protocol".

[MS-SDPEXT] Microsoft Corporation, "Session Description Protocol (SDP) Version 2.0 Extensions".

[MS-SIPREGE] Microsoft Corporation, "Session Initiation Protocol (SIP) Registration Extensions".

[RFC2046] Freed, N., and Borenstein, N., "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996, http://www.rfc-editor.org/rfc/rfc2046.txt

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

[RFC2141] Moats, R., "URN Syntax", RFC 2141, May 1997, http://www.rfc-editor.org/rfc/rfc2141.txt

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and Schooler, E., "SIP: Session Initiation Protocol", RFC 3261, June 2002, http://www.ietf.org/rfc/rfc3261.txt

[RFC3264] Rosenberg, J., and Schulzrinne, H., "An Offer/Answer Model with the Session Description Protocol (SDP)", RFC 3264, June 2002, http://www.rfc-editor.org/rfc/rfc3264.txt

[RFC3265] Roach, A. B., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002, http://www.ietf.org/rfc/rfc3265.txt

[RFC3325] Jennings, C., Peterson, J., and Watson, M., "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks", RFC 3325, November 2002, http://www.rfc-editor.org/rfc/rfc3325.txt

[RFC3326] Schulzrinne, H., Oran, D., and Camarillo, G., "The Reason Header Field for the Session Initiation Protocol (SIP)", RFC 3326, December 2002, http://www.rfc-editor.org/rfc/rfc3326.txt

[RFC3327] Willis, D., and Hoeneisen, B., "Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts", RFC 3327, December 2002, http://www.rfc-editor.org/rfc/rfc3327.txt

[RFC3548] Josefsson, S., Ed., "The Base16, Base32, and Base64 Data Encodings", RFC 3548, July 2003, http://www.rfc-editor.org/rfc/rfc3548.txt

[RFC3863] Sugano, H., Fujimoto, S., Klyne, G., et al., "Presence Information Data Format (PIDF)", RFC 3863, August 2004, http://www.ietf.org/rfc/rfc3863.txt

[RFC3892] Sparks, R., "The Session Initiation Protocol (SIP) Referred-By Mechanism", RFC 3892, September 2004, http://www.rfc-editor.org/rfc/rfc3892.txt

[RFC3966] Schulzrinne, H., "The tel URI for Telephone Numbers", RFC 3966, December 2004, http://www.rfc-editor.org/rfc/rfc3966.txt

[RFC3986] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005, http://www.ietf.org/rfc/rfc3986.txt

[RFC4028] Donovan, S., and Rosenberg, J., "Session Timers in the Session Initiation Protocol (SIP)", RFC 4028, April 2005, http://www.rfc-editor.org/rfc/rfc4028.txt

[RFC4119] Peterson, J., "A Presence-based GEOPRIV Location Object Format", RFC 4119, December 2005, http://www.rfc-editor.org/rfc/rfc4119.txt

[RFC4122] Leach, P., Mealling, M., and Salz, R., "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, July 2005, http://www.ietf.org/rfc/rfc4122.txt

[RFC4235] Rosenberg, J., Schulzrinne, H., and Mahy, R., Ed., "An INVITE-Initiated Dialog Event Package for the Session Initiation Protocol (SIP)", RFC 4235, November 2005, http://www.rfc-editor.org/rfc/rfc4235.txt

[RFC4244] Barnes, M., Ed., "An Extension to the Session Initiation Protocol (SIP) for Request History Information", RFC 4244, November 2005, http://www.rfc-editor.org/rfc/rfc4244.txt

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

[RFC5139] Thomson, M. and Winterbottom, J., "Revised Civic Location Format for Presence Information Data Format Location Object (PIDF-LO)", February 2008, http://www.rfc-editor.org/rfc/rfc5139.txt

[RFC6442] Polk, J., Rosen, B., and Peterson, J., "Location Conveyance for the Session Initiation Protocol", RFC 6442, December 2011, http://www.rfc-editor.org/rfc/rfc6442.txt

[XMLNS] Bray, T., Hollander, D., Layman, A., et al., Eds., "Namespaces in XML 1.0 (Third Edition)", W3C Recommendation, December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/

[XMLSCHEMA1] Thompson, H., Beech, D., Maloney, M., and Mendelsohn, N., Eds., "XML Schema Part 1: Structures", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[XMLSCHEMA2] Biron, P.V., Ed. and Malhotra, A., Ed., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

1.2.2 Informative References

[RFC5234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008, http://www.rfc-editor.org/rfc/rfc5234.txt

[XML10] World Wide Web Consortium, "Extensible Markup Language (XML) 1.0 (Third Edition)", February 2004, http://www.w3.org/TR/2004/REC-xml-20040204/

[XMLSCHEMA0] Fallside, D., and Walmsley, P., Eds., "XML Schema Part 0: Primer, Second Edition", W3C Recommendation, October 2004, http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/

1.3 Overview

This document discusses **Session Initiation Protocol (SIP)** extensions that are used in this protocol architecture.

Endpoint identification extensions have been designed to help route **calls** within SIP topologies with more than one protocol client endpoint. They provide unique identities and addresses to multiple communication endpoints representing the same user or service and allow the **servers** and other protocol clients to identify a specific endpoint that initiated communication and to route calls to a specific endpoint. These extensions are described in detail in section 3.2 through section 3.4.

Extensions to SIP **Uniform Resource Identifier (URI)** and **header field** syntax ensure that messages within **SIP transactions** are processed consistently and reliably delivered within SIP topologies with multiple redundant servers. These extensions also resolve addressing issues in network topologies where the protocol client and server are separated by a firewall or a **network address translation (NAT)** device. These extensions are described in detail in section 3.5, section 3.6, and section 3.7.

The phone number resolution extensions provide a way for **SIP elements** to resolve partially specified local phone numbers to a number that allows the server to route the call to a unique enterprise user or forms a unique number in a public telephone network, as defined by International Telecommunications Union Recommendation. These extensions are described in detail in section 3.8.

The routing script preamble and call designation extensions provide a way for a protocol client to describe a set of endpoints to receive calls targeted at the user as well as define parameters for routing action taken by the server when processing these calls. These extensions are described in section 3.9.

The extensions for **federation** and **public IM connectivity** provide a way to inform protocol clients whether the **SIP message** is from a **remote user**, **federated user**, or a **public IM user**. The extensions for remote users provide a way to inform a protocol client that it is connected to the server from outside the enterprise network boundary. These extensions are described in section 3.10 and section 3.11.

Section 3.12 describes an extension that provides a way to correlate multiple SIP **dialogs** for logging and monitoring purposes.

The extensions to create notes and other context information related to a given call and send them to another party during **transaction** establishment are described in section 3.13, section 3.14, and section 3.15.

The extensions to provide anonymity to a call are described in section 3.16.

Section 3.17 describes the extensions to handle emergency calls.

1.4 Relationship to Other Protocols

This protocol defines an **XML schema** that supports various extensions specified in this protocol. For more information about XML, see [XML10], [XMLNS], and [XMLSCHEMA0].

This protocol is invoked as an extension of SIP. This protocol incorporates SIP protocols.

1.5 Prerequisites/Preconditions

This protocol assumes that both the **SIP protocol clients** and the server support SIP. The prerequisites for this protocol and the SIP prerequisites are identical.

1.6 Applicability Statement

This protocol is applicable when both the SIP protocol clients and the server support SIP and intend to use one or more of the enhancements offered by this protocol.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

Standard SIP extension mechanisms as specified in [RFC3261] can be used by vendors as needed.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

This protocol does not introduce a new transport to exchange messages and is capable of being used with any transport used by SIP.

2.2 Message Syntax

This protocol relies on the SIP message format, as specified in [RFC3261] section 7, and extends definitions of URI and header field parameters by adding new values for parameter and header field names, as well as their corresponding values. This protocol defines new message body types in addition to those defined in [RFC3261]. All of the message syntax specified in this protocol is described in both prose and an **Augmented Backus-Naur Form (ABNF)**, as defined in [RFC5234].

2.2.1 Namespaces

This specification defines and references various **XML namespaces** using the mechanisms specified in [XMLNS]. Although this specification associates a specific **XML namespace prefix** for each XML namespace that is used, the choice of any particular XML namespace prefix is implementation-specific and not significant for interoperability.

Prefix	Namespace URI	Reference
xs	http://www.w3.org/2001/XMLSchema	[XMLSCHEMA1] [XMLSCHEMA2]
xsd	http://www.w3.org/2001/XMLSchema	[XMLSCHEMA1] [XMLSCHEMA2]
pidftns	urn:schema:Rtc.LIS.msftE911PidfExtn.2008	
callctns	http://schemas.microsoft.com/2008/03/sip/conversationContext	
tns	http://schemas.microsoft.com/02/2006/sip/routing	

2.2.2 SIP URI Parameter Extensions

This protocol defines several new URI parameter names and values. The original ABNF, as defined in [RFC5234], for **uri-parameter** in [RFC3261] section 25 is extended as follows:

```
uri-parameter = transport-param / user-param / method-param
                / ttl-param / maddr-param / lr-param
                / opaque-param
                / gruu-param
                / grid-param
                / received-param
                / ms-opaque-param
                / ms-received-cid-param
                / ms-route-sig-param
                / ms-key-info-param
                / ms-identity-param
                / ms-fe-param
                / ms-role-rs-to-param
                / ms-role-rs-from-param
                / ms-ent-dest-param
```

```
                    / default-param
                    / phone-context-param
                    / other-param
opaque-param = "opaque=" opaque-value
opaque-value = ua-opaque-val
               / app-voicemail-opaque-val
               / app-locationprofile-opaque-val
               / app-conf-opaque-val
               / server-opaque-val
               / state-opaque-val
               / pvalue
ua-opaque-val = "user:epid:" encoded-uuid-val
app-voicemail-opaque-val = "app:voicemail"
app-locationprofile-opaque-val = "app:locationprofile:get"
app-conf-opaque-val = "app:conf:" conf-entity-val ":id:"
                      encoded-conf-id-val
server-opaque-val = "srvr:" server-type-val ":"
                    encoded-server-instance-val
state-opaque-val = "state:" pvalue
encoded-uuid-val = 1*paramchar
conf-entity-val = "focus" / "audio-video" / "chat"
                  / "meeting" / "phone-conf"
encoded-conf-id-val = 1*paramchar
server-type-val = "HomeServer" / "MediationServer" / "MRAS" / "QoSM"
encoded-server-instance-val = 1*paramchar
gruu-param = "gruu"
grid-param = "grid" ["=" pvalue]
received-param = "received=" (IPv4address / IPv6address)
ms-opaque-param = "ms-opaque=" pvalue
ms-received-cid-param = "ms-received-cid=" pvalue
ms-route-sig-param = "ms-route-sig=" pvalue
ms-key-info-param = "ms-key-info=" pvalue
ms-fe-param = "ms-fe=" pvalue
ms-role-rs-to-param = "ms-role-rs-to"
ms-role-rs-from-param = "ms-role-rs-from"
ms-ent-dest-param = "ms-ent-dest"
ms-identity-param = "ms-identity=" pvalue
default-param = "default"
phone-context-param = "phone-context=" descriptor
descriptor = domainname / global-number-digits
```

state-opaque-val follows the product behavior in this endnote<1>.

paramchar, **pvalue**, **IPv4address**, and **IPv6address** are defined in [RFC3261] section 25.

domainname and **global-number-digits** are defined in [RFC3966] section 3.

2.2.2.1 SIP URI Parameter Extensions for Record-Route, Path, and Route Header Fields

The following SIP URI parameter extensions can be used in URIs inserted by SIP **proxies** into the **Record-Route** header fields of any message described in [RFC3261] section 16, or into the **Path** header field of the **REGISTER** request described in [RFC3327] section 5.

- **ms-opaque-param**
- **ms-route-sig-param**
- **ms-key-info-param**
- **ms-identity-param**
- **ms-fe-param**

- **ms-role-rs-to-param**
- **ms-role-rs-from-param**
- **ms-ent-dest-param**

These extensions can then appear in the **Route** header field. As specified in [RFC3261] section 12, the list of URIs in the **Record-Route** header fields, taken in order with all URI parameters, is stored in the dialog state. This list of URIs is also stored in the **Route** header fields of every **SIP request** in the SIP dialog. Additionally, as specified in [RFC3327] section 5, the content of the **Path** header fields is stored by the registrar and then used by the SIP proxy that is responsible for the **domain** of the request destination to populate **Route** header fields.

2.2.2.2 SIP URI Parameter Extensions for Contact, Route Header and Request-URI Fields

The following SIP URI parameter extensions can be inserted by SIP elements into the **URI** of the **Contact** header field:

- **opaque-param**
- **gruu-param**
- **grid-param**
- **ms-fe-param**
- **ms-opaque-param**

These extensions can then appear in the **Request-URI** field because, as specified in [RFC3261] section 12, the URI in the **Contact** header field is stored in the dialog state and is included as the **Request-URI** field of each SIP request within a dialog. Also, if the **Contact** header field is used in the REGISTER request, as described in [RFC3261] section 10, the **Contact** header field can be stored by the SIP location service and then used by the SIP proxy, as described in [RFC3261] section 16, to populate the **Request-URI** field. In addition, as described in [RFC3261] section 16.4, if the SIP element sending the request is a strict router, it can place the URI from the **Contact** header field into the **Route** header field.

2.2.2.3 SIP URI Parameter Extensions for Contact, Record-Route, Path, Route Header and Request-URI Fields

The following SIP **URI** parameter extensions can be inserted by the SIP proxy into the **URI**s of the **Contact**, **Record-Route**, or **Path** header fields created by the upstream SIP element:

- **received-param**
- **ms-received-cid-param**

If inserted into the **URI** of **Record-Route** or **Path** header fields, these parameter extensions can appear in the **Route** header field, as described in section 2.2.2.1. If inserted into the **URI** of the **Contact** header field, these extensions can appear in the Request-URI field, as described in section 2.2.2.2.

2.2.3 Syntax of Globally Routable User Agent URI

This protocol defines several **Globally Routable User Agent URI (GRUU)** syntax forms for the **SIP registrar** that is compliant with this protocol. These syntax forms are based on SIP URI parameter extensions described in section 2.2.2 and are intended to satisfy the requirements for the GRUU syntax that is defined in [IETFDRAFT-OUGRUAUSIP-10] section 6.

```
user-agent-gruu = "sip:" address-of-record *(";" user-agent-gruu-param)
user-agent-gruu-param = "gruu" / "opaque=" ua-opaque-val

voice-mail-gruu = "sip:" address-of-record *(";" voice-mail-gruu-param)
voice-mail-gruu-param = "gruu" / "opaque=" app-voicemail-opaque-val

location-profile-gruu = "sip:" address-of-record
                        *(";" location-profile-gruu-param)
location-profile-gruu-param = "gruu"
                              / "opaque=" app-locationprofile-opaque-val
                              / default-param
                              / phone-context-param

conf-endpoint-gruu = sip:" address-of-record *(";" conf-endpoint-gruu-param)
conf-endpoint-gruu-param = "gruu" / "opaque=" app-conf-opaque-val

server-instance-gruu = "sip:" server-fqdn "@" domain-fqdn
                       *(";" server-instance-gruu-param)
server-instance-gruu-param = "gruu" / "opaque=" server-opaque-val

address-of-record = userinfo host
server-fqdn = host
domain-fqdn = host
```

default-param, phone-context-param, ua-opaque-val, **app-voicemail-opaque-val**, **app-conf-opaque-val**, **server-opaque-val**, and **app-locationprofile-opaque-val** are defined in section 2.2.2.

userinfo and **host** are defined in [RFC3261] section 25.1.

2.2.4 Record-Route Header Field Extension

This protocol defines a new **Record-Route** header field parameter and its value. The original ABNF, as defined in [RFC5234], for the **Record-Route** header field in [RFC3261] section 25 is extended as follows:

```
rr-param = rr-p-ms-rrsig
           / generic-param
rr-p-ms-rrsig = "ms-rrsig=" pvalue
```

pvalue is defined in [RFC3261] section 25.

2.2.5 Contact Header Field Extensions

This protocol defines a new **Contact** header field parameter and its value. The original ABNF, as defined in [RFC5234], for the **Contact** header field in [RFC3261] section 25 is extended as follows:

```
contact-params = c-p-q / c-p-expires
                 / c-p-proxy
                 / contact-extension
c-p-proxy = "proxy=" "replace"
```

In addition to the extension defined in this protocol, this protocol uses the **sip.instance** media feature tag introduced in [IETFDRAFT-MCICSIP-11] section 12.5, with syntax defined in [IETFDRAFT-MCICSIP-

11] section 10, for use as the **Contact** header field parameter. The syntax for the **+sip.instance** parameter in the **Contact** header field from [IETFDRAFT-MCICSIP-11] section 10 is as follows:

```
c-p-instance    =  "+sip.instance" EQUAL
                   LDQUOT "<" instance-val ">" RDQUOT
instance-val    = *uric ; defined in [RFC3986]
```

Because this protocol requires that only a **universally unique identifier (UUID) Uniform Resource Name (URN)** be used as the **+sip.instance** parameter value, the **instance-val** definition is restricted to the UUID URN syntax (**UUID-URN**), as defined in [RFC4122] and [RFC2141].

The URN definition from [RFC2141], as applicable to the UUID URN defined in [RFC4122] is as follows:

```
UUID-URN = "urn:" UUID-NID ":" UUID-NSS
```

The UUID namespace identifier syntax from [RFC4122] is as follows:

```
UUID-NID = "uuid"
```

The UUID namespace specific string syntax from [RFC4122] is as follows:

```
UUID-NSS               = time-low "-" time-mid "-"
                         time-high-and-version "-"
                         clock-seq-and-reserved
                         clock-seq-low "-" node
time-low               = 4hexOctet
time-mid               = 2hexOctet
time-high-and-version  = 2hexOctet
clock-seq-and-reserved = hexOctet
clock-seq-low          = hexOctet
node                   = 6hexOctet
hexOctet               = hexDigit hexDigit
hexDigit =
            "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9" /
            "a" / "b" / "c" / "d" / "e" / "f" /
            "A" / "B" / "C" / "D" / "E" / "F"
```

Also, the SIP user agent uses the **sip.rendering** media feature tag defined in [RFC4235] section 5.2. This, in conjunction with procedures described for music-on-hold specified in [MS-SDPEXT] section 3.1.5.27, can be used by SIP **user agents** to signal that the music-on-hold feature is being invoked by including it in the SIP request that initiates music-on-hold.<2>

2.2.6 Via Header Field Extensions

This protocol defines new **Via** header field parameters and their values. The original ABNF, as defined in [RFC5234], for the **Via** header field in [RFC3261] section 25 is extended as follows:

```
via-params = via-ttl / via-maddr / via-received /via-branch
             / via-branched
             / via-ms-internal-info
             / via-ms-received-port
             / via-ms-received-cid
             / via-extension
via-branched = "branched=" ("TRUE" / "FALSE")
via-ms-internal-info = "ms-internal-info=" quoted-string
via-ms-received-port = "ms-received-port=" port
```

```
via-ms-received-cid  = "ms-received-cid=" token
```

token, **quoted-string**, and **port** are defined in [RFC3261] section 25.1.

2.2.7 From and To Header Field Extensions

This protocol defines a new **From** and **To** header field parameter and its value. The original ABNF, as defined in [RFC5234], for the **From** and **To** header fields in [RFC3261] section 25 is extended as follows:

```
from-param = tag-param
             / epid-param
             / generic-param
to-param = tag-param
             / epid-param
             / generic-param
epid-param = "epid=" epid-param-value
epid-param-value = 1*16 tokenchar
tokenchar = (alphanum / "-" / "." / "!" / "%" / "*"
             / "_" / "+" / "`" / "'" / "~" )
```

alphanum is defined in [RFC3261] section 25.

2.2.8 Location Profile Syntax

This section describes the **location profiles** syntax and associated **translation rules** used by the SIP elements to resolve partially specified local phone numbers. The **XML documents** with **location profile descriptions** are delivered as **application/ms-location-profile-definition+xml** content in the body of responses to the SIP **SERVICE** requests, as described in [IETFDRAFT-SIPSOAP-00]. The complete schema is defined in section 7.

2.2.8.1 Location Profile Description Element

Each location profile description MUST include a **Name** element and one or more **Rule** elements. The **Name** element MUST be a string suitable for use as a **phone-context** parameter in the **tel URI**, as defined in [RFC3966] section 3. As specified in [RFC3966], the content of the **tel URI** can also be used as the user portion of a SIP URI.

The location profile description can also contain the following elements:

ExternalAccessPrefix: Element that contains the prefix string that SHOULD be added when dialing external phone numbers.<3>

OptimizeDeviceDialing: Element that, if **true**, indicates to the endpoint using this location profile that the endpoint can do **optimized dialing**. If the value of this element is **false**, the endpoint (5) cannot optimize device dialing when using this location profile.<4>

```
<xsd:complexType name="LocationProfileDescriptionType">
  <xsd:sequence>
    <xsd:element ref="Name" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="Rule" type="RuleType" minOccurs="1" maxOccurs="unbounded"/>
    <xsd:element ref="ExternalAccessPrefix" minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="OptimizeDeviceDialing" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
```

2.2.8.2 Location Profile Rule Element

Each location profile **Rule** element MUST include **Pattern** and **Translation** elements. The **Pattern** element is a regular expression that uses the regular expression syntax defined in [MC-RegEx]. The **Translation** element is a replacement pattern that uses the replacement pattern syntax defined in [MC-RegEx].

The **Rule** element can also contain the following elements:

InternalEnterpriseExtension: Element that, if **true**, indicates that the phone number obtained as a result of applying this rule corresponds to an internal enterprise number. If the value of this element is **false**, the phone number obtained as a result of applying this rule cannot be assumed to be an internal enterprise number.<5>

ApplicableForDeviceDialing: Element that, if **true**, indicates that the device can use the rule for optimized dialing. If the value of this element is **false**, the device cannot use this rule for optimized dialing.<6>

```
<xsd:complexType name="RuleType">
  <xsd:sequence>
    <xsd:element name="Pattern"   type="xsd:string"/>
    <xsd:element name="Translation" type="xsd:string"/>
    <xsd:element name="InternalEnterpriseExtension" type="xsd:boolean" minOccurs="0"/>
    <xsd:element name="ApplicableForDeviceDialing" type="xsd:boolean" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

2.2.9 Routing Script Preamble Syntax

This section specifies the syntax of the routing preamble published by the protocol client in the routing category. The complete schema is defined in section 6.

```
<xs:complexType name="routing-type">
  <xs:annotation>
    <xs:documentation>The name and version attributes are both mandatory.
</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="preamble" type="tns:preamble-type" minOccurs="1" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" />
  <xs:attribute name="version" type="xs:integer" />
  <xs:attribute name="minSupportedClientVersion" type="xs:string" use="optional" />
</xs:complexType>
<xs:element name="routing" type="tns:routing-type" />
```

The routing preamble MUST contain the identification attributes specified in section 2.2.8.1, and MUST contain the preamble element. The preamble provides the data used by the server while routing audio calls sent to the protocol client. The preamble can contain additional elements specified in sections 2.2.8.3 through 2.2.9.5.

If the value of the **version** attribute is 1, the **minSupportedClientVersion** attribute SHOULD NOT be present.<7>

The **minSupportedClientVersion** attribute, if present, SHOULD be ignored while processing an incoming INVITE request. In addition any unknown element or attribute SHOULD be ignored while processing an incoming INVITE request.<8>

2.2.9.1 Identification and Version

The **name** attribute is a string value that provides a scope for the **version** attribute.

2.2.9.2 Target Element

The **target** element specifies a target the call can be routed to. The **uri** attribute, if present, SHOULD be a valid SIP URI. At least one of the **uri** or **application** attributes MUST be present.

Any unknown attributes SHOULD be ignored while processing an incoming INVITE request.<9>

```
<xs:complexType name="target-type">
    <xs:annotation>
      <xs:documentation>At least one of uri or application attributes are
required.</xs:documentation>
    </xs:annotation>
    <xs:attribute name="uri" type="xs:string" use="optional" />
    <xs:attribute name="application" type="xs:string" use="optional" />
    <xs:anyAttribute namespace="##any" processContents="lax" />
</xs:complexType>
```

2.2.9.3 List Element

The **list** element defines a list of **target** elements that are grouped together. Each **list** element SHOULD have a unique **name** attribute and can contain zero or more **target** elements.

```
<xs:complexType name="list-type">
    <xs:complexContent>
      <xs:extension base="tns:preamble-member-base-type">
        <xs:sequence>
          <xs:element name="target" type="tns:target-type" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

2.2.9.4 Flags Element

The **flags** element defines flags that can be used by the script installed on the server. Each **flags** element MUST have a **name** attribute that is unique among all **flags** elements defined in the preamble.

```
<xs:complexType name="flags-type">
    <xs:complexContent>
      <xs:extension base="tns:preamble-member-base-type">
        <xs:attribute name="value" type="xs:string" use="required" />
      </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

2.2.9.5 Wait Element

The **wait** element defines an amount of time in seconds that is referenced by the server while executing the call handling rules defined by the protocol client. This indicates the amount of time the server should wait before executing the next rule. The **name** attribute MUST be unique among all

wait elements. The **seconds** attribute value SHOULD be between 0 and 1,200 seconds (both inclusive).

```
<xs:complexType name="wait-type">
    <xs:complexContent>
      <xs:extension base="tns:preamble-member-base-type">
        <xs:attribute name="seconds" type="xs:nonNegativeInteger" use="required" />
      </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

2.2.10 Ms-Sensitivity Header Field Syntax

This protocol defines a header field called **Ms-Sensitivity** to indicate if a call can be directed to another person or diverted to another device representing the same person. The ABNF, as defined in [RFC5234], for this header is as follows:

```
Ms-Sensitivity = "Ms-Sensitivity" HCOLON ("normal" / "private" /
                 "normal-no-diversion" / "private-no-diversion")
```

A sensitivity of "normal" MUST be assumed if the **Ms-Sensitivity** header field is not present. If the header field contains a value other than one of those specified or appears more than once, a 400 response SHOULD be returned.

HCOLON is defined in [RFC3261] section 25.

2.2.11 Ms-Forking Header Field Syntax

This protocol defines a header field called **Ms-Forking**. The **Ms-Forking** header field indicates to the endpoint that sent the **INVITE** that a proxy is likely to perform either parallel or serial forking, or both based on the called user's routing rules.

```
Ms-Forking = "Ms-Forking" HCOLON "Active"
```

Endpoints can use this information to limit when they accept early media. The **Ms-Forking** header field MUST appear only in 1XX responses.

HCOLON is defined in [RFC3261] section 25.

2.2.12 Ms-Correlation-Id Header Field Syntax

This protocol defines a header field called **Ms-Correlation-Id**. The **Ms-Correlation-Id** header field is used to indicate that multiple SIP dialogs are correlated. This correlation is only used for diagnostic and monitoring purposes. It does not affect the routing behavior of the SIP proxy or endpoints.

```
Ms-Correlation-Id = "Ms-Correlation-Id" HCOLON  UUID
```

HCOLON is defined in [RFC3261] section 25. **UUID** is defined by [RFC4122].

2.2.13 Reason Header Field Extension

This protocol defines a **Reason** header field parameter. The ABNF, as defined in [RFC5234], from [RFC3326] section 2 is extended as follows:

```
Reason              =  "Reason" HCOLON reason-value *(COMMA reason-value)
reason-value        =  protocol *(SEMI reason-params)
protocol            =  "SIP" / "Q.850" / token
reason-params       =  protocol-cause / reason-text
                        / ms-acceptedby-param
                        / reason-extension
ms-acceptedby-param = "ms-acceptedby=" SIPURI
```

SIPURI is defined in [RFC3261] section 25.

2.2.14 Content-Disposition Header Field Extension

This section follows the product behavior described in endnote <10>.

This protocol defines a **Content-Disposition** header field parameter. The ABNF, as defined in [RFC5234], syntax defined in [RFC3261] section 25.1 is extended as follows:

```
Content-Disposition    = "Content-Disposition" HCOLON disp-type
                              *(SEMI disp-param)
disp-type               =  "render" / "session" / "icon" / "alert"
                              / disp-extension-token
disp-param              =  handling-param / ms-proxyfallback-param
                              / generic-param
ms-proxyfallback-param  =  "ms-proxy-2007fallback"
```

2.2.15 Extensions for Federation and Public IM Connectivity

This protocol defines an **ms-edge-proxy-message-trust** header field. This header field can be added by the SIP proxy to any incoming SIP request or **SIP response** from an **external user** to inform the destination protocol client whether the SIP message originates from a remote user, a federated user, or a public IM user. This header field MUST NOT be added by the protocol client.

The ABNF, as defined in [RFC5234], for the **ms-edge-proxy-message-trust** header field is specified as follows:

```
"ms-edge-proxy-message-trust" HCOLON sourcetype-param SEMI epfqdn-param SEMI userverify-
param SEMI sourcenetwork-param SEMI remotefqdn-param

sourcetype-param = "ms-source-type=" ("AuthorizedServer" / "AutoFederation" /
  "DirectPartner" / "EdgeProxyGenerated" / "InternetUser")

epfqdn-param = "ms-ep-fqdn=" pvalue

userverify-param = "ms-source-verified-user="  ( "verified" / "unverified")

sourcenetwork-param = "ms-source-network=" ("federation" / "publiccloud")

remotefqdn-param = "ms-remote-fqdn=" pvalue
```

HCOLON, **SEMI**, and **pvalue** are defined in [RFC3261] section 25.

Details regarding the header field parameters and their values are specified in section 3.10. Example usage for this header field is covered in section 4.9.

2.2.16 Extensions for Remote Users

This protocol defines an **ms-user-logon-data** header field. This header field can be added by the SIP proxy to any outgoing SIP request or response to remote users to inform the destination protocol client that it is connected from outside the enterprise network boundary. A protocol client MUST NOT add the **ms-user-logon-data** header field in any SIP messages sent to the server.

The ABNF, as defined in [RFC5234], for the **ms-user-logon-data** header field is specified as follows:

```
"ms-user-logon-data" HCOLON "RemoteUser"
```

HCOLON is defined in [RFC3261] section 25.

Details regarding the header field parameters and their values are specified in section 3.11. Example use of this header field is covered in section 4.10.

2.2.17 History-Info Header Field extensions

This section follows the product behavior described in endnote <11>.

This protocol defines a **History-Info** header field parameter. The ABNF, as defined in [RFC5234], from [RFC4244] section 4.1 is extended as follows:

```
History-Info           = "History-Info" HCOLON
                         hi-entry *(COMMA hi-entry)
hi-entry               = hi-targeted-to-uri *( SEMI hi-param )
hi-targeted-to-uri     = name-addr
hi-param               = hi-index / hi-ms-retarget-reason / hi-ms-line-type
                         / hi-ms-target-phone / hi-extension
hi-index               = "index" EQUAL 1*DIGIT *(DOT 1*DIGIT)
hi-ms-retarget-reason  = "ms-retarget-reason" EQUAL
                         hi-retarget-reason-val
hi-retarget-reason-val = "forwarding" / "team-call"
                         / "delegation" / token
hi-ms-line-type        = "ms-line-type" EQUAL hi-line-type-val
hi-line-type-val       = "private" / token
hi-ms-target-phone     = "ms-target-phone" EQUAL telephone-uri
hi-extension           = generic-param
```

token is defined in [RFC3261] section 25.1. **telephone-uri** is defined in [RFC3966] section 3.

2.2.18 P-Dialog-Recovery-Action Header Field Syntax

This section follows the product behavior described in endnote <12>.

This protocol defines a **P-Dialog-Recovery-Action** header field. This header can be added by the SIP proxy to a 430 Flow Failed response.

The ABNF, as defined in [RFC5234], for the **P-Dialog-Recovery-Action** header field is as follows:

```
P-Dialog-Recovery-Action = "P-Dialog-Recovery-Action" HCOLON
                           pdr-action *(COMMA pdr-action)
pdr-action               = "Registration-Route-Set-Update"
                           / "Dialog-Route-Set-Update"
                           / "Wait-For-Session-Update"
                           / pdr-action-extension
pdr-action-extension     = token
```

HCOLON is defined in [RFC3261] section 25. **COMMA** and **token** are defined in [RFC3261] section 25.1.

2.2.19 Option Tag extensions

This section follows the product behavior described in endnote <13>.

This protocol defines option tags for use in the **Supported** header field. The new tags extend the set of option tags defined in [RFC3261] section 19.2.

Ms-Dialog-Route-Set-Update: Option tag for support of the dialog route set recovery extension. Inclusion of this tag in the **Supported** header field of the request indicates that the user agent can perform dialog route set recovery, as described in section 3.7.

Ms-Safe-Transfer: Option tag for support of call transfer via SIP REFER request. Inclusion of this tag in the **Supported** header field of the request indicates that the user agent can copy parameters from the **Refer-To** header field URI of the REFER request to the INVITE request, as described in section 3.14.

2.2.20 Call Context Syntax

This section follows the product behavior described in endnote <14>.

This section describes the call context syntax that can be used by SIP elements to convey notes about the current call or the call being transferred. The call context description is delivered as **application/ms-conversation-context+xml** content in the body of a SIP INVITE request to initiate a new call.

```
      <xs:complexType name="XmlConvContextType" >
        <xs:sequence>
          <xs:element name="id" type="xs:token"  minOccurs="1" maxOccurs="1"/>
          <xs:element name="from" type="callctns:XmlConvContextParticipantType" minOccurs="1" 
    maxOccurs="1"/>
          <xs:element name="to" type="callctns:XmlConvContextParticipantType" minOccurs="1" 
    maxOccurs="1"/>
          <xs:element name="participants" 
    type="callctns:XmlConvContextParticipantCollectionType" minOccurs="1" maxOccurs="1" />
          <xs:element name="date" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
          <xs:element name="mode" type="xs:token" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="conversationId" type="xs:token" minOccurs="1" maxOccurs="1"/>
          <xs:element name="dataFormat" type="xs:string" minOccurs="1" maxOccurs="1"/>
          <xs:element name="contextData" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
      </xs:complexType>
```

The complete schema is defined in section 8.

The call context **content type** provides notes about the current call from a server to a protocol client. The call context MUST contain the elements specified in sections 2.2.20.1 through 2.2.20.9, and can contain additional elements specified in section 2.2.20.10.

2.2.20.1 Id Element

The **id** element defines a unique identifier generated by the authoring device, either the protocol client or the server, of the call context data to differentiate one set of call context data from another across all call context generated by a given author. The **id** element MUST be unique among all call context data created by a given author and appear only once in the call context data.

```
    <xs:element name="id" type="xs:token"  minOccurs="1" maxOccurs="1"/>
```

2.2.20.2 From Element

The **from** element describes the author of the call context data that is being conveyed. The **from** element MUST be present in the call context data and appear only once.

```
<xs:element name="from" type="callctns:XmlConvContextParticipantType" minOccurs="1"
maxOccurs="1"/>

  <xs:complexType name="XmlConvContextParticipantType">
    <xs:sequence>
      <xs:element name="uri" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="displayName" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="onBehalfUri" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="onBehalfDisplayName" type="xs:string" minOccurs="0"
maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
```

The **from** element MUST contain a **uri** element representing the author of the call context data, such as sip:alice@contoso.com. The **from** element can also contain the following elements:

- **displayName**
- **onBehalfUri**
- **onBehalfDisplayName**

Child element	Usage
uri	A URI representing the author of the notes, such as sip:alice@contoso.com.
displayName	A plain-text identifier of the author of the notes, such as "Alice".
onBehalfUri	The URI of the user the call context data was authored on behalf of, if created by a third party.
onBehalfDisplayName	The plain-text identifier of the user the call context data was authored on behalf of, if created by a third party.

2.2.20.3 To Element

The **to** element describes the party the call context data was originally conveyed to by the author, who is described by the **from** element. The **to** element MUST be present in the call context data and appear only once.

```
<xs:element name="to" type="callctns:XmlConvContextParticipantType" minOccurs="1"
maxOccurs="1"/>

  <xs:complexType name="XmlConvContextParticipantType">
    <xs:sequence>
      <xs:element name="uri" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="displayName" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="onBehalfUri" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="onBehalfDisplayName" type="xs:string" minOccurs="0"
maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
```

The **to** element MUST contain a **uri** element representing the user the call context data was originally conveyed to by the author of the call context data. The **to** element can also contain the following elements:

- **displayName**
- **onBehalfUri**
- **onBehalfDisplayName**

Child element	Usage
uri	A URI representing the original recipient of the notes, such as sip:alice@contoso.com.
displayName	A plain-text identifier of the original recipient of the notes, such as "Alice".
onBehalfUri	The URI of the user the call context data was original conveyed to on behalf of, if conveyed by a third party.
onBehalfDisplayName	The plain-text identifier of the user the call context data was originally conveyed to on behalf of, if conveyed by a third party.

2.2.20.4 Participants Element

The **participant** element describes a list of one or more parties that were **participants** in the call when the call context data was authored. It MUST be present in the call context data and appear only once.

```
<xs:element name="participants" type="callctns:XmlConvContextParticipantCollectionType"
minOccurs="1" maxOccurs="1" />

  <xs:complexType name="XmlConvContextParticipantCollectionType">
    <xs:sequence>
      <xs:element name="participant" type="callctns:XmlConvContextParticipantType"
minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
```

The **participants** element MUST contain one or more **participant** elements.

2.2.20.5 Participant Element

The **participant** element describes a party involved with the call when the related call context data was authored. A **participant** element MUST be present for the author of the call context data and can be present for other parties in the call.

```
<xs:element name="participant" type="callctns:XmlConvContextParticipantType"
minOccurs="1" maxOccurs="unbounded" />

  <xs:complexType name="XmlConvContextParticipantType">
    <xs:sequence>
      <xs:element name="uri" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="displayName" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="onBehalfUri" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

```
        <xs:element name="onBehalfDisplayName" type="xs:string" minOccurs="0"
maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
```

The **participant** element MUST contain a URI representing the address of a given participant to the call. The **participant** element can also contain the following elements:

- **displayName**
- **onBehalfUri**
- **onBehalfDisplayName**

Child element	Usage
uri	A URI representing a participant (2) of the call related to the call context data, such as "sip:alice@contoso.com".
displayName	A plain-text identifier of the participant (2) identified by the URI, such as "Alice".
onBehalfUri	The URI of the user the participant (2) is acting on behalf of, if the participant (2) is acting in a third-party capacity.
onBehalfDisplayName	The plain-text identifier of the user the participant (2) is acting on behalf of, if the participant (2) is acting in a third-party capacity.

2.2.20.6 Date element

The **date** element provides a **Coordinated Universal Time (UTC)** timestamp that denotes when the author created the call context data. It MUST be present in the call context data and MUST appear only once.

```
<xs:element name="date" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
```

2.2.20.7 ConversationId element

The **conversationId** element provides a correlating identifier between the call context data and the related call that the data was authored for. It MUST be present in the call context data and MUST appear only once.

```
<xs:element name="conversationId" type="xs:token" minOccurs="1" maxOccurs="1"/>
```

The **conversationId** element MUST reflect a unique identifier related to the call that the call context data was authored for.

2.2.20.8 DataFormat element

The **dataFormat** element denotes the **Multipurpose Internet Mail Extensions (MIME)** type format of the **contextData** element in the call context data. It MUST be present in the call context data, and MUST appear only once in the call context data.

```
<xs:element name="dataFormat" type="xs:string" minOccurs="1" maxOccurs="1"/>
```

The **dataFormat** element MUST have a value of "text/plain".

2.2.20.9 ContextData element

The **contextData** element conveys the textual notes about the call that the author created to provide further context about the related call. It MUST be present in the call context data, and MUST appear only once.

```
<xs:element name="contextData" type="xs:string" minOccurs="1" maxOccurs="1"/>
```

The **contextData** element is a free-text element.

2.2.20.10 Mode element

The **mode** element provides an indication of a communications mode that was in use on the call at the time the call context data was authored.

```
<xs:element name="mode" type="xs:token" minOccurs="0" maxOccurs="unbounded"/>
```

The **mode** element can be present one or more times in the call context data, although each **mode** value SHOULD represent a unique modality involved in the call related to the call context data. The following **tokens** are supported:

- **audio**
- **video**
- **im**
- **applicationSharing**

Mode	Meaning
audio	An audio modality was involved for the call relating to the call context data.
video	A video modality was involved in the call relating to the call context data.
im	The instant messaging modality was involved in the call relating to the call context data.
applicationSharing	The application sharing modality was involved in the call related to the call context data.

2.2.21 Ms-Call-Info Header Field Syntax

This protocol defines a header field called **Ms-Call-Info**<15>**.** The **Ms-Call-Info** header field is used to communicate a call property to a client endpoint.

The ABNF, as defined in [RFC5234], or the **Ms-Call-Info** header field is specified as follows:

```
"Ms-Call-Info" HCOLON "Rgs.Anonymization"
```

HCOLON is specified in [RFC3261] section 25. If the header field contains a value other than the one specified, the header SHOULD be ignored.

A server endpoint that performs anonymization SHOULD send this header. The anonymization is provided to the recipient of the header. The identity of the originator of the request can still be shown.

2.2.22 P-Agent-On-Behalf-Of Header Field Syntax

This protocol defines a header field called **P-Agent-On-Behalf-Of**.<16> If a client endpoint attempts to establish a call on behalf of, it MUST use the **P-Agent-On-Behalf-Of** header field.

The ABNF, as defined in [RFC5234], for the **P-Agent-On-Behalf-Of** header field is specified as follows:

```
"P-Agent-On-Behalf-Of" HCOLON name-addr / addr-spec
```

HCOLON, **name-addr** and **addr-spec** are specified in [RFC3261] section 25. This header SHOULD be present only in a SIP INVITE.

The server endpoint can use a back-to-back agent to establish the call. If the server endpoint cannot provide the service, it SHOULD decline the request.

2.2.23 E911 Call Syntax

This section describes the E911 call syntax that MUST be used by SIP endpoints to initiate an E911 call. The SIP INVITE is marked by the presence of a **Priority** header with value "emergency", as specified in [RFC3261] section 20.26, and a **geolocation** header that identifies the content identifier of the call context that is delivered as an **application/pidf+xml** MIME part within the body of the request and a Supported header field containing geolocation. The **geolocation** header is defined in [RFC6442]. The **pidf:presence** element is specified in **Presence Information Data Format (PIDF)**, as specified in [RFC3863], with a **GEOPRIV** location object, as specified in [RFC4119], extension for the status value embedded in it. The **location-info** element embedded in the **GEOPRIV** element MUST conform to the civic location format specified in [RFC5139]. If the address cannot be trusted to match the location of the endpoint initiating the request, the method element embedded in the **GEOPRIV** element MUST have the value "Manual". The **GEOPRIV status** element embedded in the **pidf:presence** element is followed by an **msftE911PidfExtn extension** element, as described in section 9.

For an example E911 INVITE, see section 4.14.

3 Protocol Details

3.1 Common Details

Endpoint Identification Extensions

This protocol provides several mechanisms for identification of SIP endpoints. These mechanisms produce an identifier that carries some or all of the following properties:

- **Long-lived:** Can persist across device, application, or server shutdowns.
- **Distinguishes a specific instance:** Can distinguish a specific endpoint among several endpoints that share the same user or service or application **address-of-record** to maintain per-endpoint state, such as **security association (SA)**, registration state, and presence state, in various SIP elements.
- **Routes to specific instance:** Can be used to address calls to a specific SIP endpoint among several endpoints that share the same user or service or application address-of-record event outside of the SIP transaction.

To maintain compliance with this protocol, the user agent MUST use one of the mechanisms described in sections 3.2, 3.3, and 3.4 to identify each SIP endpoint that it represents.

3.1.1 Abstract Data Model

None.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

None.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 EPID Mechanism Details

The **endpoint identifier (EPID)** mechanism uses an **epid** parameter in the **From** or **To** header fields. When combined with the address-of-record in the **From** or **To** header field, it forms an identifier that carries all of the endpoint identification properties, which are **long-lived**, **distinguishes a specific instance**, and **routes to specific instance**, defined in section 3.

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

User agents are responsible for generating **epid** parameter values in accordance with requirements in section 3.2.3.1; however, the exact mechanism is outside the scope of this protocol. To create a value for an **epid** parameter, the user agent SHOULD use a hexadecimal string no more than 16 hexadecimal characters long. A 64-bit random number or the 8-byte **Media Access Control (MAC) address** of the local network interface card can be encoded as a 16-character hexadecimal string to form a value for an **epid** parameter.

3.2.2 Timers

None.

3.2.3 Initialization

Except as specified in the following sections, the rules for initialization are as specified in [RFC3261].

3.2.3.1 User Agent Initialization

To use the EPID endpoint identification mechanism defined in this section, a user agent MUST obtain an identifier that complies with the **epid-param-value** syntax defined in section 2.2.7 and uniquely identifies itself within all user agents that share the same address-of-record. This identifier SHOULD be persisted across power cycles of the SIP endpoint that the user agent represents.

3.2.4 Higher-Layer Triggered Events

Except as specified in the following sections, the rules for message processing are as specified in [RFC3261].

3.2.4.1 User Agent Operation

To use the EPID endpoint identification mechanism defined in this section, a user agent MUST add the **epid** parameter with a value obtained as described in section 3.2.3 to the **From** header field of every request that it generates, whether or not the request is part of a SIP transaction.

The SIP dialog state created by the user agent that is compliant with this protocol MUST include the **remote epid** parameter in addition to other elements defined in [RFC3261] section 12. For a **user agent client (UAC)**, a **remote epid** is set to the value of the **epid** parameter in the **To** header field, if it is present, and is set to empty if it is not present. For a **user agent server (UAS)**, the **remote epid** parameter is set to the **epid** parameter value in the **From** header field, if it is present, and is set to empty if it is not present.

When forming a request within an existing SIP transaction that contains a non-empty **remote epid** in its state, the user agent that is compliant with this protocol MUST add the **epid** parameter with the value of **remote epid** to the **To** header field.

If the user agent that is compliant with this protocol initiates a call to a specific SIP endpoint, it SHOULD obtain the address-of-record and the value of the **epid** parameter for such an endpoint. The user agent can obtain the address-of-record and the **epid** parameter from the previous dialog with the same endpoint or from the presence document described in [MS-PRES], or it can use any other mechanism. The user agent SHOULD then create a request with the desired address-of-record placed in the **Request-URI** field, place the same address-of-record in the URI of the **To** header field, and add an **epid** parameter to the **To** header field.

3.2.5 Message Processing Events and Sequencing Rules

Except as specified in the following sections, the rules for message processing are as specified in [RFC3261].

3.2.5.1 User Agent Operation

If the **To** header field of the request received by the user agent compliant with this protocol contains an **epid** parameter and its value differs from the user agent's own **epid** parameter value obtained as described in section 3.2.3, the user agent MUST discard the request instead of processing it and generating a response.

3.2.5.2 SIP Registrar Operation

If the REGISTER request processed by the SIP registrar compliant with this protocol contains an **epid** parameter in the **From** header field, the registrar MUST obtain the value of the **epid** parameter and add it to the SIP location service record maintained by this registrar, in addition to the other required information described in [RFC3261] section 10.

3.2.5.3 SIP Proxy Operation

If a SIP proxy compliant with this protocol stores any state associated with SIP endpoints, it SHOULD use the value of the **epid** parameter, if one is present in the **From** or **To** header fields, combined with the address-of-record from the URI of the corresponding header, as an index into its state table. Specifically, the address-of-record and **epid** parameter from the **From** header field SHOULD be used to identify UAC endpoints, and **address-of-record** and **epid** parameters from the **To** header field SHOULD be used to identify UAS endpoints.

If a SIP proxy compliant with this protocol receives a request targeted at the address-of-record that belongs to the domain that this proxy is responsible for, and it is supposed to access a SIP location service to compute the request targets, as specified in [RFC3261] section 16, it MUST perform two additional steps:

1. The SIP proxy MUST examine the **To** header field of the request. If the **To** header field contains an **epid** parameter, the proxy MUST ignore any records returned by the SIP location service that do not have the same **epid** parameter value when computing request targets.

2. If the SIP proxy uses any record returned by the SIP location service as a request target, and the record contains an **epid** parameter value placed there by the SIP registrar, as described in section 3.2.5.2, it MUST add the **epid** parameter value to the **To** header field as an **epid** parameter, unless the **To** header field of the request already has an **epid** parameter. In the latter case, the value in the parameter is expected to be the same as in the SIP location service record; otherwise, the SIP proxy would have ignored the record, as discussed in step 1.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

3.3 SIP.INSTANCE Mechanism

This method is based on [IETFDRAFT-MCICSIP-11]. It employs the **+sip.instance** media feature tag as a **Contact** header field parameter. The value of the **+sip.instance** parameter in combination with the address-of-record in the **From** or **To** header fields forms an identifier that carries the following two properties defined in section 3:

- **Long-lived.**
- **Distinguishes a specific instance.**

It does not carry the **routes to specific instance** property because the **Contact** header field and its parameters are associated with the source, but not the destination, of the message.

This protocol specifies that the user agent MUST use only the UUID URN identifier, as defined in [RFC4122] as its instance identifier in the **+sip.instance** media feature tag.

3.3.1 Abstract Data Model

None.

3.3.2 Timers

None.

3.3.3 Initialization

User agents are responsible for generating **+sip.instance** parameter values in accordance with the requirements in section 3.3.3.1; however, the exact mechanism is outside the scope of this protocol. To create a value for the **+sip.instance** parameter, a user agent can use methods described in [IETFDRAFT-MCICSIP-11] section 4. Specifically, the user agent can use the methods of UUID URN computation based on time, unique names such as MAC address, or a random number generator, which are defined in [RFC4122].

Except as specified in the following sections, the rules for initialization are as specified in [RFC3261].

3.3.3.1 User Agent Initialization

To use the **SIP.INSTANCE** endpoint identification mechanism defined in this section, a user agent MUST obtain a UUID using any of the procedures described in [RFC4122]. However, if the same user agent also uses the EPID mechanism, as described in section 3.2, it MUST compute an EPID namespace UUID using the algorithm for name-based UUID described in [RFC4122] section 4.3, with specific constants and algorithm choices applicable to the EPID namespace defined in this protocol.

To compute an EPID namespace:

1. Allocate a UUID to use as a namespace ID for all UUIDs generated from names in that namespace. For UUIDs in the EPID namespace defined in this protocol, the following UUID has been allocated:

   1. fcacfb03-8a73-46ef-91b1-e5ebeeaba4fe

2. Choose the **SHA-1 hash** algorithm described in [FIPS180].

3. Convert the EPID value to a canonical sequence of octets, which for the EPID namespace has been defined as ASCII encoding of the **epid** parameter value as it appears in the **From** or **To** header field of the SIP message.

4. Compute the **hash** of the namespace ID concatenated with the name.

5. Set octets zero through 3 of the **time_low** field to octets zero through 3 of the hash.

6. Set octets zero and 1 of the **time_mid** field to octets 4 and 5 of the hash.

7. Set octets zero and 1 of the **time_hi_and_version** field to octets 6 and 7 of the hash.

8. Set the four most significant bits, which are bits 12 through 15, of the **time_hi_and_version** field to the 4-bit version number, as specified in [RFC4122] section 4.1.3. For name-based UUIDs computed with the **SHA-1** function, this sequence is 0101.

9. Set the **clock_seq_hi_and_reserved** field to octet 8 of the hash.

10. Set the two most significant bits, which are bits 6 and 7, of the **clock_seq_hi_and_reserved** to zero and 1, respectively.

11. Set the **clock_seq_low** field to octet 9 of the hash.

12. Set octets zero through 5 of the node field to octets 10 through 15 of the hash.

13. Convert the resulting UUID to local byte order.

In the previous procedure, the UUID obtained SHOULD be persisted across power cycles of the SIP endpoint that the user agent represents.

3.3.4 Higher-Layer Triggered Events

Except as specified in the following sections, the rules for message processing are as specified in [RFC3261].

3.3.4.1 User Agent Operation

To use the **SIP.INSTANCE** endpoint identification mechanism defined in this section, the user agent MUST add the **+sip.instance** parameter with an obtained UUID URN value, as described in section 3.3.3, to the **Contact** header field of the messages which carry the **Contact** header field because of SIP protocol requirements. [RFC3261] requires the addition of the **Contact** header field to the dialog creating requests and responses and a REGISTER request. The **+sip.instance** parameter syntax is defined in section 2.2.5.

3.3.5 Message Processing Events and Sequencing Rules

Except as specified in the following sections, the rules for message processing are as specified in [RFC3261].

3.3.5.1 SIP Registrar Operation

If a REGISTER request processed by a SIP registrar compliant with this protocol contains a **+sip.instance** parameter in the **Contact** header field, the registrar MUST obtain the **+sip.instance** parameter value and validate that it conforms to the UUID URN syntax described in [RFC2141] and

[RFC4122]. Furthermore, if the REGISTER request also contains an **epid** parameter in the **From** header field, the registrar MUST validate that the name-based UUID, derived as described in section 3.3.3 from the **epid** parameter value, is equal to the UUID extracted from the **+sip.instance** parameter value.

If either of these validations fails, the registrar MUST reject the REGISTER request with a 400 response code. Otherwise, the registrar MUST add the UUID value that is extracted from the **+sip.instance** parameter value to the SIP location service record maintained by this registrar in addition to the other required information described in [RFC3261] section 10.

3.3.5.2 SIP Proxy Operation

If a SIP proxy compliant with this protocol stores any state associated with SIP endpoints, it SHOULD use the value of the UUID from the **+sip.instance** parameter in the **Contact** header field, if one is present, combined with the address-of-record from the URI of the **From** or **To** header field as an index into its state table. Specifically, the UUID from the **+sip.instance** parameter and the address-of-record from the **From** header field SHOULD be used to identify the UAC endpoint in requests, and the UUID from the **+sip.instance** and address-of-record from the **To** header field SHOULD be used to identify the UAS endpoint in each response.

Before the UUID from the **+sip.instance** parameter is used, the SIP proxy MUST obtain the value of the **+sip.instance** parameter and validate that it conforms to the UUID URN syntax specified in the [RFC2141] and [RFC4122]. Furthermore, if the message is a request and it also contains an **epid** parameter in the **From** header field or the message is a response and it also contains an **epid** parameter in the **To** header field, the SIP proxy MUST validate that the name-based UUID derived as described in section 3.3.3 from the **epid** parameter value is equal to the UUID extracted from the **+sip.instance** parameter value. If validation fails, the proxy SHOULD respond with 400 response code.

3.3.6 Timer Events

None.

3.3.7 Other Local Events

None.

3.4 GRUU Mechanism

This method is based on [IETFDRAFT-OUGRUAUSIP-10] and uses the GRUU to provide an identifier that carries all of the properties, which are **long-lived**, **distinguishes a specific instance**, and **routes to specific instance**, defined in section 3. As described in [IETFDRAFT-OUGRUAUSIP-10] section 6, only the SIP registrar authoritative for the domain can generate the GRUU for all addresses-of-record that belong to the domain and user agents MUST use either a SIP registration procedure or some other protocol or administrative mechanism to obtain a GRUU.

3.4.1 Abstract Data Model

None.

3.4.2 Timers

None.

3.4.3 Initialization

Except as specified in the following sections, the rules for initialization are as specified in [RFC3261].

3.4.3.1 User Agent Initialization

To use a GRUU-based endpoint identification mechanism defined in this section, a user agent MUST obtain a GRUU from a SIP registrar using either the registration procedure defined in [MS-SIPREGE] or, if the user agent is a part of a server application or a conferencing endpoint, it can obtain a GRUU using an administrative method outside the scope of this protocol.

3.4.4 Higher-Layer Triggered Events

Except as specified in the following sections, the rules for message processing are as specified in [RFC3261].

3.4.4.1 User Agent Operation

To use the GRUU-based endpoint identification mechanism defined in this section, a user agent MUST use the GRUU that it previously obtained, as described in section 3.4.3.1, to populate the URI in the **Contact** header field of the messages which would otherwise carry the **Contact** header field because of SIP protocol requirements. [RFC3261] requires the addition of the **Contact** header field to the dialog creating the requests. Although [RFC3261] also requires the presence of a **Contact** header field in the REGISTER request, the GRUU MUST NOT be used to populate it.

When using GRUU as a URI in the **Contact** header field, the user agent can also add a **grid** URI parameter to the **Contact** header field with a value that satisfies the syntax defined in section 2.2.2. As noted in [IETFDRAFT-OUGRUAUSIP-10] section 8.1.1, the user agent can manufacture an infinite supply of GRUUs, each of which differs by the value of the **grid** parameter. When a user agent receives a request that was sent to the GRUU, it is able to tell which GRUU was invoked by looking at the **grid** parameter.

When sending a request that contains a GRUU in the **Contact** header field, the user agent compliant with this protocol MUST forward it to a SIP registrar or proxy in the same domain as the one from which the user agent obtained the GRUU.

If the same user agent also uses the EPID mechanism, as described in section 3.2, and it uses the registration procedure defined in [MS-SIPREGE] to obtain the GRUU, it MUST insert the same **epid** parameter value into the **From** header field of every request as the one it used when performing the registration.

3.4.5 Message Processing Events and Sequencing Rules

Except as specified in the following sections, the rules for message processing are as specified in [RFC3261].

3.4.5.1 SIP Registrar Operation

A SIP registrar compliant with this protocol can generate a GRUU by creating a SIP URI with an address-of-record in the domain that the registrar is responsible for as the user and domain portion. It then MUST add a mandatory **GRUU** parameter, and it SHOULD add an additional **opaque** parameter with a value that encodes information about one the following entities:

- the user agent type and an identifier of a specific endpoint bound with the user agent address-of-record, as specified in [RFC3261] section 10.2.1,
- an instance of an application endpoint,

- an instance of a server endpoint.

When generating a GRUU for a user agent that follows the registration procedure defined in [MS-SIPREGE], the registrar can create a URI using ABNF, as defined in [RFC5234], for **user-agent-gruu** syntax, as defined in section 2.2.3. The address-of-record value in the ABNF comes from the URI in the **To** header field. The ABNF for **ua-opaque-val** syntax is defined in section 2.2.2, where **encoded-uuid-val** value is obtained by applying an encoding procedure to the binary form of the UUID obtained from the **+sip.instance** parameter of the **Contact** header field. The encoding procedure MUST produce a string that satisfies the syntax of a SIP **URI** parameter, as defined in [RFC3261] section 25. One example of an encoding procedure is defined in [RFC3548] section 4.

When generating a GRUU for an application that implements voice mail service for a user, the registrar can create a URI using ABNF for **voice-mail-gruu** syntax, as defined in section 2.2.3. The address-of-record value in the ABNF MUST belong to the user whose voice mail service is represented by the GRUU. The ABNF **app-voicemail-opaque-val** syntax is defined in section 2.2.2.

When generating a GRUU for an application that implements location profile service for a user, the registrar can create a URI using ABNF for **location-profile-gruu** syntax, as defined in section 2.2.3. The address-of-record value in the ABNF MUST belong to the user whose location profile service is represented by the GRUU. The ABNF **app-locationprofile-opaque-val** syntax is defined in section 2.2.2.

When generating a GRUU for a multimedia **conference** endpoint created by the user agent that follows the procedure for conference creation defined in [MS-CONFBAS], the registrar can create a URI using ABNF for **conf-endpoint-gruu** syntax, as defined in section 2.2.3. The address-of-record value in the ABNF MUST be associated, as specified in [RFC3261] section 10.2.1, with the user that organized the conference. The ABNF for **app-conf-opaque-val** syntax is defined in section 2.2.2, where **conf-entity-val** value describes the type of conferencing endpoint. The **encoded-conf-id-val** value can be obtained by applying the procedure defined in [RFC3548] section 4 to the binary form of conference identifier, which is defined in [MS-CONFPRO] section 2.2.1.2.

When generating a GRUU for a server deployed within a domain for which a SIP registrar is responsible, the registrar can create a URI using ABNF for **server-instance-gruu** syntax defined in section 2.2.3. The **server-fqdn** value in the ABNF is a **fully qualified domain name (FQDN)** of the server. The **domain-fqdn** value is the FQDN of the domain for which the SIP registrar is responsible. The ABNF for **server-opaque-val** syntax is defined in section 2.2.2, where **server-type-val** value describes the type of service provided by the server with the **HomeServer** string representing the SIP registrar and presence server, the **MRAS** string representing the media relay **authentication** server, the **MediationServer** string representing the mediation server, and a **QoSM** string representing the quality of service monitoring server. The **encoded-server-instance-val** value can be obtained by applying the procedure defined in [RFC3548] section 4 to the binary form of the **GUID** that is associated with the server instance entry in **Active Directory**.

When a SIP registrar compliant with this protocol creates a SIP location service record for user agents that use the registration procedure defined in [MS-SIPREGE], it MUST generate a GRUU that satisfies all of the following requirements:

- When a request is sent to the GRUU, it routes to a SIP proxy with access to the SIP location service record that this registrar creates.
- The GRUU MUST include the **gruu** URI parameter.
- If the GRUU contains an **opaque** URI parameter, the URI that results from stripping out the **opaque** and **gruu** URI parameters MUST be equivalent to the address-of-record for which the SIP location service record is created.

The registrar then MUST store the GRUU with the SIP location service record that it creates as the result of the registration procedure in addition to other information described in [RFC3261] section 10. It MUST also return the GRUU to the user agent requesting it as a part of the registration procedure defined in [MS-SIPREGE] section 3.1. The registrar can also use other methods of delivering GRUUs to user agents that represent server application or conferencing endpoints in the registrar domain.

3.4.5.2 SIP Proxy Operation

If a SIP proxy compliant with this protocol stores any state associated with SIP endpoints, it SHOULD use the value of the GRUU, if one is present in the **Contact** header field, as an index into its state table. Specifically, the GRUU from the **Contact** header field of SIP request messages SHOULD be used to identify UAC endpoints, and the GRUU from the **Contact** header field of SIP response messages SHOULD be used to identify UAS endpoints.

If a SIP proxy compliant with this protocol receives a request outside of the dialog, with no **Route** header fields, targeted at the URI that belongs to the domain that this proxy is responsible for, and it is supposed to access a SIP location service so that it can compute the request targets, as specified in [RFC3261] section 16, it MUST examine the target URI of the request.

For example, the **Request-URI** field is examined. If the URI contains a **gruu** parameter, and thus is a GRUU, and the URI does not refer to any GRUU known in the domain, the proxy rejects the request with a 404 response.

The proxy MUST ignore any records returned by the SIP location service that do not have the same GRUU value when computing request targets.

If the SIP proxy uses any record returned by the SIP location service as a request target, it MUST copy the **grid** parameter and its value from the original target URI, or GRUU, into the new target URI obtained from the SIP location service record. If the original target URI did not contain a **grid** parameter or the parameter value was empty, the proxy MUST insert a **grid** parameter value into the new target URI.

If a SIP proxy compliant with this protocol receives a mid-dialog request with **Route** header fields and a **Request-URI** field that belongs to the domain that this proxy is responsible for, and the proxy has access to the SIP location service in the domain, it MUST examine the URI and the **Request-URI** field. If the URI contains a **gruu** parameter, which means that it is a GRUU, and the URI does not refer to any GRUU known in the domain, the proxy MUST reject the request with a 404 response.

The proxy MUST contact the SIP location service for the domain for records where the address-of-record in the record matches the address-of-record in the URI and, from the returned set of records, select the records that have the same **GRUU** value that appears in the **Request-URI**.

If at least one record is selected:

- The SIP proxy MUST arbitrarily choose one of the selected records as a new request target. It MUST then copy the **grid** parameter and its value from the original target URI (**GRUU**) into the new target. If the original target URI did not contain the **grid** parameter or the parameter value was empty, the proxy MUST insert a **grid** parameter value into the new target URI.
- If there are no **Route** headers in the request after the proxy removes the topmost **Route** header pointing to it, as specified in [RFC3261] section 16.4, the proxy MUST copy all routing information from the selected SIP location service record to the **Route** header of the request.

If no records were selected, the proxy SHOULD reject the request with a 480 Temporarily Unavailable response.

3.4.6 Timer Events

None.

3.4.7 Other Local Events

None.

3.5 Firewall and Network Address Translation Traversal Aid Extensions

When a user agent forms a connection to a SIP proxy, SIP registrar, or other SIP servers and that connection traverses a firewall or a NAT device, the server might be unable to make a connection back to the user agent because of the firewall or NAT device. Because, during normal SIP operation, servers have to send responses back to the user agent, as well as initiate and forward requests destined to the user agent, the transport layer on the SIP server has to route messages to the user agent over the existing connection established from the user agent. To aid the transport layer on the SIP server in routing messages over the connection from the protocol client, this protocol defines mechanisms that help save connection identification information in **Via, Contact, Record-Route,** and **Path** header fields of the incoming SIP requests. The header fields described in this protocol are designed to preserve routing information for use by the transport layer. Specifically, the following list of header fields serves this purpose:

- **Via** header fields MUST be copied from the SIP requests to responses, as specified in [RFC3261] section 8.2.6.2.
- **Contact** and **Record-Route** header fields MUST be preserved in dialog state, as specified in [RFC3261] section 12.1.1, and copied to mid-dialog requests, as specified in [RFC3261] section 12.2.1.1.
- **Contact** and **Path** header fields are saved in the SIP location service database for the user agent's domain, as specified in [RFC3327] section 5.3, and then inserted into the requests forwarded by the SIP proxies authorized for the domain, as specified in [RFC3327] section 5.4.

3.5.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

[RFC3261] section 18 specifies that the transport layer of every SIP element is responsible for managing persistent connections over the **Transmission Control Protocol (TCP)** and other connection-oriented transport protocols and then index them based on the **tuple** formed from transport address, port, and protocol of the far end of the connection. Far end is defined in [RFC3261] section 18 as the destination for connections opened by the transport layer and as a source for connections accepted by the transport layer.

If a TCP connection accepted by the transport layer traverses a NAT device, the address and port in the tuple of the far end of the connection belong to the NAT device, and not to the user agent. If the original user agent disconnects for any reason, and another user agent is allocated the same address and port, the transport layer of the SIP element cannot distinguish the new user agent from the old user agent. To avoid misidentifying the connection, the transport layer of the SIP element can maintain a counter that gets incremented with each created connection, and can make this counter a part of the tuple that indexes connections. The counter is of sufficient length that it does not wrap around before the end of the lifetime of all transactions, dialogs, and SIP location service records that were created based on the messages that had the value identifying the connection populated into their header fields.

3.5.2 Timers

None.

3.5.3 Initialization

None.

3.5.4 Higher-Layer Triggered Events

Except as specified in the following sections, the rules for message processing are as specified in [RFC3261].

3.5.4.1 User Agent Operation

To use the firewall and NAT device traversal mechanism defined in this section, the user agent MUST add a **proxy** parameter with the value "replace" to the **Contact** header field of the messages that carry the **Contact** header field because of SIP protocol requirements and when the URI in the **Contact** header field contains the user agent's IP address in its host portion or as the value of the **maddr** parameter. The exact syntax for the **proxy** parameter is defined in section 2.2.5, and the syntax for the SIP URI, including the host portion and the **maddr** parameter, is defined in [RFC3261] section 25.1.

3.5.5 Message Processing Events and Sequencing Rules

Except as specified in the following sections, the rules for message processing are as specified in [RFC3261].

3.5.5.1 SIP Server (Proxy, Registrar) Operation

When a SIP proxy, SIP registrar, or any SIP server compliant with this protocol receives a message that has a **Contact** header field with the **proxy** parameter, it MUST perform the following steps in addition to the processing described in the [RFC3261]:

1. If the server is not the first node after the user agent, it MUST reject the message with a 400 response if the message is a request, and then discard the message if it is a response. The SIP server can determine if it is the first hop by examining the **Via** header field. More than one value in this field indicates that the SIP server is not the first hop.

2. If the **proxy** parameter in the **Contact** header field has any value other than "replace", the server MUST reject the message with a 400 response if message is a request, and discard the message if it is a response.

3. If the URI in the **Contact** header field has a **transport** parameter and the value of this parameter is not the same as the transport protocol of the connection over which the message was received, the server MUST reject the message with a 400 response if the message is a request, and discard the message if it is a response.

4. The server MUST remove the **proxy** parameter and its value from the **Contact** header field.

5. If the URI in the **Contact** header field has a **maddr** parameter, the server MUST replace its value with the value of the IP address of the far end of the connection on which the message was received.

6. If the URI in the **Contact** header field does not have a **maddr** parameter and the host portion of the URI is not an IP address, such as a host name, the server MUST add a **maddr** parameter with the value of the IP address of the far end of the connection on which the message was received to the **Contact** header field.

7. If the URI in the **Contact** header field does not have a **maddr** parameter and the host portion of the URI is an IP address and its value is not the same as the value of the IP address of the far end

of the connection on which the message was received, the server MUST replace the host portion of the URI with the value of the IP address of the far end of the connection on which the message was received.

8. If the URI in the **Contact** header field does not have a port portion or if the port portion value is not the same as the value of the port of the far end of the connection on which the message was received, the server MUST add the port or replace its value with the value of the port of the far end of the connection on which the message was received.

9. The server MUST add a parameter with a value that uniquely identifies the connection on which the message was received among all other connections that were or could in the future be established by the server with the same tuple (address, port, and transport) on the far end to the URI of the **Contact** header field. The server can use the **ms-received-cid** parameter for this purpose and populate it with the value of the counter described in section 3.5.1.

10. If the server is a SIP proxy, it MUST insert the **Record-Route** header field into the message, as described in [RFC3261] section 16, to remain on the path of all the subsequent messages in the dialog that is created by the message.

The syntax for a SIP URI, including host and port portions and a **maddr** parameter, is defined in [RFC3261] section 25.1.

When a SIP server compliant with this protocol processes a request from another SIP element, it SHOULD save the identification information of the connection on which it received the request in the topmost **Via** header field. To do this, the server SHOULD use the following **Via** header field parameter values:

- **received** parameter value, as defined in [RFC3261] section 25.1, to save the IP address of the far end of the connection.
- **ms-received-port** parameter value, as defined in section 2.2.6, to save the port number of the far end of the connection.
- **ms-received-cid** parameter value, defined in section 2.2.6, to save unique connection identifiers, which are values that uniquely identify the connection on which the message was received among all other connections that were or could in the future be established by the server with the same tuple (address, port, and transport. The server can populate **ms-received-cid** with the value of the counter described in section 3.5.1.

3.5.6 Timer Events

None.

3.5.7 Other Local Events

None.

3.6 Extensions for Reliable and Consistent Message Routing Within Redundant Server Network

Messages between user agents in a SIP element network traverse a set of one or more servers or proxies that run and provide services such as network edge traversal, authentication, call data records, and message content archiving. It is often essential for the SIP protocol itself, as well as for the services provided by the SIP proxies, that the related messages, such as responses to the requests or all messages in the dialog, traverse the same set of proxies in a specific order. Furthermore, core functionality of the SIP proxy, such as routing, as well as potential services that it runs and provides depend on the capability to propagate contextual information between related messages. For example, the transport layer of the SIP proxy that adds the **received** parameter to the

Via header field in the request depends on the availability of this parameter in the response to route the response.

[RFC3261] defines two basic mechanisms that ensure that the response follows the path of the request in reverse order, which are a mechanism to insert and process the **Via** header field, and that all requests in the dialog traverse the proxies that specifically chose to be on the dialog's path, which are a mechanism to insert **Record-Route** header fields, store them in the dialog route set, and populate request **Route** header fields from the dialog route set. This protocol compliments these basic mechanisms with the following additional specific functions:

- Storing references to the information that spans the lifetime of multiple SIP transactions and dialogs, such as references to data associated with the identity represented by the user agent.
- Storing information about specific services provided by the SIP proxies within the context of the dialog.
- Storing the FQDN of a specific server in a set of multiple redundant SIP proxies sharing the same common FQDN that handles messages in the dialog.
- Ensuring that the essential context information in the **Via** or **Record-Route** header fields that the proxy inserted into the message or information in the **Via**, **Record-Route**, and **Contact** header fields inserted by other SIP elements was preserved and populated correctly without modifications into related messages by the user agents.

3.6.1 Abstract Data Model

None.

3.6.2 Timers

3.6.2.1 SIP Proxy Operation

If the SIP proxy uses a **Hash-based Message Authentication Code (HMAC)** algorithm, as described in [FIPS198a], to protect the integrity of the **Record-Route**, **Contact**, or **Via** headers and it periodically changes the key used in the HMAC computation, as recommended by [FIPS198a], or if it uses a similar algorithm that depends on periodically updated keys, the proxy MUST start a timer per key when the key is last used to compute the HMAC before it gets changed and it MUST retain the key until the timer fires. The timer SHOULD fire no earlier than 1 hour after it is started for keys used to protect information in **Via** and **Record-Route** header fields that are copied from the request to the response. The timer SHOULD fire no earlier than 8 hours for keys used to protect information in **Contact** and **Record-Route** header field URIs that is preserved in the dialog route set and used to populate **Route** header fields in mid-dialog requests.

3.6.3 Initialization

The SIP proxy SHOULD create one or more tables to maintain the information that spans the lifetime of the dialog and then store an index to this type of table in the **Record-Route** header field that it inserts into the dialog-creating messages. Specifically, the SIP proxy SHOULD create a table of endpoints that user agents communicating with the proxy represent.

Consequently, the SIP proxy SHOULD add an index to an entry in the **endpoint** table as a value of the **ms-opaque** parameter in the **Record-Route** header field **URI** which this proxy inserts into the messages, as described in [RFC3261] section 16. When the **Record-Route** header field **URI** is then stored in the dialog route set, and later copied to the **Route** header field of the mid-dialog request, the value of the **ms-opaque** parameter represents the identity of the UAS endpoint.<17>

Furthermore, the SIP proxy SHOULD add an index to an entry in the **endpoint** table as a value of the **ms-identity** parameter of the **Record-Route** header field **URI** which this SIP proxy inserts into the messages, as described in [RFC3261] section 16. When the **Record-Route** header field **URI** is then stored in the dialog route set and later copied to the **Route** header field of the mid-dialog request, the value of the **ms-identity** parameter can represent the identity of the UAC endpoint.<18>

The SIP proxy can add **ms-role-rs-to** or **ms-role-rs-from** parameters to the **Record-Route** header field **URI** so that when the **Record-Route** header field **URI** is stored in the dialog route set, and later copied to the **Route** header field of the mid-dialog request, the **ms-role-rs-to** parameter indicates that this SIP proxy is an authorized proxy for the UAS endpoint domain while the **ms-role-rs-from** parameter indicates that the SIP proxy is an authorized proxy for the domain of the UAC endpoint.<19>

If the SIP server is a member of a set of multiple redundant proxies that appear to share the same FQDN with some or all other SIP elements that communicate with them, the SIP server can add its specific unique FQDN as the value of the **ms-fe** parameter of the **Record-Route** or **Contact** header field **URI** so that when the **Record-Route** or **Contact** header field **URI** is stored in the dialog route set, and later copied to the Request-URI field or **Route** header field of the mid-dialog request, the **ms-fe** parameter contains the unique FQDN of the server.

The SIP proxy can add an **ms-ent-dest** parameter to the **Record-Route** header field **URI** so that when the **Record-Route** header field **URI** is stored in the dialog route set, and later copied to the **Route** header field of the mid-dialog request, the **ms-ent-dest** parameter indicates that if the SIP proxy is an authorized proxy for the domain of the UAC endpoint, the UAS endpoint belongs to the same domain.<20>

The SIP proxy can combine all state information that it maintains for the endpoints in the dialog that spans the lifetime of the dialog, encode it using a method that produces output that satisfies the SIP **URI** parameter syntax, such as the method defined in [RFC3548] section 4, and add it as a value of an **opaque** parameter to the **Record-Route** header field **URI** that this SIP proxy inserts into messages, as described in [RFC3261] section 16.<21> When the **Record-Route** header field **URI** is then stored in the dialog route set, and later copied to the **Route** header field of the mid-dialog request, the **opaque** parameter value can be decoded and all of the information that the proxy previously stored can be made available to it.<22>

3.6.4 Higher-Layer Triggered Events

None.

3.6.5 Message Processing Events and Sequencing Rules

3.6.5.1 SIP Proxy Operation

If the SIP proxy uses an HMAC algorithm, as specified in [FIPS198a], to protect the integrity of the **Record-Route** or **Contact** header fields, and it periodically changes the key used in the HMAC computation, as recommended by the [FIPS198a], or if it uses a similar algorithm that depends on periodically updated keys, and it receives a SIP request that contains the HMAC that the SIP proxy previously inserted, and the SIP proxy no longer has the key to compute the HMAC, the SIP proxy SHOULD reject the request with a 481 Call Leg Does Not Exist response.<23> However, if the SIP proxy implements the extensions for dialog state recovery, as described in section 3.7, it SHOULD follow the procedure defined there to send a 430 Flow Failed or a 481 Call Leg Does Not Exist response.<24>

3.6.6 Timer Events

When the timer described in section 3.6.2.1 fires, the SIP proxy can destroy the key for which the timer was started. The SIP proxy SHOULD then reject all requests that contain an HMAC generated with the destroyed key with a 481 Call Leg Does Not Exist response, as described in section 3.6.5.1.<25> However, if the SIP proxy implements the extensions for dialog state recovery, as described in section 3.7, it MUST follow the procedure defined there to send a 430 Flow Failed or a 481 Call Leg Does Not Exist response.<26>

3.6.7 Other Local Events

None.

3.7 Extensions for Dialog State Recovery in Case of Outages in SIP and other Network Elements on the Dialog Path

This section follows the product behavior described in endnote <27>.

To achieve reliability of message delivery between SIP endpoints, typical installations deploy sets of redundant SIP proxies and other network elements, such as firewalls or NAT devices, providing an alternate path to process and route traffic between endpoints in cases of unplanned or scheduled outages. However, as described in section 3.5 and section 3.6, both SIP and other network elements often maintain state information that they associate directly or indirectly, through indexing, with the SIP dialog state, and when the main SIP proxy or other network device goes out of service, the alternate, or redundant, element, which does not have the corresponding state, cannot continue processing or routing messages. This protocol defines extensions that allow SIP proxies to communicate to the endpoints that the SIP dialog state carried in the mid-dialog messages no longer has necessary information. It also provides a mechanism for endpoints to update, or recover, the dialog state without breaking the SIP dialog and associated media, such as audio or video, session.

A SIP endpoint can register with its SIP registrar via one or more SIP proxies, as specified in [RFC3261] and [MS-SIPREGE]. If the SIP registrar gets recycled because of unplanned or scheduled outages, the binding information associated with the SIP endpoint can be lost. In such a scenario, SIP message delivery to the endpoint is impacted until the client re-registers and recreates the registration binding. If the SIP endpoint tries to establish a new dialog with another SIP endpoint, mid-dialog messages are not deliverable until the SIP endpoint refreshes its registration binding. This protocol defines extensions that allow SIP registrars to communicate to the endpoints (5) that the SIP registration binding is no longer valid. It also provides a mechanism for endpoints (5) to update the registration binding without breaking any other SIP dialogs and associated media sessions that it is participating in.<28>

3.7.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

3.7.1.1 SIP Proxy Operation

Section 3.7.5.1 describes a way for a SIP proxy to associate the state information needed to process and route mid-dialog messages with the dialog route set. This state information can include references to transport connection identifiers, SAs, and endpoint registration information, and can be used by the SIP proxy to detect that referenced information is either missing or invalid because it was created and maintained by another redundant SIP proxy.

3.7.1.2 User Agent Operation

A user agent supporting the dialog state recovery can keep states for recovery mode and can remember state for transaction retries specified in section 3.7.5.3 for dialogs where recovery is enabled.

3.7.2 Timers

3.7.2.1 User Agent Operation

If a user agent enables recovery procedures described in this section for a specific SIP dialog for which it also negotiated a session timer as described in [RFC4028], it SHOULD start a recovery refresh timer upon creation, with the interval set to at least the interval it negotiated for the session timer.

3.7.3 Initialization

3.7.3.1 User Agent Operation

A user agent compliant with this specification SHOULD enable recovery procedures for dialogs where loss of communications on SIP signaling path leads to loss of valuable state and content information, such as media state and content in an audio call, that cannot be easily recovered. User agents SHOULD NOT enable the recovery procedures for dialogs where state and content can be seamlessly restored by creation of the replacement dialog, such as the presence subscription dialog described in [MS-PRES].

3.7.4 Higher-Layer Triggered Events

3.7.4.1 User Agent Operation

If a user agent enables recovery procedures described in this section for a specific SIP dialog, it MUST include the **Ms-Dialog-Route-Set-Update** option tag in the **Supported** header field of all the requests in the dialog.

The user agent SHOULD negotiate a mechanism to periodically refresh the dialog with recovery procedures enabled. For INVITE based dialogs, the user agent SHOULD use the session timer mechanism described in [RFC4028]. For **SUBSCRIBE** based dialogs, the user agent SHOULD use the subscription refreshes described in [RFC3265]. Regardless of the specific refresh mechanism chosen by the user agent, all dialog refresh requests MUST be target refresh requests specified in [RFC3261] section 6.

3.7.5 Message Processing Events and Sequencing Rules

3.7.5.1 SIP Proxy Operation

When a SIP proxy receives a mid-dialog request and it extracts references to the state information, such as transport connection identifier, security association, or endpoint registration information, that it previously encoded into the dialog route set, as described in section 3.7.1.1, the SIP proxy SHOULD check if the corresponding state information is available and valid for request processing and routing. If the information is no longer available or cannot be used to process and route the mid-dialog request, the proxy MUST perform the following steps:

1. Check if the **Ms-Dialog-Route-Set-Update** option tag is present in the **Supported** header field of the request. If the **Ms-Dialog-Route-Set-Update** option tag is NOT present, the SIP proxy

SHOULD reject the request with a 481 Call Leg Does Not Exist response and stop further processing.

2. If the **Ms-Dialog-Route-Set-Update** option tag is present, the SIP proxy MUST reject the request with a 430 Flow Failed response and add a **P-Dialog-Recovery-Action** header field. The value of the **P-Dialog-Recovery-Action** header field indicates the actions that either the source or destination endpoint of the currently processed mid-dialog request needs to take to make processing or routing possible for subsequent requests in the dialog. The value of the **P-Dialog-Recovery-Action** header field MUST be set as follows:

   Dialog-Route-Set-Update: The proxy can recover if the source endpoint of the mid-dialog request performs a dialog recovery procedure, as described in section 3.7.5.3.4.

   Registration-Route-Set-Update, **Dialog-Route-Set-Update:** The proxy determines that it can recover if the source endpoint of the current request first refreshes its registration, as described in [RFC3261] section 10.2.4, and then performs a dialog recovery procedure, as described in section 3.7.5.3.4.

   Wait-For-Session-Update: The proxy determines that it can recover if the destination endpoint of the current request in the dialog either refreshes its registration or sends the target refresh request in the dialog.

3.7.5.2 SIP Registrar Operation

When a SIP registrar receives a dialog creating request from a SIP endpoint, it MUST<29> check if the **Contact** header specifies the GRUU of the endpoint, as specified in section 3.4.5.1. If it does, it MUST check whether the SIP endpoint registration is valid and the **Routable** flag is set to "TRUE", as specified in [MS-SIPREGE] section 3.1.2.1. If the binding is absent or the **Routable** flag is set to "FALSE", it SHOULD reject the request with a 430 Flow Failed response and add a **P-Dialog-Recovery-Action** header field. The value of the **P-Dialog-Recovery-Action** indicates the actions that the source endpoint of the currently processed dialog creating the request needs to take to make processing or routing possible for requests originating from, or destined to, that endpoint. The value of the **P-Dialog-Recovery-Action** header field MUST be set to "Registration-Route-Set-Update, Dialog-Route-Set-Update".

3.7.5.3 User Agent Operation

The following sections document message processing events and sequencing rules for user agent operations for the dialog state recovery extensions.

3.7.5.3.1 Processing 430 (Flow Failed) Responses

When a user agent receives a 430 Flow Failed response for a mid-dialog request and the response contains a **P-Dialog-Recovery-Action** header field, the user agent MUST examine the value of this field to decide if it needs to perform dialog recovery procedures. Based on the value, the user agent takes the following actions:

- If the **P-Dialog-Recovery-Action** header field contains a **P-Dialog-Recovery-Action** tag, the user agent MUST indicate the failure to the upper layer and then perform registration refresh, as described in [RFC3261] section 10.2.4, on the endpoint that received the 430 Flow Failed response. If registration is successfully refreshed, the user agent MUST execute dialog recovery procedures, as described in section 3.7.5.3.4, on all dialogs associated with the registered endpoint (5) that have dialog recovery enabled. The user agent SHOULD also terminate and re-create all dialogs associated with the registered endpoint (5) that did not have dialog recovery enabled.

- If the **P-Dialog-Recovery-Action** header field contains a single **Dialog-Route-Set-Update** tag, the user agent MUST perform a dialog recovery procedure described in section 3.7.5.3.4. If the refresh request for the dialog recovery procedure results in a successful response, the user agent MUST re-send the request that resulted in the 430 Flow Failed response with the route set and **Request-URI** field populated from the updated route set and remote target fields in the dialog state. If the refresh request for the dialog recovery procedure does not result in a successful response, the user agent MUST indicate the failure of the original request to the upper layer.

- If as the result of performing dialog recovery procedures, the same request is re-sent two or more times and it again receives a 430 Flow Failed response, the user agent SHOULD stop retrying the same request and report the failure to the user. If the **P-Dialog-Recovery-Action** header field contains a single **Wait-For-Session-Update** tag and the user agent has negotiated a session timer, as described in [RFC4028] on the dialog, it SHOULD start or reset the recovery refresh timer with the interval set to at least the interval it negotiated for the session timer.

When a user agent receives a 430 Flow Failed response for a dialog creating request and the response contains a **P-Dialog-Recovery-Action** header field, the user agent MUST examine the value of this field to decide if it needs to perform dialog recovery procedures<30>. Based on the value, the user agent takes the following actions:

- If the **P-Dialog-Recovery-Action** header field contains a **P-Dialog-Recovery-Action** tag, the user agent MUST indicate the failure to the upper layer and then perform registration refresh, as described in [RFC3261] section 10.2.4, on the endpoint that received the 430 Flow Failed response. If the registration is successfully refreshed, the user agent MUST execute dialog recovery procedures, as described in section 3.7.5.3.4, on all dialogs associated with the registered endpoint (5) that have dialog recovery enabled. The user agent SHOULD also terminate and recreate all dialogs associated with registered endpoints (5) that did not have dialog recovery enabled. Finally, it SHOULD re-send the dialog creating request that originally received the 430 response.
- If as the result of performing dialog recovery procedures, the same request is re-sent two or more times and it again receives a 430 Flow Failed response, the user agent SHOULD stop retrying the same request and report the failure to the user.

3.7.5.3.2 Processing Registration Refresh Responses

When a user agent refreshes endpoint registration, as described in [MS-SIPREGE], and receives a successful response containing a **Presence-State** header field with a **register-action-value** of "added" or "fixed", the user agent SHOULD execute dialog recovery procedures, as described in section 3.7.5.3.4, on all dialogs associated with the registered endpoint (5) that have dialog recovery enabled. The user agent SHOULD also terminate and recreate all dialogs associated with registered endpoints (5) that did not have dialog recovery enabled.

3.7.5.3.3 Processing Mid- Dialog Refresh Requests

When a user agent receives a session refresh request, as described in [RFC4028], on a dialog that has recovery procedures enabled, it SHOULD start or reset the recovery refresh timer with the interval set to at least the interval it negotiated for the session timer.

When a user agent receives a mid-dialog target refresh request, as described in [RFC3261] section 6, on a dialog that has recovery procedures enabled, it SHOULD extract the URIs from the **Contact** and **Record-Route** header fields in the request and update the route set and remote target field in the dialog state. If the user agent does not update the route set and remote target, subsequent outgoing requests are sent with a stale route and result in a 430 Flow Failed response.

3.7.5.3.4 Dialog Recovery Procedure

The user agent MUST execute the following steps to recover the dialog state:

1. The user agent MUST construct and send an appropriate target refresh request for the dialog. For example, the user agent sends an UPDATE request for an INVITE dialog or a SUBSCRIBE request

for a SUBSCRIBE dialog. The user agent then waits for completion of the associated SIP transaction. The target refresh request MUST carry a value, as specified in section 2.2.2.2, in the **Contact** header field and **Record-Route** header fields.

2. If the transaction initiated by the target refresh request succeeds, the user agent MUST extract the URIs from the **Contact** and **Record-Route** header fields in the response and update the route set and remote target field in the dialog state.

3. If the target refresh fails with a 430 Flow Failed response that carries a **P-Dialog-Recovery-Action** header field with a single **Wait-For-Session-Update** tag as its value, the user agent SHOULD start or reset the recovery refresh timer with the interval set to at least the interval it negotiated for the session timer.

When the dialog recovery procedure succeeds for a given dialog, the user agent SHOULD also initiate recovery procedures for other dialogs that are logically related to the recovered dialog. For example, the user agent initiates dialog recovery for the dialogs in the conference, as described in [MS-CONFBAS], when it recovers one of them.

3.7.6 Timer Events

3.7.6.1 User Agent Operation

When the recovery refresh timer defined in section 3.7.2.1 fires, the user agent MUST execute dialog recovery procedures, as described in section 3.7.5.3.4.

3.7.7 Other Local Events

None.

3.8 Phone Number Resolution Extensions

[RFC3966] defines a notion of a Local Number as a phone number that is only valid within a certain geographical area or certain part of the telephony network. As specified in [RFC3966] section 5.1.5, Local Numbers SHOULD only be used in the environment where all local entities can successfully set up the call by passing this Local Number to dialing software.

This protocol provides a way to create such an environment, and employs a notion of location profile to describe it. Each location profile description carries a set of translation rules that resolve partially specified (local) numbers to identifiers which either route to unique enterprise users or form unique numbers in public telephone networks as defined by International Telecommunications Union Recommendation, contained in [E164]. A translation rule, in turn, is a tuple consisting of the regular expression that matches a subset of local numbers and a replacement pattern that provides an identifier that is no longer tied to a geographical area or part of the telephony network. This type of replacement identifier can be used for routing to a specific enterprise user or for identifying a subscriber in the public telephone network. The regular expressions and replacement patterns are based on .NET Regular Expression Language, as specified in [MC-RegEx]. In addition to defining the location profiles and translation rules that comprise them, this protocol describes a protocol that can be used by the protocol clients to obtain these profiles from the server.

3.8.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

3.8.1.1 User Agent Operation

A user agent compliant with this protocol SHOULD obtain the name of the default location profile to use with the partially specified phone numbers entered by the user. It SHOULD also obtain location profile descriptions with the set of translation rules to convert the partially specified local phone numbers that it receives in SIP messages from other SIP elements.

3.8.1.2 SIP Proxy Operation

A SIP proxy compliant with this protocol SHOULD maintain location profile descriptions for all local geographical areas that it serves. It SHOULD also maintain a database that maps each address-of-record in the domain for which it is responsible to a location profile description, effectively establishing a default location profile for each user.

3.8.2 Timers

None.

3.8.3 Initialization

3.8.3.1 User Agent Operation

A user agent compliant with this protocol SHOULD obtain the name of the default location profile. It SHOULD use the **in-band provisioning** protocol defined in [MS-SIPREGE] section 3.2

3.8.4 Higher-Layer Triggered Events

3.8.4.1 User Agent Operation

To obtain a location profile description, the user agent MUST send a SIP SERVICE request, as specified in [IETFDRAFT-SIPSOAP-00], with the following parameters:

- The **Request-URI** field and **To** header field **URI** MUST be set to the **location-profile-gruu**, as defined in section 2.2.3, whose address-of-record matches the address-of-record that the user agent represents. If the form of the **location-profile-gruu** that contains the **default** URI parameter is used, the default location profile description for the address-of-record is returned. Otherwise, the location profile description for the profile specified in the **phone-context** parameter is returned.
- The **From** header field URI MUST be set to the address-of-record that the user agent represents.
- The **Accept** header field MUST be set to **application/ms-location-profile-definition+xml**.
- Other fields of the SERVICE request MUST be set as described in [RFC3261] and [IETFDRAFT-SIPSOAP-00], and the request MUST be sent using the rules in [RFC3261].

3.8.5 Message Processing Events and Sequencing Rules

3.8.5.1 SIP Proxy Operation

When a SIP proxy compliant with this protocol receives a SERVICE request targeted to a URI built according to **location-profile-gruu** syntax, as described in section 2.2.3, whose address-of-record matches an address-of-record in the domain for which this SIP proxy is responsible, it MUST process the request as follows:

1. Perform standard routing procedures against the **Request-URI** field, as described in [RFC3261]. One of the standard routing procedures in [RFC3261] specifies that it MUST respond with a 404 response if the address-of-record in the **Request-URI** field does not exist in the domain that the proxy is responsible for.

2. Extract the name of the location profile from the **location-profile-gruu** URI. If the **location-profile-gruu** URI contains the **default** parameter, the proxy SHOULD consult its internal database to determine the name of the location profile whose address-of-record matches the address-of-record in the **location-profile-gruu** URI. Otherwise, it MUST extract the name of the location profile from the **phone-context** URI parameter. If neither the **default** or **phone-context** parameters are present in the **location-profile-gruu** URI, the SIP proxy MUST reject the request with a 485 Ambiguous response.

3. The SIP proxy MUST then check its location profile descriptions database and attempt to locate the profile with the name extracted in Step 2. If the location profile description with the given name does not exist, the SIP proxy MUST reject the request with a 404 Not Found response. Otherwise, it MUST read the location profile description from its database and form an XML document according to the syntax described in section 2.2.8.

4. The proxy MUST form and send the response to the SERVICE request as described in [RFC3261] and [IETFDRAFT-SIPSOAP-00] and insert the following fields:

    1. The **Content-Type** header field MUST be set to **application/ms-location-profile-definition+xml**.

    2. The body of the response MUST be set to the location profile description XML document created in step 3.

3.8.6 Timer Events

None.

3.8.7 Other Local Events

None.

3.9 Extensions for Call Processing and Routing Based on Routing Script Preamble and Call Designation Parameters

This protocol specifies the Routing Script Preamble mechanism for protocol client endpoints (5) to publish rules for routing INVITEs targeted to the address-of-record of the user the user agent represents. The preamble MUST be published by the user agent into the **routing** category, as specified in [MS-PRES] section 2.2.2.7.7, and is used for all audio INVITEs except those that are exposed to policy restrictions on the server.

The user agent can publish preambles into multiple instances of the routing category. The different preambles MUST meet the following conditions:

- Each preamble publication MUST be in accordance with the preamble **XSD**.
- **List** elements with the same name can appear in multiple instances. The **name** attribute value of all **list** elements occurring in the same instance MUST be unique.
- The **name** attribute values of all other elements MUST be unique within that element type. For example, the preambles cannot contain two **wait** elements with the same name.

If any of the preceding conditions are not met, a server that is a SIP proxy authorized for the domain of the target user's address-of-record SHOULD use a default routing script that routes only to the registered endpoints of the target address-of-record.

If the server finds multiple instances that are valid, it MUST generate an aggregated preamble that is then used for routing. If multiple **list** elements with the same name are found, the aggregated preamble SHOULD contain one **list** with that name containing all of the **target** elements from different instances. If the **version** attribute of the instances are different, the aggregated preamble's version MUST be the highest **version** attribute value among all instances.

The preamble published by the protocol client SHOULD match a corresponding script installed on the server (2). If no match is found, a server (2) that is a SIP proxy authorized for the domain of the target user's address-of-record SHOULD use a default routing script that routes only to the registered endpoints (5) of the target address-of-record.

If any element required by the script is not present in the preamble, the server (2) can reject the INVITE with a 480 response.

3.9.1 Abstract Data Model

None.

3.9.2 Timers

3.9.2.1 Registered Endpoints Timer

If the call is being routed to the registered endpoints whose address-of-record matches the address-of-record in the **Request-URI** field, a registered endpoints timer is started. The amount of time to wait is defined by the **wait** element named **total**, as specified in section 3.9.5.1.3, that is defined in the preamble. If no preamble is published, the default wait time is 20 seconds. If a preamble is published but a **wait** element named **total** is not defined, the default wait time is 15 seconds.

3.9.2.2 Call Forwarding Timer

If call forwarding is enabled, which means that the **enablecf** flag is set, as specified in section 3.9.5.1.2, and the call is routed to the target in the **forwardto** list, as specified in section 3.9.5.1.4, the call forwarding timer is started for 60 seconds.

3.9.2.3 Primary User Timer

This section follows the product behavior described in endnote <31>.

If team ringing is enabled, a primary user timer is started instead of the registered endpoints timer. The amount of time to wait is defined by the **wait** element named **user**, as specified in section 3.9.5.1.3, that is defined in the preamble. If a preamble is published but a **wait** element named **user** is not defined, the default wait time is 15 seconds.

3.9.2.4 Secondary Target Timer

This section follows the product behavior described in endnote <32>.

If the call is being routed to the targets in the **team** or **delegates** list, as specified in section 3.9.5.1.4, a secondary target timer is started. The amount of time to wait is defined by the **wait** element named **team2**, as specified in section 3.9.5.1.3, that is defined in the preamble. If a preamble is published but a **wait** element named **team2** is not defined, the default wait time is 0 seconds.

3.9.3 Initialization

The default routing behavior for a SIP proxy authorized for the domain of the target user's address-of-record if no preamble is published by the protocol client or if the preamble name and version do not match is to ring registered endpoints for 20 seconds and then forward the call to the target user's voice mail, if it is configured.

3.9.4 Higher-Layer Triggered Events

None.

3.9.5 Message Processing Events and Sequencing Rules

3.9.5.1 Call Processing and Routing Elements

User agents that are publishing can publish any preamble that is in accordance with the preamble XSD. However, the server SHOULD only act on a specific list of elements, and other elements MUST be ignored. The server that is a SIP proxy authorized for the domain of the target user's address-of-record SHOULD apply the routing rules based on the preamble only for INVITEs that meet one of the following criteria:

- The Content-Type header field has the value "application/SDP" and the **Session Description Protocol (SDP)** body includes **audio**.
- The Content-Type header field contains the string "application/ms-conf-invite" and the request body is an XML document that contains an **XML element** named "audio". The said element should also contain an **XML attribute** named "available" with value "true".<33>
- The content type is "multipart/MIME" and at least one part contains an SDP body that includes audio.<34>

The construction of the INVITE requests with an "application/SDP" content type is described in [RFC3264], the "multipart/MIME" content type is defined [RFC2046].

All other INVITEs SHOULD be routed as specified in [RFC3261]**.** The routing mechanism specified in this section is applicable only if one of the preceding three conditions is met.

An INVITE whose **Content-Type** header field contains the string **application/ms-conf-invite** and the XML body indicates that audio is available is called an **audio app-invite**.

3.9.5.1.1 Routing Element Name and Version

The **routing** element has **name** and **version** attributes that SHOULD be one of the supported values. The supported values for these attributes are the following:

- The **name** attribute value is **rtcdefault** and the **version** attribute is 1.
- The **name** attribute value is **rtcdefault** and the **version** attribute is 2.<35>

3.9.5.1.2 Routing Element Flags

The server MUST use the **flags** element named **clientflags** to determine which features are currently enabled or disabled. Any other **flags** element or flags in **clientflags** element MUST be ignored by the server. The following table describes how each flag is used. The "Working hours only" column indicates if the flag can be used in conjunction with the **work_hours** flag.

Flag name	Usage	Working hours only
block	Causes all calls to the user to fail. This flag SHOULD be the only value present in a preamble intended to block inbound calls.	No
work_hours	Indicates that the routing logic SHOULD only be applied if the current time falls within the **calendarData** publication, as specified in [MS-PRES] section 2.2.2.7.8.	Not Applicable
forward_immediate	Causes calls to be forwarded to the address specified in the **forwardto** list if the **enablecf** flag is also present, or to voice mail if the **enablecf** flag is not present.	Yes
simultaneous_ring	Causes the first target listed in the **list** element named **simultaneous_ring** to be called at the same time any registered endpoints are called.	Yes
enablecf	Enables call forwarding to the target in the **forwardto** list. This flag is used to toggle between activating voice mail and call forwarding.	Yes
delegate_ring<36>	Indicates that the call SHOULD be forked to the targets specified in the **delegates** list. This flag SHOULD NOT be used in combination with **team_ring**. If **team_ring** is set at the same time, **team_ring** takes precedence. This flag is applicable only if the **routing** element version is 2.	Yes
team_ring<37>	Indicates that the call SHOULD be forked to the targets specified in the **team** list. This flag is applicable only if the **routing** element version is 2.	Yes
skip_primary<38>	Indicates that the registered endpoints and simultaneous ring device of the **callee** SHOULD NOT be rung unless the call is coming from or transferred by a URI in the **breakthrough** or **delegates** list. This flag is applicable only if the **routing** element version is 2. This flag is applicable only if the **delegate_ring** flag is also set.	Yes
forward_audio_app_invites<39>	Indicates that audio app-invites, as described in section 3.9.5.1, SHOULD be routed in the same way as all other audio invites to this user. This flag is applicable only if the **routing** element version is 2.	Yes
e911active<40>	Causes all routing rules to be suspended and calls to be forked only to registered endpoints (5). This is set by the client when the user makes an emergency call.	No

3.9.5.1.3 Routing Element Wait

The server MUST use only the **wait** element with names defined as follows. All other **wait** elements are ignored.

Wait name	Usage
total	Number of seconds to wait for the called party to answer. Used when routing version is 1 or when version is 2 and **team_ring** and **delegate_ring** flags are not set.
user<41>	Number of seconds to ring the user’s registered endpoints and simultaneous

Wait name	Usage
	ring device before ringing the team. Applicable only if routing version is 2.
team1<42>	Reserved for future use. SHOULD be ignored.
team2<43>	Number of seconds to ring the team or **delegates**. Applicable only if routing version is 2.

3.9.5.1.4 Routing Element Lists

The server MUST use only the lists specified in the following table. These lists can be empty if there is no relevant data provided by the user. All other **list** elements published by client endpoints are ignored.

List name	Usage
forwardto	This list contains the URI that SHOULD be used when the user has selected call forwarding, which means that the **enablecf** is set under **clientflags**. Even though the **list** element syntax allows more than one item, the list SHOULD contain only one entry. If more than one entry is present, the server SHOULD only use the first destination.
simultaneous_ring	This list contains the URI that defines a device that SHOULD ring at the same time as the user's registered devices. Even though the **list** element syntax allows more than one item, the list SHOULD contain only one entry. If more than one entry is present, the server SHOULD only use the first destination.
team<44>	This list contains the URIs corresponding to the team members of the user. This list is applicable only if the routing version is 2.
delegates<45>	This list contains the URIs corresponding to the delegates of the user. This list is applicable only if the routing version is 2.
first_delegate<46>	Reserved for future use. SHOULD be ignored.
breakthrough<47>	List of identities that can ring the user directly even when the **skip_primary** flag is set. This is applicable only if routing version is 2.
add_voice<48>	Reserved for future use. SHOULD be ignored.

3.9.5.2 Incoming INVITE Processing

When an INVITE arrives at the SIP proxy authorized for the address-of-record in the Request-URI field, the proxy MUST process the request based on the preamble published for that address-of-record.

3.9.5.2.1 Ms-Sensitivity Header

The presence of the **Ms-Sensitivity** header field in the incoming request is used to tailor how the request is routed.

Level of sensitivity	Usage
normal	This is the default value. All possible destinations will be selected by the server subject to the routing rules as specified by the preamble.
normal-no-diversion	This has the effect of disabling voice mail and call forwarding. If the **Ms-Sensitivity** header has this value, the server MUST NOT route the call to voice mail or the call forwarding target defined in the **forwardto** list or to the targets defined in the **team** list. Note that calls to the simultaneous ring target are not considered a diversion and the call MUST be forwarded to the simultaneous ring target if present.
private	Reserved for future use. MUST be treated the same way as **Normal**.
private-no-diversion	MUST be treated the same way as **normal-no-diversion**.

3.9.5.2.2 Rules for Handling the INVITE

The SIP proxy authorized for the address-of-record in the Request-URI field SHOULD perform the following steps in order when handling the INVITE request:

1. If the **block** flag is set, the proxy SHOULD reject the request with a 480 Temporarily Unavailable response, and further processing of rules SHOULD be stopped.
2. If the **e911active** flag is set, the proxy SHOULD route the call only to registered endpoints. The registered endpoints timer SHOULD NOT be started and further processing of rules SHOULD be stopped.<49>
3. If the INVITE is an audio app-invite and the **forward_audio_app_invites** flag is not set, the proxy SHOULD route the call only to registered endpoints. The registered endpoints timer SHOULD NOT be started and further processing of rules SHOULD be stopped.<50>
4. If the INVITE is targeted at the **private line** of the user, the call SHOULD be processed as specified in section 3.9.5.2.2.4.
5. If the INVITE was routed to the user as a result of team or delegate ringing processing for some other user, the proxy SHOULD route the call only to registered endpoints and the registered endpoints timer SHOULD NOT be started. Further processing of rules SHOULD be stopped.<51>
6. If the address-of-record in the URI of the **From** or **Referred-By** header fields, as defined in [RFC3892] section 3, is present in the **breakthrough** list, the call SHOULD be routed to the primary targets as specified in section 3.9.5.2.2.1, and further processing of rules SHOULD be stopped.<52>
7. If the **work_hours** flag is set and the current time is outside the working hours in the **calendarData** publication, as specified in [MS-PRES] section 2.2.2.7.8, the call MUST be forked to the registered endpoints whose address-of-record matches the address-of-record in the **Request-URI** field, except that **do-not-disturb** presence state MUST be handled as specified in step 10.
8. If the **team_ring** flag is set, team ringing SHOULD be processed as specified in section 3.9.5.2.2.3 and further processing of rules SHOULD be stopped<53>

9. If the **delegate_ring** flag is set, delegate ringing SHOULD be processed as specified in section 3.9.5.2.2.2 and further processing of rules SHOULD be stopped.<54>

10. If the user's presence published in the **state** for the container to which the **caller** belongs, as described in [MS-PRES], is "do-not-disturb", the call MUST be routed to the target user's voice mail and further processing of rules SHOULD be stopped. If the call cannot be routed to voice mail because of **Ms-Sensitivity** header field value considerations described in section 3.9.5.2.1, a response indicating failure SHOULD be returned.

11. If none of the preceding conditions apply, the call MUST be routed to primary targets as specified in section 3.9.5.2.2.1.

3.9.5.2.2.1 Ringing Primary Targets

If in the processing of the INVITE based on the routing rules, the proxy decides to ring the primary targets, the following actions MUST be taken:

- If the **forward_immediate** flag is set in the protocol client flags:
  - The call SHOULD be routed to the destination in the **forwardto** list or voice mail depending on whether the **enablecf** flag is set.
  - If a **simultaneous_ring** target exists, it MUST NOT be honored if the **forward_immediate** flag is set.
  - If the call was routed to the target in the **forwardto** list, the call forwarding timer MUST be started. If the call cannot be routed because of the **Ms-Sensitivity** header field value considerations described in section 3.9.5.2.1, a response indicating failure SHOULD be returned.
- Otherwise, if the **forward_immediate** flag is not set in the protocol client flags:
  - The call MUST be forked to the registered endpoints whose address-of-record matches the address-of-record in the **Request-URI** field.
  - If the **simultaneous_ring** flag is set, the INVITE MUST be routed to the target specified in the **simultaneous_ring** list. The proxy MUST then start the registered endpoints timer.

3.9.5.2.2.2 Delegate Ringing

This section follows the product behavior described in endnote <55>.

If in the processing of the INVITE based on the routing rules, the proxy decides to honor delegate ringing, the following actions MUST be taken:

- If the address-of-record in the URI of the **From** or the **Referred-By** header field is present in the **delegates** list, the INVITE MUST be routed to primary targets, as specified in section 3.9.5.2.2.1.
- If the user's presence published in the **state** category for the container to which the caller belongs, as described in [MS-PRES], is "do-not-disturb", the call MUST be forked to the targets present in the **delegates** list and the secondary target timer MUST be started.
- If the user’s presence state is not "do-not-disturb", the call MUST be routed to all the registered endpoints of the user and the primary user timer MUST be started. <56>
- If the user’s presence state is not "do-not-disturb", the call MUST be routed to all of the targets present in the **delegates** list. The secondary target timer MUST be started.

3.9.5.2.2.3 Team Ringing

This section follows the product behavior described in endnote <57>.

If in the processing of the INVITE based on the routing rules, the proxy decides to honor team ringing, the following actions MUST be taken:

- If the address-of-record in the URI of the **From** field or the **Referred-By URI** field is present in the **team** list, the INVITE MUST be routed to primary targets as specified in section 3.9.5.2.2.1.
- If the user's presence published in the **state** category for the container to which the caller belongs, as described in [MS-PRES], is "do-not-disturb", the call MUST be forked to the targets present in the **team** list and the secondary target timer MUST be started.
- If the user's presence state is not "do-not-disturb", the call MUST be routed to all the registered endpoints of the user. The primary user timer MUST be started.

3.9.5.2.2.4 Ringing Private Line

This section follows the product behavior described in endnote <58>.

If the incoming INVITE is targeted at the private line of the user, the call MUST be forked to the registered endpoints whose address-of-record matches the address-of-record of the target. In addition, if the **simultaneous_ring** flag is set, the INVITE MUST be routed to the target specified in the **simultaneous_ring** list. The proxy MUST then start the registered endpoints timer.

3.9.5.3 Handling 303 Response

Any destination to which the call is forked can send a 303 Proxy Redirect response back to the server. [IETFDRAFT-RCDPR-303-01] specifies how this response is handled.

3.9.5.4 Handling 605 Response

Any destination to which the call is forked can send a 605 Decline All response back to the server. [IETFDRAFT-SF-605-01] specifies how this response is handled.

3.9.5.5 Handling 415 Response

This section follows the product behavior described in endnote <59>.

If a SIP proxy compliant with this protocol receives a 415 response from one of the targets to which the proxy forked the call, the proxy MUST handle the response as follows:

1. If the request that was sent to the target did not contain a body with a "multipart/MIME" content type, no special processing is applied and the 415 response MUST be handled as any 4XX response, as described in [RFC3261], section 16.7.
2. If **multipart/MIME** retry has been attempted for this target, the 415 response MUST be handled as any 4XX response.
3. If any **Accept** header in the response indicates that the UAS supports **multipart/MIME**, no special processing is applied and the 415 response MUST be handled as any 4XX response.
4. If any part of a **multipart/MIME** body has a **Content-Disposition** header field with an **ms-proxy-2007fallback** parameter and that part has SDP content with **media description for audio media type (SDP content and media descriptions defined in** [RFC4566]**)**, the proxy takes the following actions:
   1. The proxy MUST re-send the INVITE to the target with only the SDP body, and

2. The proxy MUST update its call context for that target to indicate that **multipart/MIME** retry has been attempted for this target.

The **multipart/MIME** content type is defined in [RFC2046].

3.9.5.6 Handling 2XX Responses

A SIP proxy compliant with this protocol SHOULD handle 2XX responses according to proxy behavior described in [RFC3261] section 16.7. In addition, the CANCEL requests sent out as a result of a 2XX response SHOULD have an **ms-acceptedby** parameter in the Reason header field. The **ms-acceptedby** parameter value SHOULD be set to the address-of-record of the destination user agent that sent the 2XX response.

3.9.5.7 Other Responses

All other responses SHOULD be treated as specified in [RFC3261].

3.9.5.8 Generating 199 Response

This section follows the product behavior described in endnote <60>.

If a proxy receives a non 2XX final response from one of the targets and the SIP proxy decides to keep or drop the final response, the proxy SHOULD generate a 199 response in accordance with [IETFDRAFT-RCITD-199-01] if:

1. A 18X response from that target had been proxied through to the caller, and

2. A 199 response was not already sent for this target.

3.9.5.9 1XX Responses Generated

Any time the SIP proxy authorized for the domain in the address-of-record of the **Request-URI** field processes an audio call as described in this protocol, a 183 response with an **Ms-Forking** header field MUST be sent back to the caller.

Any time the request was sent to one or more registered endpoints, a 101 response MUST be sent back to the caller.

Any time the request was forwarded to a target other than the registered endpoints (5), a 181 response MUST be sent back to the caller.

3.9.5.10 History-Info Header Field Processing

This section follows the product behavior described in endnote <61>.

When the SIP proxy authorized for the domain in the address-of-record of the Request-URI field processes the INVITE request using the published preamble, as described in section 3.9.5.2, it MUST process the **History-Info** header field in the request, if present, as follows:

1. The proxy MUST perform basic validation of the **History-Info** header field entries according to the syntax in section 2.2.17 so that it can extract the value of the **hi-index** parameter of the last entry. If validation of the **History-Info** header field fails, the proxy MUST stop further processing. The proxy can reject the request with a 480 response.

2. If validation of the **History-Info** header field succeeds, the proxy MUST store the value of the **History-Info** header field except the last entry, which is the entry targeted at the address-of-

record for which the proxy processes the INVITE request, in the INVITE transaction processing context.

3. The proxy MUST also extract the value of the **hi-index** parameter from the last entry and store it in the INVITE transaction processing context.

If a **History-Info** header field is not present in the request, the proxy MUST store an empty **History-Info** header field and **hi-index** parameter value of 1 in the INVITE transaction processing context.

The proxy MUST also initialize a value of branch index to 1 in the INVITE transaction processing context.

When, as part of processing the INVITE transaction, the INVITE request is proxied or forwarded to any destination, the SIP proxy MUST copy the **History-Info** header field that it stored in the INVITE transaction processing context to the proxied or forwarded request and append one or more **History-Info** header field entries as follows:

- If the destination is a registered endpoint whose address-of-record matches the address-of-record of the target of the original INVITE request or the INVITE request is forked to the destination at the same time as it is being sent to the registered endpoints, the proxy MUST add one **History-Info** header field entry with a **hi-targeted-to-uri** parameter set to the SIP URI of the registered endpoint address-of-record, and a **hi-index** parameter set to the current value of the **hi-index** parameter in the INVITE transaction processing context.

- If the destination is a registered endpoint whose address-of-record matches the address-of-record of the target and the request was targeted at the private line of the user, the proxy SHOULD add a **hi-ms-line-type** parameter with the value "**private**"<62>.

  In addition, the proxy SHOULD add a **hi-ms-target-phone** parameter with the phone line associated with the user as a TEL URI as its value<63>.

- For other destinations, the proxy MUST add two **History-Info** header field entries:

1. An entry with the parameters set as follows:

   hi-targeted-to-uri value MUST be set to the SIP URI of the address-of-record of the target in the original INVITE request.

   hi-index parameter value MUST be set to the current value of the **hi-index** parameter in the INVITE transaction processing context.

   hi-ms-retarget-reason parameter value MUST be set to the value of **team-call** if the current destination was selected as the result of team ringing, or to the value of **delegation** if the current destination was selected as the result of delegate ringing, or to the value of **forwarding** in all other cases.

   hi-ms-target-phone parameter value MUST be set to the phone line associated with the target user as a TEL URI<64>.

   reason parameter MUST NOT be set if the request is being sent to a registered endpoint of the target or if the INVITE request is being sent to the current destination while any previous forks to registered endpoints are still active (as is the case with simultaneous ringing, delegate ringing or team-call as described in section 3.9.5.2.2.1, section 3.9.5.2.2.2, and section 3.9.5.2.2.3). The **reason** parameter MUST be set to the value "SIP;cause=303;text=Redirect**"** if the INVITE request is forwarded to the current destination as the result of the processing of a 303 response, as described in section 3.9.5.3, or with the value of "SIP;cause=302;text=Moved Temporarily" if the INVITE request is forwarded to the current destination for any other reason. The **reason** parameter is an optional parameter for

History-Info header field, reflected in the History-Info header by including the **reason** header escaped in the **hi-targeted-to-uri**. The **reason** parameter MAY use the **Reason** header extensions defined in this specification.

2. An entry with the parameters set as follows:

   hi-targeted-to-uri parameter value MUST be set to the SIP URI of the address-of-record of the destination.

   hi-index parameter value MUST be set to the concatenation of a) the current value of the **hi-index** parameter in the INVITE transaction processing context , b) the "." separator, and c) the current value of the branch index in the INVITE transaction processing context.

   The proxy MUST then increment by 1 the value of the branch index in the current INVITE transaction processing context.

When, as part of processing the INVITE transaction, the proxy generates a 181 response, it MUST add a **History-Info** header field with a single entry with the parameters set as follows:

- **hi-targeted-to-uri** parameter value MUST be set to the SIP URI of the address-of-record of the target in the original INVITE request.
- **hi-index** parameter value MUST be set to the value of 1.
- **hi-ms-retarget-reason** parameter value MUST be set to the value of **team-call** if the 181 response was generated when the original INVITE was sent to the destination as the result of team ringing, or to the value of **delegation** if the 181 response was generated when the original INVITE was sent to the destination as the result of delegate ringing, or to the value of **forwarding** in all other cases.
- **reason** parameter MUST NOT be set if the INVITE request is being sent to the current destination while any previous fork to registered endpoints are still active. The **reason** parameter MUST be set to the value of "SIP;cause=303;text=Redirect**"** if the INVITE request is forwarded to the current destination as the result of the processing of a 303 response, as described in section 3.9.5.3, or with the value of "SIP;cause=302;text=Moved Temporarily" if the INVITE request is forwarded to the current destination for any other reason.

3.9.6 Timer Events

3.9.6.1 Registered Endpoint Timer Expiry

When the registered endpoint timer expires, the following actions MUST be executed by the server:

If the **Ms-Sensitivity** header field value does not contain **no-diversion** and the incoming INVITE is not targeted at the private line of the user and the **enablecf** flag is set:

1. The call MUST be forwarded to the destination defined in the **forwardto** list.
2. A 181 response MUST be sent back to the caller indicating that the call is being forwarded.
3. The call forwarding timer MUST be started.

If the **Ms-Sensitivity** header field value does not contain **no-diversion** and the **enablecf** flag is not set and voice mail is configured for the callee:

1. The call MUST be forwarded to voice mail by setting the Request URI field to the user's **voice-mail-gruu** as defined in section 2.2.3, and

2. A 181 response MUST be sent back to the caller indicating that the call is being forwarded.

3.9.6.2 Call Forwarding Timer Expiry

When the **call forwarding** timer expires, the call MUST be forwarded to the user's voice mail if voice mail is configured for the user by setting the Request URI field to the user's voice-mail-gruu as defined in section 2.2.3.

3.9.6.3 Primary User Timer Expiry

This section follows the product behavior described in endnote <65>.

When the primary user timer expires and the **team_ring** flag is set, the call MUST be routed to the targets specified in the team list and the secondary target timer MUST be started. Existing transactions MUST NOT be cancelled.

When the primary user timer expires and the **delegate_ring** flag is set, the call MUST be routed to the targets specified in the **delegates** list and the secondary target timer MUST be started.[<66>] Existing transactions MUST NOT be cancelled.

3.9.6.4 Secondary Target Timer Expiry

This section follows the product behavior described in endnote <67>.

When the secondary target timer expires, all existing transactions MUST be cancelled. If the **enablecf** flag is set, the call MUST be routed to the target specified in the **forwardto** list and the call forwarding timer MUST be started. If the **enablecf** flag is not set, the call MUST be forwarded to the user's voice mail, if one is configured by setting the Request URI field to the user's voice-mail-gruu as defined in section 2.2.3.

3.9.7 Other Local Events

None.

3.10 Extensions for Federation and Public IM Connectivity

As specified in section 2.2.15, this protocol defines the **ms-edge-proxy-message-trust** header field. The following sections specify the header parameters, their values, and the message processing events for this header field.

3.10.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

3.10.1.1 ms-source-type parameter

The header field can contain the **ms-source-type** parameter. This parameter represents the type of connectivity between the remote user or peer server and the enterprise SIP network:

- A parameter value of **AuthorizedServer** can be used to indicate that the peer server is authorized to represent a public IM provider.

- A parameter value of **AutoFederation** can be used to indicate that the **From** user's SIP domain is authorized for federation and resolves through a DNS SRV record to a peer server FQDN (1).
- A parameter value of **DirectPartner** can be used to indicate that the **From** user's SIP domain and the peer server is authorized for direct federation.
- A parameter value of **EdgeProxyGenerated** can be used to indicate the SIP message was generated by a server that is responsible for processing messages from SIP elements outside of the enterprise network.
- A parameter value of **InternetUser** can be used to indicate that the SIP message is received from a remote user.

3.10.1.2 ms-ep-fqdn parameter

The header field can contain the **ms-ep-fqdn** parameter. The parameter value can be used to represent the FQDN of the server that adds the header field.

3.10.1.3 ms-source-verified-user parameter

The header field can contain the **ms-source-verified-user** parameter. If the **ms-source-type** parameter value is equal to "InternetUser", the value of the **ms-source-verified-user** parameter MUST be set to "verified" because **From** user's identity is always verified for messages received from remote users.

If the **ms-source-verified-user** parameter is added:

- A parameter value of "verified" can be used to indicate that the **federated partner** or **public IM provider** is trusted to verify the **From** user's identity and that the federated partner or public IM provider has verified the **From** user's identity.
- A parameter value of "unverified" can be used to indicate that either the federated partner or public IM provider is not trusted to verify the **From** user's identity or that the federated partner or public IM provider has not been able to verify the **From** user's identity.

3.10.1.4 ms-source-network parameter

If the protocol client needs to be informed that the message is from a federated partner or a public IM provider, the header field MUST contain the **ms-source-network** parameter. This parameter MUST NOT be added if the **ms-source-type** parameter exists and its value is equal to "InternetUser". If the **ms-source-network** parameter is added, one of the following two items applies:

- A parameter value of "federation" MUST be used to indicate that the SIP message is from a federated user.
- A parameter value of "publiccloud" MUST be used to indicate that the SIP message is from a public IM user.

If the header field does not contain the **ms-source-network** parameter, this means that the SIP message is from a user that belongs to the same enterprise.

3.10.1.5 ms-remote-fqdn parameter

If the protocol client needs to be informed that the message is from a public IM provider, the header field MAY contain the **ms-remote-fqdn** parameter.<68>

3.10.2 Timers

None.

3.10.3 Initialization

None.

3.10.4 Higher-Layer Triggered Events

None.

3.10.5 Message Processing Events and Sequencing Rules

Except as specified in the following section, the rules for message processing are as specified in [RFC3261].

3.10.5.1 Server Behavior

If the server forwards any message, either a request or a response, to the client that was originally received from a SIP element located outside of the enterprise network, it SHOULD insert an **ms-edge-proxy-message-trust** header field into the message. This header field provides information about source of the SIP element as determined by the server that is responsible for processing messages from SIP elements outside of the enterprise network. The syntax of the **ms-edge-proxy-message-trust** header field is described in section 2.2.15.

3.10.5.2 Client Behavior

The following section specifies protocol client behavior based on parameter values contained in the **ms-edge-proxy-message-trust** header field, as follows:

- If it is identified through the SIP **NOTIFY** message that the user is a federated user or a public IM user, an indication to this effect for this user can be displayed in the contact list.
- If one or more parties in a conversation are users that do not belong to the same enterprise, an indication to this effect can be displayed in the conversation window.
- If it is identified through the SIP NOTIFY message that the user is a public IM user, an indication showing the name and a specific icon identifying the public IM network can be displayed in the contact list for this user.

3.10.6 Timer Events

None.

3.10.7 Other Local Events

None.

3.11 Extensions for Remote Users

As specified in section 2.2.16, this protocol defines the **ms-user-logon-data** header field. The following sections specify the header parameters, their values, and the message processing events for this header field.

3.11.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

If this header field is present, the header field value MUST be "RemoteUser".

3.11.2 Timers

None.

3.11.3 Initialization

None.

3.11.4 Higher-Layer Triggered Events

None.

3.11.5 Message Processing Events and Sequencing Rules

Except as specified in the following section, the rules for message processing are as specified in [RFC3261].

3.11.5.1 Server Behavior

When a server forwards any message, either a request or a response, to the client that connects to it from the outside of the enterprise network, it SHOULD insert an **ms-user-logon-data** header field into the message with a value of "RemoteUser".

3.11.5.2 Client Behavior

The following section specifies protocol client behavior based on the **ms-user-logon-data** header field.

If this header field is present in the reply to a REGISTER request and has a value of "RemoteUser", the protocol client SHOULD treat the requester as an external protocol client connecting from outside of the enterprise network. Under this condition, the protocol client SHOULD do the following:

- Use a **Web service Uniform Resource Locator (URL)** that is accessible from the public Internet for distribution list expansion, address book download, and calendar services.
- Assume that it does not have direct media connectivity to the enterprise network.

3.11.6 Timer Events

None.

3.11.7 Other Local Events

None.

3.12 Extensions for Logging and Monitoring

This section follows the product behavior described in endnote <69>.

As specified in section 2.2.12, this protocol defines the **ms-correlation-id** header field. The following sections specify the header parameters, their values, and the message processing events for this header field.

3.12.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

If an **ms-correlation-id** header field is present, it MUST contain a UUID, as defined in [RFC4122] Section 3. If the same value of the **ms-correlation-id** header field is included in messages for multiple SIP dialogs, those dialogs are considered to be correlated. No specific semantics are defined for which dialogs can be considered correlated; the correlation identifier is intended solely as a hint which log analysis and diagnostic tools can use to infer a relationship between two otherwise-unrelated dialogs.

For example, consider Client B that acts as a back-to-back user agent. This client receives an INVITE from Client A, and sends another INVITE to the final recipient of the message, Client C. Client B generates a new random correlation identifier, and includes the ID in the INVITE to Client C and the response to Client A. Once Client C responds, two otherwise-unrelated dialogs, D1 and D2, have been established. Server processing for both dialogs is unaffected by the additional header, but a server captures and stores the correlation identifier in a log. A log analysis or diagnostic tool later run on the log uses the correlation identifier to identify that dialogs D1 and D2 are related, and hence that Client A and Client C were in communication via the intermediary back-to-back user agent.

If the header is absent, or the value of the header is not used by any other dialog, the dialog is not correlated.

3.12.2 Timers

None.

3.12.3 Initialization

None.

3.12.4 Higher-Layer Triggered Events

3.12.4.1 Client Behavior

If the SIP endpoint creates two dialogs that are related to each other, it SHOULD generate a UUID using a procedure compatible with [RFC4122] Section 4, and add an **Ms-Correlation-Id** header field with this value to the INVITE or REFER messages that created the dialogs.

3.12.5 Message Processing Events and Sequencing Rules

Except as specified in the following section, the rules for message processing are as specified in [RFC3261].

3.12.5.1 Client Behavior

If the SIP endpoint receives an INVITE or REFER containing an **Ms-Correlation-Id** header field, and in response it wishes to create another dialog that is related to the dialog created by that request, it SHOULD add an **Ms-Correlation-Id** header field with the same value it received to the INVITE or REFER message it uses to create the second dialog.

If the SIP endpoint receives an INVITE or REFER without an **Ms-Correlation-Id** header field, and in response it wishes to create another dialog that is related to the dialog created by that request, it SHOULD generate a UUID using a procedure compatible with [RFC4122] Section 4 and add an **Ms-Correlation-Id** header field with this value both to its final response to the message received, and to the INVITE or REFER request it uses to create the second dialog.

3.12.5.2 Proxy Behavior

When a SIP proxy that logs dialog creation events processes a dialog creating request or final response to a dialog creating request that has an **Ms-Correlation-Id** header field present and the value in this field is a valid UUID, as defined in [RFC4122] section 3, it can record the value in the log. If the value is not a valid UUID, the proxy SHOULD ignore the presence of the header.

3.12.6 Timer Events

None.

3.12.7 Other Local Events

None.

3.13 Extensions for Call Context

This section follows the product behavior described in endnote <70>.

This protocol specifies the call context mechanism for protocol client and server endpoints to create notes related to a given call that can be sent to another party receiving the INVITE that creates a new call. There are a number of pieces of information contained within the call context content that helps the endpoint to correlate and render the call context data and notes to the user. The call context data is carried within the related INVITE request as a MIME type in the message body of the request.

3.13.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

User agents creating notes in relation to a call can convey those text notes using the call context data type.

3.13.2 Timers

None.

3.13.3 Initialization

None.

3.13.4 Higher-Layer Triggered Events

None.

3.13.5 Message Processing Events and Sequencing Rules

Except as specified in the following section, the rules for message processing are as specified in [RFC3261].

3.13.5.1 Client Behavior

The following section specifies client behavior based on the **application/ms-conversation-context+xml** content type. The following apply:

- Can only use the SIP INVITE request to convey call context data.
- Can only include a single call context MIME body in the request.
- MUST set the content type to **application/ms-conversation-context+xml** for the MIME body conveying call context data.
- The **id** element for each call context body MUST be unique among all call context data created by the server, and MUST appear only once in the call context data.
- The **from** element MUST be present in the call context data and appear only once.
- The **uri** child element MUST be present within the **from** element.
- The **displayName**, **onBehalfUri**, and **onBehalfDisplayName** child elements can appear in the **from** element and SHOULD be present if the data is available at the server for that call, but MUST NOT appear more than once each.
- The **to** element MUST be present in the call context data and appear only once.
- The **uri** child element MUST be present within the **to** element.
- The **displayName**, **onBehalfUri**, and **onBehalfDisplayName** child elements can appear in the **to** element and SHOULD be present if the data is available at the server for that call, but MUST NOT appear more than once each.
- The **participants** element MUST be present in the call context data and appear only once and MUST contain one or more **participant** elements.
- A **participant** element MUST be present for the author of the call context data.
- Other **participant** elements can be present for each party involved with the call.
- The **uri** child element MUST be present within the **participant** element.
- The **displayName**, **onBehalfUri**, and **onBehalfDisplayName** child elements can appear in the **participant** element, SHOULD be present if the data is available at the server for that call, but MUST NOT appear more than once each.
- The **date** element MUST be in UTC format, MUST be present in the call context data and MUST appear only once.
- The **conversationId** element MUST be present in the call context data, MUST appear only once, and MUST be unique among all call context data created by the server.

- The **dataFormat** element MUST be present in the call context data, MUST appear only once, and MUST have a value of "text/plain".
- The **contextData** element MUST be present in the call context data.
- The **mode** element can be present one or more times in the call context data, each time with a unique value, and SHOULD consist of one of the following values:
  - audio
  - video
  - im
  - applicationSharing

3.13.5.2 Server Behavior

The following section specifies protocol server behavior based on the **application/ms-conversation-context+xml** content type. The following apply:

- Can ignore call context data that does not comply with the **application/ms-conversation-context+xml** XSD or is conveyed through other SIP messages other than the INVITE request to initiate a new dialog.
- Can ignore call context data with a **dataFormat** element value other than "text/plain".
- Can ignore call context data with a **mode** element that has a value other than one of the following:
  - audio
  - video
  - im
  - applicationSharing

3.13.6 Timer Events

None.

3.13.7 Other Local Events

None.

3.14 Safe Call Transfer Extension

This section follows the product behavior described in endnote <71>.

The safe call transfer extension tailors the routing behavior while transferring calls using the REFER request. Using this extension, a user agent transferring calls can request that the transferee disable call forwarding and voice mail for the triggered INVITE request.

3.14.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

3.14.2 Timers

None.

3.14.3 Initialization

None.

3.14.4 Higher-Layer Triggered Events

If the user agent supports the safe call transfer procedure described in this section, it MUST advertise this by placing the **ms-safe-transfer** option tag in the **Supported** header of both the INVITE request and the **200 OK** response to the INVITE request.

3.14.5 Message Processing Events and Sequencing Rules

When the user agent receives a REFER request in the INVITE dialog in which it previously advertised support for safe call transfer, as described in section 3.14.4, the user agent MUST examine the **Refer-To** header field of the REFER request. If the **Ms-Sensitivity** header field is present in the **headers** parameter of the URI in the **Refer-To** header field, the user agent MUST extract the **Ms-Sensitivity** header field and its value and add it to the INVITE request that it generates as the result of processing the REFER request.

3.14.6 Timer Events

None.

3.14.7 Other Local Events

None.

3.15 Extensions for ICE SDP Interworking and Multipart MIME Support

This section follows the product behavior described in endnote <72>.

User agents use multi-part MIME to convey multiple SDP parts and call context data in an INVITE request during session initialization. This document describes a method of using multi-part MIME to enable interoperability with SIP elements for which it cannot be determined in advance whether they support [IETFDRAFT-ICENAT-06] or [IETFDRAFT-ICENAT-19] or both.

3.15.1 Abstract Data Model

None.

3.15.2 Timers

None.

3.15.3 Initialization

None.

3.15.4 Higher-Layer Triggered Events

3.15.4.1 Outgoing INVITE

This section follows the product behavior described in endnote <73>.

When a user agent initiates a SIP dialog using an INVITE containing SDP, as defined in [MS-SDPEXT], it MUST use one of the following MIME structures to construct the INVITE request body.

```
3-level deep multipart
   L1: Multipart/mixed
      L2:Multipart/alternative
         L3:SDP ICEv6  (with ms-proxy-2007fallback parameter)
         L3:SDP ICEv19
      L2:Call context
If there is no call context, the following structure is used.
2-level deep multipart
   L1:Multipart/alternative
      L2:SDP ICEv6  (with ms-proxy-2007fallback parameter)
      L2:SDP ICEv19
```

SDP ICEv6 and SDP ICEv19 are specified in [IETFDRAFT-ICENAT-06] and [IETFDRAFT-ICENAT-19] respectively. Call context is described in this section.

L1 refers to the first level in the SIP message body, L2 refers to the second level, and L3 refers to the third level.

The **ms-proxy-2007fallback** parameter in the **Content-Disposition** header field is used as a hint to the proxy server to retry the INVITE with only a single body part when a 415 response is received indicating that the remote user agent does not accept multi-part. The syntax of the **ms-proxy-2007fallback** parameter is described in section 2.2.14, and the applicable proxy server processing of the 415 response is described in section 3.9.5.5.

For 2-level deep multi-part, the SDP MUST be ICEv6, ICEv19 or it does not contain any **Interactive Connectivity Establishment (ICE)**.

If ICEv19 SDP is carried in the multi-part MIME, it MUST be placed in the last part of the multi-part MIME that is carrying all the SDPs.

The 3-level deep multi-part must follow the same rules for carrying SDPs as in the 2-level deep multi-part. The only difference being that the SDPs are level 3(L3) instead of level 2(L2).

3.15.5 Message Processing Events and Sequencing Rules

3.15.5.1 Processing INVITE

When an incoming INVITE is received that contains multi-part MIME structures described in section 3.15.4.1, the user agent MUST pick SDP ICEv19 as the offer if the UAS supports [IETFDRAFT-ICENAT-19], as specified in [MS-SDPEXT].

Alternatively, if the UAS does not support [IETFDRAFT-ICENAT-19], as specified in [MS-SDPEXT], but supports [IETFDRAFT-ICENAT-06], as specified in [MS-SDPEXT], the user agent MUST pick SDP ICEv6 as the offer.<74>

If the incoming INVITE does not contain any Interactive Connectivity Establishment(ICE), it will have only one SDP as specified in [RFC3261]

3.15.5.2 Processing 415Response

When an INVITE with the body described in section 3.15.4.1 is rejected with a 415 response, the user agent SHOULD retry the INVITE without multi-part MIME. The body SHOULD contain only SDP ICEv6 without the **ms-proxy-2007fallback** parameter in the **Content-Disposition** header field.

3.15.6 Timer Events

None.

3.15.7 Other Local Events

None.

3.16 Extensions for Agent Anonymity

As specified in section 2.2.21 and section 2.2.22, this protocol defines the **Ms-Call-Info** and **P-Agent-On-Behalf-Of** header fields. The following sections specify the headers and the message processing events for these header fields when anonymization is performed.<75>

3.16.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

3.16.1.1 Ms-Call-Info Header

The **Ms-Call-Info** header conveys information about calls. The server endpoint SHOULD set the value of the **Ms-Call-Info** header to "rgs.anonymization". Client endpoints SHOULD ignore any other value.

A server endpoint SHOULD add the **Ms-Call-Info** header to outgoing SIP INVITE and SIP responses to communicate the fact that the call is anonymized. The server endpoint SHOULD provide anonymity. For example, this can be achieved by using a signaling back-to-back agent.

3.16.1.2 P-Agent-On-Behalf-Of Header

When a client endpoint makes a call on behalf of an identity, it MUST use the **P-Agent-On-Behalf-Of** header.

The server endpoint SHOULD validate that the user has the permission to make on-behalf-of requests.

3.16.2 Timers

None.

3.16.3 Initialization

None.

3.16.4 Higher-Layer Triggered Events

None.

3.16.5 Message Processing Events and Sequencing Rules

3.16.5.1 Server Behavior

The server endpoint SHOULD send an **Ms-Call-Info** header set to "rgs.anonymization" if it provides anonymity, such as through a back-to-back agent.

Responses to new dialogs established by a user endpoint SHOULD contain an **Ms-Call-Info** header set to "rgs.anonymization" if the server endpoint provides anonymity, such as through a back-to-back agent.

If the server endpoint receives an INVITE with a **P-Agent-On-Behalf-Of** header, it SHOULD validate that the requestor, which is identified by the **P-Asserted-Identity** header, as specified in [RFC3325] section 9.1, has permission to make on-behalf-of requests. If the **P-Asserted-Identity** header is not present or the requestor does not have the required permission, the request SHOULD be declined with a 403 response.

If the request is valid, the server endpoint SHOULD proceed with the establishment of the call and, if the call is made anonymously, SHOULD add an **Ms-Call-Info** header set to "rgs.anonymization" in its response to the client endpoint.

3.16.6 Timer Events

None.

3.16.7 Other Local Events

None.

3.17 E911 Message Processing

This section describes the processing of the E911 INVITE<76>, as defined in section 2.2.23.

3.17.1 Abstract Data Model

None.

3.17.2 Timers

None.

3.17.3 Initialization

None.

3.17.4 Higher-Layer Triggered Events

None.

3.17.5 Message Processing Events and Sequencing Rules

Except as specified in the following section, the rules for message processing are as specified in [RFC3261].

3.17.5.1 Client Behavior

The client retrieves the **locationPolicy** in-band provisioning group, as specified in [MS-SIPREGE] section 2.2.2.5.7. The location policy indicates whether Enhanced Emergency Services are enabled for the endpoint and if enabled, the location policy specifies the **EmergencyDialString**, **EmergencyDialMask**, **NotificationUri**, **ConferenceUri**, **ConferenceMode**, and **LocationPolicyTagID** for the endpoint. The client obtains its location by either making a request to the location information service, as specified in [MS-E911WS], or by capturing the location based on user input. The client composes the INVITE specified in 2.2.23. The client publishes a time-bound routing category instance of the preamble containing the **e911active** flag, as specified in section 3.9.5.1.2, to disable all call forwarding rules, as specified in [MS-SIPREGE]. The client sends the previously composed E911 INVITE to the server.

3.17.5.2 Server Behavior

The server identifies an emergency call when it detects a **Priority** header with value "emergency" in the INVITE. The server retrieves the location policy based on the **LocationPolicyTagID** sent within the **Presence Information Data Format Location Object (PIDF-LO)** embedded as a MIME part inside the message body of the INVITE. The **PIDF-LO** format is specified in [RFC4119].The server ignores the **geolocation** header and picks the last MIME part that has a **PIDF-LO** embedded in it. The **geolocation** header is defined in [RFC6442]. Upon receiving the emergency call, in addition to routing the call to E911 Service providers or **public switched telephone network (PSTN)**, the server MUST send an IM message on behalf of the client endpoint making the E911 call to each target in the **NotificationUri** specified in the location policy. The IM INVITE request MUST be constructed as follows:

1. The request MUST contain a **Priority** header with the value "emergency".
2. The request MUST contain a **Call-Info** header with the SIP URI of the user making the emergency call. The **Call-Info** header MUST have a **purpose** parameter with the value "ms-emergency-notification". The ABNF, as defined in [RFC5234], for the **Call-Info** header is defined in [RFC3261], section 25.1.
3. The body of the message MUST be plain text containing all the descendants of the **civicAddress** and **method** elements in the **PIDF-LO** as name-value pairs. The **civicAddress** and **method** element schema are defined in [RFC4119].

The server MUST continue to route the emergency call regardless of any errors encountered while generating or routing the IM message.

3.17.6 Timer Events

None.

3.17.7 Other Local Events

None.

4 Protocol Examples

4.1 EPID Mechanism

The following REGISTER request demonstrates use of the **epid** parameter in the **From** header field.

```
REGISTER sip:contoso.com SIP/2.0
From: <sip:alice@contoso.com>;tag=33975904fc;epid=01010101
To: <sip:alice@contoso.com>
Call-ID: 21c7d6e384c249afac26e3f3016140a6
CSeq: 88 REGISTER
```

Note that other SIP headers in the SIP request are not included.

4.2 SIP.INSTANCE Mechanism

This example first shows the generation of the **+sip.instance** parameter value for a user agent that uses both **epid** and **+sip.instance** parameters to identify its endpoint, as described in section 3.3.3.1.

Given an **epid** parameter value of 01010101, it is first converted to a canonical sequence of octets:

```
0x30 0x31 0x30 0x31 0x30 0x31 0x30 0x31
```

Next, the hash of the name-space identifier concatenated with the canonical representation of the **epid** value is computed:

```
sha1 (0x03 0xfb 0xac 0xfc 0x73 0x8a 0xef 0x46 0x91 0xb1 0xe5 0xeb 0xee 0xab 0xa4 0xfe
0x30 0x31 0x30 0x31 0x30 0x31 0x30 0x31) = 0xA8 0x82 0x16 0x4B 0x68 0xF9 0x01 0xE7 0x03
0xFC 0x7C 0x67 0x41 0xDC 0x66 0x97 0xB8 0xA1 0xA9 0x3E
```

Finally, the previous hash is used to obtain the following UUID:

```
4b1682a8-f968-5701-83fc-7c6741dc6697
```

The following REGISTER request demonstrates the use of the **+sip.instance** parameter in the **Contact** header field and the **epid** parameter in the **From** header field.

```
REGISTER sip:contoso.com SIP/2.0
From: <sip:alice@contoso.com>;tag=33975904fc;epid=01010101
To: <sip:alice@contoso.com>
Call-ID: 21c7d6e384c249afac26e3f3016140a6
CSeq: 88 REGISTER
Contact: <sip:192.0.2.1:27221; transport=tls; ms-
opaque=29c344caf9>; methods="INVITE, MESSAGE, INFO, OPTIONS, BYE, CANCEL, NOTIFY, ACK, RE
FER, BENOTIFY"; proxy=replace; +sip.instance="<urn:uuid:4b1682a8-f968-5701-83fc-
7c6741dc6697>"
```

Note that other SIP headers in the SIP request are not included.

4.3 GRUU Mechanism

The following examples demonstrate various GRUU syntaxes:

A **GRUU** for the user agent that follows the registration procedure defined in [MS-SIPREGE] is as follows:

```
sip:alice@contoso.com;gruu;opaque=user:epid:qIIWS2j5AVeD_HxnQdxmlwAA
```

A **GRUU** for an application that implements the voice mail service for the user is as follows:

```
sip:alice@contoso.com;gruu;opaque=app:voicemail
```

GRUUs for multimedia conference endpoints are as follows:

```
sip:alice@contoso.com;gruu;opaque=app:conf:focus:id:36022956C3FC3243B8121CD611363ED0
sip:alice@contoso.com;gruu;opaque=app:conf:chat:id:36022956C3FC3243B8121CD611363ED0
sip:alice@contoso.com;gruu;opaque=app:conf:audiovideo:id:36022956C3FC3243B8121CD611363ED0
```

GRUUs for servers are as follows:

```
sip:homeserver.contoso.com@contoso.com;gruu;opaque=srvr:HomeServer:dL8cwxBrTuG8eC4-
Q_GNGAAA
sip:mediationserver.contoso.com@contoso.com;gruu;opaque=srvr:MediationServer:_tRfGncbQyun
3v75Q1qr9QAA
sip:mrasserver.contoso.com@contoso.com;gruu;opaque=srvr:MRAS:OKPDbAVxIEKtPh2g624vPAAA
sip:qosmserver.contoso.com@contoso.com;gruu;opaque=srvr:QoSM:WftfTuTVQCSAB0ZJi-j7qAAA
```

4.4 Firewall and Network Address Translation Traversal Aid Extensions

The following example demonstrates how the original REGISTER request was modified by the SIP proxy to preserve transport layer information necessary for NAT traversal.

The original REGISTER request is as follows:

```
REGISTER sip:contoso.com SIP/2.0
From: <sip:alice@contoso.com>;tag=33975904fc;epid=01010101
To: <sip:alice@contoso.com>
Call-ID: 21c7d6e384c249afac26e3f3016140a6
CSeq: 88 REGISTER
Via: SIP/2.0/TLS 192.0.2.1:27221
Contact: <sip:192.0.2.1:27221; transport=tls; ms-
opaque=29c344caf9>; methods="INVITE, MESSAGE, INFO, OPTIONS, BYE, CANCEL, NOTIFY, ACK, RE
FER, BENOTIFY"; proxy=replace; +sip.instance="<urn:uuid:4b1682a8-f968-5701-83fc-
7c6741dc6697>"
```

The REGISTER request after proxy processing is as follows:

```
REGISTER sip:contoso.com SIP/2.0
From: <sip:alice@contoso.com>;tag=33975904fc;epid=01010101
To: <sip:alice@contoso.com>
Call-ID: 21c7d6e384c249afac26e3f3016140a6
CSeq: 88 REGISTER
Via: SIP/2.0/TLS 192.0.2.1:27221; received=192.168.0.2; ms-received-port=1201; ms-
received-cid=3540900
Contact: <sip:192.168.0.2:1201; transport=tls; ms-opaque=29c344caf9; ms-received-
cid=3540900>; methods="INVITE, MESSAGE, INFO, OPTIONS, BYE, CANCEL, NOTIFY, ACK, REFER, B
ENOTIFY"; +sip.instance="<urn:uuid:4b1682a8-f968-5701-83fc-7c6741dc6697>"
```

4.5 Reliable and Consistent Message Routing Within Redundant Server Network

The following example demonstrates SIP proxies placing various pieces of information into the **Record-Route** header fields of the dialog creating a 200 OK response message to a SUBSCRIBE request.

```
SIP/2.0 200 OK
FROM: <sip:alice@contoso.com>;tag=2187d9f392;epid=01010101
TO: <sip:bob@contoso.com>;tag=3l3qz7tx
CSEQ: 3 SUBSCRIBE
CALL-ID: f0ec9c595c1f412ca6b71318beb599bb
RECORDROUTE: <sip:server1.contoso.com:5061;transport=tls;lr;ms-key-
info=mACAAODZIzT_XXbu1V_IAQECAAADZgAAAKQAANMFUpbsXZoVmYcoLP8PT9anIkOw7BnvcFRRkZewoiMYj3B6
1YacQGTK4TMsKnJXCM86liVZHosw8jUyFf2OXMyOLLv3ZVw477ajvdErKm0E5OQybBg8o6e3g1wK9rua4xUHwyZ1T
6_CkS6TQvpebxXJG5Y8dA40VIzMIllIjAHfRSo9XMZW1lyJnpHoa53vuD1BV1QccxH9ht5dw3sKqKAgsyBT4Bmm3a
bFJ6nKhZpNlybt6EkVqBD7ArG5dyNPrUlcT8VLOPINVSGwvviWBygEVRfIGauMqIbMooXLq6PMYUAg6TIYfEIdugq
RnIYgu_hnihBK6wKjV2w;ms-route-sig=ga3IN7MltlsglDvxIE_bYt5lVbZ3E>
RECORDROUTE: <sip:server2.contoso.com:5061;transport=tls;ms-role-rs-from;lr;ms-route-
sig=ec1Fe_32fglb4iILWFJb5iKqeNeps7y6vY9zXAAA>
CONTACT: <sip:alice@contoso.com;gruu;opaque=user:epid:qIIWS2j5AVeD_HxnQdxmlwAA>
```

4.6 Dialog State Recovery

This section follows the product behavior described in endnote <77>.

The following example shows messages exchanged between the user agent and the proxy server when the proxy detects dialog state loss and communicates this to the user agent, which subsequently recovers the dialog.

The user agent sends a mid-dialog request with the route set from the current dialog state.

```
MESSAGE sip:Alice@contoso.com;gruu;opaque=user:epid:qIIWS2j5AVeD_HxnQdxmlwAA SIP/2.0
Route: <sip:server.contoso.com:5061;transport=tls;opaque=state:F:T:Ci.D1100:Ti.dyHFp3e3J0
mXFhCDvmsQ7QAA;lr;ms-route-sig=aag0AbAT3mK4Ga8lsHSyTeZnAETjcRJpFx8YnUbQAA>
From: sip:Bob@contoso.com;epid=02020202;tag=02020202
To: sip:Alice@contoso.com;epid=01010101;tag=01010101
Call-Id: f0ec9c595c1f412ca6b71318beb599bb
Via: SIP/2.0/TLS 192.0.2.1:27221;branch=z9hG4bK94bd
Cseq: 3 MESSAGE
Supported: Ms-Dialog-Route-Set-Update
Content-Length: 27

Alice, are you still there?
```

The proxy detects that the references to the state information stored in the route set are not valid and that the user agent supports the dialog state recovery procedure as indicated by the **Ms-Dialog-Route-Set-Update** option tag in the **Supported** header field. The proxy responds with a 430 Flow Failed response, requesting the user agent to update the dialog state information.

```
SIP/2.0 430 Flow Failed
From: sip:Bob@contoso.com;epid=02020202;tag=02020202
To: sip:Alice@contoso.com;epid=01010101;tag=01010101
Call-Id: f0ec9c595c1f412ca6b71318beb599bb
Via: SIP/2.0/TLS 192.0.2.1:27221;branch=z9hG4bK94bd;ms-received-cid=3540900
Cseq: 3 MESSAGE
P-Dialog-Recovery-Action: dialog-route-set-update
Content-Length: 0
```

The user agent sends the correct target refresh request without the route set to recover the dialog state.

```
INVITE sip:Alice@contoso.com;gruu;opaque=user:epid:qIIWS2j5AVeD_HxnQdxmlwAA SIP/2.0
From: sip:Bob@contoso.com;epid=02020202;tag=02020202
To: sip:Alice@contoso.com;epid=01010101;tag=01010101
Call-Id: f0ec9c595c1f412ca6b71318beb599bb
Via: SIP/2.0/TLS 192.0.2.1:27221;branch=z9hG4bKa8d4
Cseq: 4 INVITE
Supported: Ms-Dialog-Route-Set-Update
Contact: <sip:Bob@contoso.com;gruu;opaque=user:epid:uVUjrngkI1wHVm3r2esBAAA>
Content-Length: 0
```

The user agent receives the 200 OK response and updates its dialog state with the new route set.

```
SIP/2.0 200 OK
RecordRoute: <sip:server.contoso:5061;transport=tls;opaque=state:F:T:Ci.D1200:Ti.dyHFp3e3
J0mXFhCDvmsQ7QAA;lr;ms-route-sig=aalzpOt84oODZx4KmWgmgJLf_WGfEsKwh8YnUbQAA>
From: sip:Bob@contoso.com;epid=02020202;tag=02020202
To: sip:Alice@contoso.com;epid=01010101;tag=01010101
Call-Id: f0ec9c595c1f412ca6b71318beb599bb
Via: SIP/2.0/TLS 192.0.2.1:27221;branch=z9hG4bKa8d4;ms-received-cid=3540900
Cseq: 4 INVITE
Contact: <sip:Alice@contoso.com;gruu;opaque=user:epid:qIIWS2j5AVeD_HxnQdxmlwAA>
Content-Length: 0
```

The user agent then resends the request with the updated route set.

```
MESSAGE sip:Alice@contoso.com;gruu;opaque=user:epid:qIIWS2j5AVeD_HxnQdxmlwAA SIP/2.0
Route: <sip:server.contoso:5061;transport=tls;opaque=state:F:T:Ci.D1200:Ti.dyHFp3e3J0mXFh
CDvmsQ7QAA;lr;ms-route-sig=aalzpOt84oODZx4KmWgmgJLf_WGfEsKwh8YnUbQAA>
From: sip:Bob@contoso.com;epid=02020202;tag=02020202
To: sip:Alice@contoso.com;epid=01010101;tag=01010101
Call-Id: f0ec9c595c1f412ca6b71318beb599bb
Via: SIP/2.0/TLS 192.0.2.1:27221;branch=z9hG4bK97b2
Cseq: 5 MESSAGE
Supported: Ms-Dialog-Route-Set-Update
Content-Length: 27

Alice, are you still there?
```

The request gets through and the user agent receives a successful response.

```
SIP/2.0 200 OK
From: sip:Bob@contoso.com;epid=02020202;tag=02020202
To: sip:Alice@contoso.com;epid=01010101;tag=01010101
Call-Id: f0ec9c595c1f412ca6b71318beb599bb
Via: SIP/2.0/TLS 192.0.2.1:27221;branch=z9hG4bK97b2;ms-received-cid=3540900
Cseq: 5 MESSAGE
Content-Length: 0
```

4.7 Routing Preamble

4.7.1 Blocking Preamble

The following is an example of a preamble that blocks the call.

```
<?xml version="1.0" encoding="utf-8"?>
<routing xmlns="http://schemas.microsoft.com/02/2006/sip/routing"
        name="rtcdefault" version="1" >
  <preamble>
```

```
    <flags name="clientflags" value="block"/>
  </preamble>
</routing>
```

In the previous example, because the **clientflags** contains "block", the call is blocked.

4.7.2 Simultaneous Ring

```
<?xml version="1.0" encoding="utf-8"?>
<routing xmlns="http://schemas.microsoft.com/02/2006/sip/routing"
        name="rtcdefault" version="1" >
  <preamble >
    <list name="forwardto">
      <target uri="sip:+14255550199@contoso.com;user=phone"/>
    </list>
    <list name="simultaneous_ring" >
      <target uri="sip:+14255550100@contoso.com;user=phone"/>
    </list>
    <flags name="clientflags" value="work_hours simultaneous_ring enablecf"/>
    <wait name="total" seconds="18"/>
  </preamble>
</routing>
```

In the previous example, the call is forked to all the registered endpoints of the user and, because the **simultaneous_ring** flag is set, the call is also forked to the simultaneous ring device "sip:+14255550100@contoso.com;user=phone". If no success response is received within 18 seconds, which is the wait time specified in the **wait** element named **total**, all forks are cancelled. Because the **enablecf** flag is set, the call is then forked to the forwarding destination indicated in the **forwardto** list, which is "sip:+14255550199@contoso.com;user=phone".

If the **simultaneous_ring** target SIP URI's **userinfo** part carries a parameter **ms-skip-rnl-param** ="ms-skip-rnl=" ("true" / "false") as shown in the below example, no reverse number lookup will be performed on the number if the value is "true". Default behavior without this parameter would be to perform reverse number lookup.

```
<?xml version="1.0" encoding="utf-8"?>
<routing xmlns="http://schemas.microsoft.com/02/2006/sip/routing"
        name="rtcdefault" version="2" minSupportedClientVersion="4.0.0.0">
  <preamble >
    <list name="simultaneous_ring" >
      <target uri="sip:+14255550100;ms-skip-rnl=true@contoso.com;user=phone"/>
    </list>
    <flags name="clientflags" value="simultaneous_ring"/>
    <wait name="total" seconds="20"/>
  </preamble>
</routing>
```

4.7.3 Call Forward

```
<?xml version="1.0" encoding="utf-8"?>
<routing xmlns="http://schemas.microsoft.com/02/2006/sip/routing"
        name="rtcdefault" version="1" >
  <preamble >
    <list name="forwardto">
      <target uri="sip:+14255550199@contoso.com;user=phone"/>
    </list>
    <list name="simultaneous_ring" >
      <target uri="sip:+14255550100@contoso.com;user=phone"/>
    </list>
```

```
    <flags name="clientflags" value="work_hours forward_immediate simultaneous_ring
enablecf"/>
    <wait name="total" seconds="18"/>
  </preamble>
</routing>
```

In the previous example, the **forward_immediate** flag indicates that the call is forwarded immediately. Because the **enablecf** flag is also present, the call is forwarded to the destination in the **forwardto** list. If the **enablecf** flag is not present, the call is forwarded to the user's voice mail. In either case, the registered endpoints and the simultaneous ring device are not rung.

If the **forwardto** target SIP URI's **userinfo** part carries a parameter **ms-skip-rnl-param**="ms-skip-rnl=" ("true" / "false") as shown in the below example, no reverse number lookup will be performed on the number if the value is "true". Default behavior without this parameter would be to perform reverse number lookup.

```
<?xml version="1.0" encoding="utf-8"?>
<routing xmlns="http://schemas.microsoft.com/02/2006/sip/routing"
        name="rtcdefault" version="2" minSupportedClientVersion="4.0.0.0">
  <preamble >
    <list name="forwardto">
      <target uri="sip:+14255550199;ms-skip-rnl=true@contoso.com;user=phone"/>
    </list>
    <flags name="clientflags" value="enablecf forward_immediate"/>
    <wait name="total" seconds="20"/>
  </preamble>
</routing>
```

4.7.4 Team Ring

This section follows the product behavior described in endnote <78>.

```
<?xml version="1.0" encoding="utf-8"?>
<routing xmlns="http://schemas.microsoft.com/02/2006/sip/routing"
        name="rtcdefault" version="2"
minSupportedClientVersion="2.0.0.0" >
  <preamble >
    <list name="team">
      <target uri="sip:Alice@contoso.com "/>
      <target uri="sip:Bob@contoso.com "/>
    </list>
    <flags name="clientflags" value="team_ring"/>
    <wait name="user" seconds="10"/>
    <wait name="team2" seconds="10"/>
  </preamble>
</routing>
```

In this example, the **team_ring** flag indicates that team ringing is enabled. The call is forked to all registered endpoints. If no success response is received within **user** seconds, which is 10 seconds in this example, the call is routed to the targets specified in the **team** list, Alice@contoso.com and Bob@contoso.com. Note that the registered endpoints are not cancelled at this time. If no success response is received within 10 additional seconds, which is the **team2** wait time, all existing forks are cancelled and the call is forwarded to voice mail if the user is enabled for voice mail.

4.8 History-Info

This section follows the product behavior described in endnote <79>.

The following example shows the **History-Info** header field inserted by the proxy in the INVITE request forwarded to the registered endpoint.

```
INVITE sip:192.0.2.1:51152;transport=tls;ms-opaque=bab87d7e6e;ms-received-cid=244100
SIP/2.0
RecordRoute: <sip:server.contoso.com:5061;transport=tls;opaque=state:F:Ci.R2>;ms-
rrsig=djvCtpOBl7EzJlJIPA8FZ2TtCdfcZHZduS3M4K_QAA;tag=C2FBFDDF86D85988E2FE9C475D8B20D0
Via: SIP/2.0/TLS 192.168.0.2:5061;branch=z9hG4bK.A1ABD;branched=TRUE;ms-internal-
info="bvL4ijJzvRAsUh9KHAufCF_yfKiWpHZduSTBXqAAAA"
Via: SIP/2.0/TLS 192.168.0.3:1199;branch=z9hG4bK94bd;msreceivedcid=A552C00
Authentication-Info: NTLM rspauth="01000000ECFE1CAD61AAC15164000000", srand="AC62DEB8",
snum="504", opaque="DC8F829A", qop="auth", targetname="server.contoso.com", realm="SIP
Communications Service"
Max-Forwards: 68
Content-Length: 0
From: <sip:Alice@contoso.com>;epid=01010101
To: <sip:Bob@contoso.com>;epid=02020202C
Seq: 39513
INVITECall-ID: 772937b8-0e12-4639-8c79-9d2ac32f2a56
Contact: <sip:alice@contoso.com;gruu;opaque=user:epid:qIIWS2j5AVeD_HxnQdxmlwAA>
Supported: gruu-10History-Info: <sip:Bob@contoso.com>;index=1
```

4.9 Extension for Federation and Public IM Connectivity

The following examples show the extension header field **ms-edge-proxy-message-trust** used for federation and public IM connectivity. The format for this header field is specified in section 2.2.15.

In this example, the **ms-edge-proxy-message-trust** header field indicates that the SIP NOTIFY message was received from a federated partner:

```
NOTIFY sip:192.0.2.1:18168; transport=tls; ms-opaque=7eacdda82d; ms-received-
cid=7C9B00; grid SIP/2.0
ms-edge-proxy-message-trust: ms-source-type=AutoFederation; ms-ep-
fqdn=edgeserver.contoso.com; ms-source-verified-user=verified; ms-source-
network=federation
```

Note that other SIP headers in the SIP request are not included.

In this example, the **ms-edge-proxy-message-trust** header field indicates that the SIP NOTIFY message was received from a public IM provider:

```
NOTIFY sip:192.0.2.1:18168; transport=tls; ms-opaque=7eacdda82d; ms-received-
cid=7C9B00; grid SIP/2.0
ms-edge-proxy-message-trust: ms-source-type=AuthorizedServer;ms-ep-
fqdn=edgeserver.contoso.com;ms-source-verified-user=verified;ms-source-
network=publiccloud;ms-remote-fqdn=edgeserver.publicnetwork.com
```

Note that other SIP headers in the SIP request are not included.

In this example, the **ms-edge-proxy-message-trust** header field indicates that the SIP response was generated by a server on the enterprise network edge because it could not route the outbound message:

```
SIP/2.0 504 Server time-out
ms-edge-proxy-message-trust: ms-source-type=EdgeProxyGenerated; ms-ep-
fqdn=edgeserver.contoso.com; ms-source-verified-user=verified; ms-source-
network=federation
```

Note that other SIP headers in the SIP response are not included.

4.10 Extension for Remote Users

The following examples show the extension header field **ms-user-logon-data**. The format for this header field is specified in section 2.2.16.

The following example shows a response to a REGISTER request. The **ms-user-logon-data** header field indicates that the user is a remote user.

```
SIP/2.0 200 OK
From: <sip:alice@contoso.com>;tag=1b3884236d;epid=e06accb078
To: <sip:alice@contoso.com>;tag=D4EF81E564DD858A326CC721EF4A8FAF
Call-ID: 5899a88068934f8385a0b0b5e03be045
CSeq: 3 REGISTER
ms-user-logon-data: RemoteUser
Authentication-
Info: NTLM rspauth="0100000000000000046DD35D06323180F", srand="64306136", snum="1", opaque
="0A79BAD2", qop="auth", targetname="ocsserver.contoso.com", realm="SIP Communications Se
rvice"
RecordRoute: <sip:server1.contoso.com:5061;transport=tls;lr;ms-received-cid=3AFDE300>
Contact: <sip:192.0.2.4:2904;transport=tls;ms-opaque=2cd64e3000;ms-received-
cid=1D8AF00>;expires=2905;+sip.instance="<urn:uuid:75ab1008bcc45544924daa177c824291>";gru
u="sip:alice@contoso.com;opaque=user:epid:CBCrdcS8RFWSTaoXfIJCkQAA;gruu"
```

4.11 Extension for Call Context

This section follows the product behavior described in endnote <80>.

The following examples show the extension content type **application/ms-conversation-context+xml**. The format for this content type is specified in section 2.2.20.

The following example shows an INVITE request containing the **application/ms-conversation-context+xml** content type in the message body of the request.

```
INVITE sip:192.0.2.3:59682;transport=tls;ms-opaque=f297889669;ms-received-cid=4EA600
SIP/2.0From: <sip:alice@contoso.com>;epid=42933B3A88;tag=f962b589a8To:
<sip:marco@contoso.com>;epid=7913c4c11dContent-Length: ...
Content-Type: multipart/mixed;boundary=0VUf5fZQGOkBjYIfaZ2yOZCi5OdMrt2A


--0VUf5fZQGOkBjYIfaZ2yOZCi5OdMrt2A
CONTENT-TYPE: multipart/alternative; boundary=4FqyUUSfl7GyNwhB0PABKoF6PTFb6Ovl

--4FqyUUSfl7GyNwhB0PABKoF6PTFb6OvlContent-Type: ...Content-ID: e22b7561-b5df-4b86-89c0-
b20702e2de83Content-Disposition: ...

...

--4FqyUUSfl7GyNwhB0PABKoF6PTFb6OvlContent-Type: ...Content-ID: 8a09b2b6-afdc-47d3-bc33-
5fda39d66463

...

--4FqyUUSfl7GyNwhB0PABKoF6PTFb6Ovl--

--0VUf5fZQGOkBjYIfaZ2yOZCi5OdMrt2AContent-ID: 5c44530a-8955-4514-8527-
eaddf24b30aeContent-Type: application/ms-conversation-context+xmlContent-Disposition:
render;handling=optional

<cc:XmlConvContext
xmlns:cc="http://schemas.microsoft.com/2008/03/sip/conversationContext">
<cc:id>0734aae0-a714-45d9-87bc-20ed9d432b80</cc:id>
```

```
<cc:from><cc:uri>sip:alice@contoso.com</cc:uri></cc:from>
<cc:to><cc:uri>sip:marco@contoso.com</cc:uri></cc:to>
<cc:participants>
<cc:participant>
<cc:uri>sip:alice@contoso.com</cc:uri>
<cc:displayName>Alice</cc:displayName>
</cc:participant>
<cc:participant>
<cc:uri>sip:bob@contoso.com</cc:uri>
</cc:participant>
</cc:participants>
<cc:date>2008-09-03T21:34:55.831063Z</cc:date>
<cc:mode>audio</cc:mode>
<cc:conversationId>a4f266f1a6914acb99cddef15659e38c</cc:conversationId>
<cc:dataFormat>text/plain</cc:dataFormat>
<cc:contextData>Waiting time: 00:00:18

Bob is calling, it's his birthday today.
</cc:contextData></cc:XmlConvContext>--0VUf5fZQGOkBjYIfaZ2yOZCi5OdMrt2A--
```

4.12 Multipart MIME

4.12.1 Two- level Multipart MIME

All content in section 4.12 follows the product behavior described in endnote <81>.

The following example shows a two-level multi-part MIME, as described in section 3.15.

```
Content-Type: multipart/alternative;boundary="----=_NextPart_000_0059_01C91A7C.B83AD4E0"
Content-Length: 4014
------=_NextPart_000_0059_01C91A7C.B83AD4E0
Content-Type: application/sdp
Content-Transfer-Encoding: 7bit
Content-Disposition: session; handling=optional; ms-proxy-2007fallback
v=0
o=- 0 0 IN IP4 10.80.20.10
s=session
c=IN IP4 10.80.20.10
b=CT:35980
t=0 0
m=audio 50019 RTP/AVP 114 111 112 115 116 4 8 0 97 13 118 101
k=base64:9Izc9LPyPH3s1sl7XB0umY6R1B8H93Ru2knWs9pLcqxIlsPKgGq9iLaWcNNy
a=candidate:lLh4oR2NlwKLCbqk7rt7UJdJqHFEn9QeGNyYH6y8lGo 1 gKxsnl/9hhaK8j1Bc2tp4g UDP
0.830 10.80.20.10 50019
a=candidate:lLh4oR2NlwKLCbqk7rt7UJdJqHFEn9QeGNyYH6y8lGo 2 gKxsnl/9hhaK8j1Bc2tp4g UDP
0.830 10.80.20.10 50014
a=candidate:fI9holTcjzGzlUSH+fI+8hpZi/D+Y0bREpI35R6xbOY 1 V4xXN538Z4zIurS6nPYZiw TCP
0.190 131.107.1.36 52668
a=candidate:fI9holTcjzGzlUSH+fI+8hpZi/D+Y0bREpI35R6xbOY 2 V4xXN538Z4zIurS6nPYZiw TCP
0.190 131.107.1.36 52668
a=candidate:8/ugcPvoRu7X7870q7LcuZOAz8H1w1UZ1iz0JcyBfNI 1 Hv+ChtZX/SeNamyISSwstQ UDP
0.490 131.107.1.36 58325
a=candidate:8/ugcPvoRu7X7870q7LcuZOAz8H1w1UZ1iz0JcyBfNI 2 Hv+ChtZX/SeNamyISSwstQ UDP
0.490 131.107.1.36 50664
a=candidate:HSUcTjchkwG7k7cMX0tALAz4bty/uV/KvfSkV7Cc73I 1 nbUV3FDCmrixfcyP4PwwVQ TCP
0.250 10.80.20.10 50019
a=candidate:HSUcTjchkwG7k7cMX0tALAz4bty/uV/KvfSkV7Cc73I 2 nbUV3FDCmrixfcyP4PwwVQ TCP
0.250 10.80.20.10 50019
a=cryptoscale:1 client AES_CM_128_HMAC_SHA1_80
inline:1KjtxsXPzJi3Llf7jhKlGv9YSEdr0sPzwx9p7wQ2|2^31|1:1
a=crypto:2 AES_CM_128_HMAC_SHA1_80
inline:xgZxo13cfXDz1Vflqw2x+EB5cCdBh2Q0gsZfmE8D|2^31|1:1
a=maxptime:200
a=rtcp:50014
a=rtpmap:114 x-msrta/16000
```

```
a=fmtp:114 bitrate=29000
a=rtpmap:111 SIREN/16000
a=fmtp:111 bitrate=16000
a=rtpmap:112 G7221/16000
a=fmtp:112 bitrate=24000
a=rtpmap:115 x-msrta/8000
a=fmtp:115 bitrate=11800
a=rtpmap:116 AAL2-G726-32/8000
a=rtpmap:4 G723/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:97 RED/8000
a=rtpmap:13 CN/8000
a=rtpmap:118 CN/16000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=encryption:required
------=_NextPart_000_0059_01C91A7C.B83AD4E0
Content-Type: application/sdp
Content-Transfer-Encoding: 7bit
Content-Disposition: session; handling=optional
v=0
o=- 0 0 IN IP4 10.80.20.10
s=session
c=IN IP4 10.80.20.10
b=CT:35980
t=0 0
m=audio 50023 RTP/AVP 114 111 112 115 116 4 8 0 97 13 118 101
k=base64:9Izc9LPyPH3s1sl7XB0umY6R1B8H93Ru2knWs9pLcqxIlsPKgGq9iLaWcNNy
a=ice-ufrag:wdB31g
a=ice-pwd:yAbXGTFPoM+Kt2+fvhUUdKkclwSChFQj
a=candidate:1 1 UDP 2130706431 10.80.20.10 50023 typ host
a=candidate:1 2 UDP 2130705918 10.80.20.10 50016 typ host
a=candidate:2 1 TCP-PASS 6556159 131.107.1.36 50370 typ relay raddr 131.107.1.36 rport
50370
a=candidate:2 2 TCP-PASS 6556158 131.107.1.36 50370 typ relay raddr 131.107.1.36 rport
50370
a=candidate:3 1 UDP 16648703 131.107.1.36 56997 typ relay raddr 131.107.1.36 rport 56997
a=candidate:3 2 UDP 16648702 131.107.1.36 56644 typ relay raddr 131.107.1.36 rport 56644
a=candidate:4 1 TCP-ACT 7076863 131.107.1.36 50370 typ relay raddr 131.107.1.36 rport
50370
a=candidate:4 2 TCP-ACT 7076350 131.107.1.36 50370 typ relay raddr 131.107.1.36 rport
50370
a=candidate:5 1 TCP-ACT 1684797951 10.80.20.10 50018 typ srflx raddr 10.80.20.10 rport
50018
a=candidate:5 2 TCP-ACT 1684797438 10.80.20.10 50018 typ srflx raddr 10.80.20.10 rport
50018
a=cryptoscale:1 client AES_CM_128_HMAC_SHA1_80
inline:1KjtxsXPzJi3Llf7jhKlGv9YSEdr0sPzwx9p7wQ2|2^31|1:1
a=crypto:2 AES_CM_128_HMAC_SHA1_80
inline:xgZxo13cfXDz1Vflqw2x+EB5cCdBh2Q0gsZfmE8D|2^31|1:1
a=maxptime:200
a=rtcp:50016
a=rtpmap:114 x-msrta/16000
a=fmtp:114 bitrate=29000
a=rtpmap:111 SIREN/16000
a=fmtp:111 bitrate=16000
a=rtpmap:112 G7221/16000
a=fmtp:112 bitrate=24000
a=rtpmap:115 x-msrta/8000
a=fmtp:115 bitrate=11800
a=rtpmap:116 AAL2-G726-32/8000
a=rtpmap:4 G723/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:97 RED/8000
a=rtpmap:13 CN/8000
a=rtpmap:118 CN/16000
a=rtpmap:101 telephone-event/8000
```

```
a=fmtp:101 0-16
a=encryption:required
------=_NextPart_000_0059_01C91A7C.B83AD4E0--
```

4.12.2 Three- level Multipart MIME

The following example shows a three-level multi-part MIME, as described in section 3.15.

```
Content-Type: multipart/mixed; boundary=HkS4RpzThV2XRK9lcuE3NJUcskesnr9w
Content-Type: multipart/alternative; boundary=sYRNyS9rx1iUksZ4fH8roFi2MbQU6dbo
--sYRNyS9rx1iUksZ4fH8roFi2MbQU6dbo
Content-Type: application/sdp
Content-ID: ccbe8227-c734-4d4a-b1ce-0ed219097ff4
Content-Disposition: session;handling=optional;ms-proxy-2007fallback
v=0
o=- 0 0 IN IP4 172.29.105.158
s=session
c=IN IP4 172.29.105.158
b=CT:1000
t=0 0
m=audio 23160 RTP/AVP 8 0 4 116 3 115 112 111 114 13 118 97 101
c=IN IP4 172.29.105.158
a=rtcp:29398
a=candidate:mDUVW7BtzxI1duehZtgEB9+HmyHI2DNgAY1V0UrdYIo 1 tKxTKKdnyDIj5nLnGLIXpw UDP
0.900 172.29.105.158 23160
a=candidate:mDUVW7BtzxI1duehZtgEB9+HmyHI2DNgAY1V0UrdYIo 2 tKxTKKdnyDIj5nLnGLIXpw UDP
0.900 172.29.105.158 29398
a=candidate:6pJIvJXR/PECSSKwaR+ygUx9hRd360XbnImL36GTD6M 1 eaPFs6Wp3vVT+WMStx5WDg TCP
0.150 172.29.105.171 51143
a=candidate:6pJIvJXR/PECSSKwaR+ygUx9hRd360XbnImL36GTD6M 2 eaPFs6Wp3vVT+WMStx5WDg TCP
0.150 172.29.105.171 51143
a=candidate:HuZ/qrwBjoj/TpiTR07CLJpJ1JpKVzjHu+EYh5G8uTg 1 ut9XFV7u5hWESZuqESPHLQ UDP
0.450 172.29.105.171 53824
a=candidate:HuZ/qrwBjoj/TpiTR07CLJpJ1JpKVzjHu+EYh5G8uTg 2 ut9XFV7u5hWESZuqESPHLQ UDP
0.450 172.29.105.171 52048
a=candidate:1/UjDo+KnYxwlJvWgELKP93RoXKk+vOKxfjCHpmh9nk 1 73jZjOF9LVx/jQTKT/bySA TCP
0.250 172.29.105.158 3512
a=candidate:1/UjDo+KnYxwlJvWgELKP93RoXKk+vOKxfjCHpmh9nk 2 73jZjOF9LVx/jQTKT/bySA TCP
0.250 172.29.105.158 3512
a=cryptoscale:1 client AES_CM_128_HMAC_SHA1_80
inline:/h4AObPXOlrc7LkgLj03byQ7PVvuzfmwx3NJXn1+|2^31|1:1
a=crypto:2 AES_CM_128_HMAC_SHA1_80
inline:OR/d0mnfMTRGa6IFw0JN5CeR6ZwMTWTWoz54IiOM|2^31|1:1
a=crypto:3 AES_CM_128_HMAC_SHA1_80 inline:ha8qW6njHa9nEDqV78Iy1aDfDQb3dsXidivURp0+|2^31
a=label:main-audio
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:4 G723/8000
a=rtpmap:116 AAL2-G726-32/8000
a=rtpmap:3 GSM/8000
a=rtpmap:115 x-msrta/8000
a=fmtp:115 bitrate=11800
a=rtpmap:112 G7221/16000
a=fmtp:112 bitrate=24000
a=rtpmap:111 SIREN/16000
a=fmtp:111 bitrate=16000
a=rtpmap:114 x-msrta/16000
a=fmtp:114 bitrate=29000
a=rtpmap:13 CN/8000
a=rtpmap:118 CN/16000
a=rtpmap:97 RED/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
--sYRNyS9rx1iUksZ4fH8roFi2MbQU6dbo
Content-Type: application/sdp
Content-ID: 38fcdc48-dc5e-48a0-9681-532010d92196
```

```
v=0
o=- 0 0 IN IP4 172.29.105.158
s=session
c=IN IP4 172.29.105.158
b=CT:1000
t=0 0
m=audio 25170 RTP/AVP 8 0 4 116 3 115 112 111 114 13 118 97 101
c=IN IP4 172.29.105.158
a=rtcp:14396
a=ice-ufrag:2UclRQ
a=ice-pwd:So72NmoVpGdlUE7zWhKQKsP+zteJmfSc
a=candidate:1 1 UDP 2130706431 172.29.105.158 25170 typ host
a=candidate:1 2 UDP 2130705918 172.29.105.158 14396 typ host
a=candidate:2 1 tcp-pass 6555135 172.29.105.171 56700 typ relay raddr 172.29.105.171
rport 56700
a=candidate:2 2 tcp-pass 6555134 172.29.105.171 56700 typ relay raddr 172.29.105.171
rport 56700
a=candidate:3 1 UDP 16647679 172.29.105.171 53833 typ relay raddr 172.29.105.171 rport
53833
a=candidate:3 2 UDP 16647678 172.29.105.171 57341 typ relay raddr 172.29.105.171 rport
57341
a=candidate:4 1 tcp-act 7076863 172.29.105.171 56700 typ relay raddr 172.29.105.171 rport
56700
a=candidate:4 2 tcp-act 7076350 172.29.105.171 56700 typ relay raddr 172.29.105.171 rport
56700
a=candidate:5 1 tcp-act 1684797951 172.29.105.158 26980 typ srflx raddr 172.29.105.158
rport 26980
a=candidate:5 2 tcp-act 1684797438 172.29.105.158 26980 typ srflx raddr 172.29.105.158
rport 26980
a=cryptoscale:1 client AES_CM_128_HMAC_SHA1_80
inline:/h4AObPXOlrc7LkgLj03byQ7PVvuzfmwx3NJXn1+|2^31|1:1
a=crypto:2 AES_CM_128_HMAC_SHA1_80
inline:OR/d0mnfMTRGa6IFw0JN5CeR6ZwMTWTWoz54IiOM|2^31|1:1
a=crypto:3 AES_CM_128_HMAC_SHA1_80 inline:ha8qW6njHa9nEDqV78Iy1aDfDQb3dsXidivURp0+|2^31
a=label:main-audio
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:4 G723/8000
a=rtpmap:116 AAL2-G726-32/8000
a=rtpmap:3 GSM/8000
a=rtpmap:115 x-msrta/8000
a=fmtp:115 bitrate=11800
a=rtpmap:112 G7221/16000
a=fmtp:112 bitrate=24000
a=rtpmap:111 SIREN/16000
a=fmtp:111 bitrate=16000
a=rtpmap:114 x-msrta/16000
a=fmtp:114 bitrate=29000
a=rtpmap:13 CN/8000
a=rtpmap:118 CN/16000
a=rtpmap:97 RED/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
--sYRNyS9rx1iUksZ4fH8roFi2MbQU6dbo--
--HkS4RpzThV2XRK9lcuE3NJUcskesnr9w
Content-ID: 2b700e68-70cd-4de9-b8e6-78625ca48b3f
CONTENT-TYPE: application/ms-conversation-context+xml
Content-Disposition: render;handling=optional
<cc:XmlConvContext
xmlns:cc="http://schemas.microsoft.com/2008/03/sip/conversationContext">
  <cc:id>fb578ae6-577c-4f9f-8510-d74c29c71e2e</cc:id>
  <cc:from>
    <cc:uri>sip:help_desk@fabrikam.com</cc:uri>
  </cc:from>
  <cc:to>
    <cc:uri>sip:Agent9@fabrikam.com</cc:uri>
  </cc:to>
  <cc:participants>
    <cc:participant>
```

```
        <cc:uri>sip:danp@fabrikam.com</cc:uri>
        <cc:displayName>Dan Park</cc:displayName>
      </cc:participant>
      <cc:participant>
        <cc:uri>sip:help_desk@fabrikam.com</cc:uri>
      </cc:participant>
      <cc:participant>
        <cc:uri>sip:Agent9@fabrikam.com</cc:uri>
      </cc:participant>
    </cc:participants>
    <cc:date>2008-09-11T21:07:33.6378654Z</cc:date>
    <cc:mode>audio</cc:mode>
    <cc:conversationId>61020efc64bb4f2f87f631c99bb65b7e</cc:conversationId>
    <cc:dataFormat>text/plain</cc:dataFormat>
    <cc:contextData>Waiting time: 00:00:05
  IVR information:
  Question: Press or say one for Benefits press or say two for Human Resources
  Answer: 1
  </cc:contextData>
  </cc:XmlConvContext>
  --HkS4RpzThV2XRK9lcuE3NJUcskesnr9w--
```

4.13 Agent Anonymity

This section follows the product behavior described in endnote <82>.

The following example shows the INVITE a server endpoint sends to establish an anonymous call, excluding common required headers and the SDP part.

```
INVITE sip:Alice@contoso.com;gruu;opaque=user:epid:qIIWS2j5AVeD_HxnQdxmlwAA SIP/2.0
From: sip:Bob@contoso.com;epid=02020202;tag=02020202
To: sip:Alice@contoso.com;
Call-Id: f0ec9c595c1f412ca6b71318beb599bb
Via: SIP/2.0/TLS 192.0.2.1:27221;branch=z9hG4bKa8d4
Cseq: 4 INVITE
Ms-Call-Info: Rgs.Anonymization
Contact:<sip:server1@contoso.com;gruu;opaque=srvr:HomeServer:VWIdpJWTA1eatgf05sHGswAA>;au
tomata;actor="attendant";text;audio;video;image
```

In this example, the server endpoint is impersonating Bob. The contact remains the server endpoint GRUU.

The following example show the 200 OK response a server endpoint sends to establish an anonymous call initiated by a user endpoint, excluding common required headers and the SDP part.

```
SIP/2.0 200 OK
From: sip:Alice@contoso.com;epid=02020202;tag=02020202
To: sip:Helpdesk@contoso.com;epid=01010101;tag=01010101
Call-Id: f0ec9c595c1f412ca6b71318beb599bb
Via: SIP/2.0/TLS 192.0.2.1:27221;branch=z9hG4bKa8d4
Cseq: 4 INVITE
Ms-Call-Info: Rgs.Anonymization
Contact:<sip:server1@contoso.com;gruu;opaque=srvr:HomeServer:VWIdpJWTA1eatgf05sHGswAA>;au
tomata;actor="attendant";text;audio;video;image
```

The following example show the request a client endpoint (5) can send to request a call on behalf of the Helpdesk and the response from the server (2) endpoint (5), using anonymity and excluding common required headers and the SDP part.

```
INVITE sip:Helpdesk@contoso.com;gruu;opaque=user:epid:qIIWS2j5AVeD_HxnQdxmlwAA SIP/2.0
```

```
From: sip:Alice@contoso.com;epid=02020202;tag=02020202
To: sip:Bob@contoso.com;
Call-Id: f0ec9c595c1f412ca6b71318beb599bb
Via: SIP/2.0/TLS 192.0.2.1:27221;branch=z9hG4bKa8d4
Cseq: 4 INVITE
P-Agent-On-Behalf-Of: sip:Helpdesk@contoso.com

SIP/2.0 200 OK
From: sip:Alice@contoso.com;epid=02020202;tag=02020202
To: sip:Bob@contoso.com;epid=01010101;tag=01010101
Call-Id: f0ec9c595c1f412ca6b71318beb599bb
Via: SIP/2.0/TLS 192.0.2.1:27221;branch=z9hG4bKa8d4
Cseq: 4 INVITE
Ms-Call-Info: Rgs.Anonymization
Contact:<sip:server1@contoso.com;gruu;opaque=srvr:HomeServer:VWIdpJWTA1eatgf05sHGswAA>;au
tomata;actor="attendant";text;audio;video;image
```

4.14 E911 INVITE

This section follows the product behavior described in endnote <83>.

The following example shows an E911 INVITE that the client endpoint can send to establish an E911 call. This example excludes common required headers.

```
INVITE sip:911;phone-context=Redmond@192.168.1.12;user=phone SIP/2.0
From: "voip_911_user1"<sip:voip_911_user1@contoso.com>;epid=1D19090AED;tag=d04d65d924
To: <sip:911;phone-context=Redmond@192.168.1.12;user=phone>
CSeq: 8 INVITE
Call-ID: e6828be1-1cdd-4fb0-bdda-cda7faf46df4
VIA: SIP/2.0/TLS 192.168.0.244:57918;branch=z9hG4bK528b7ad7
CONTACT:
<sip:voip_911_user1@contoso.com;opaque=user:epid:R4bCDaUj51a06PUbkraS0QAA;gruu>;text;audio;vi
deo;image
PRIORITY: emergency
Supported: geolocation
CONTENT-TYPE: multipart/mixed; boundary= ------=_NextPart_000_4A6D_01CAB3D6.7519F890
geolocation: <cid:sip:voip_911_user1@contoso.com>;inserted-by="sip:voip_911_user1@contoso
.com"
Message-Body:
------=_NextPart_000_4A6D_01CAB3D6.7519F890
Content-Type: application/sdp ; charset=utf-8
v=0
o=- 0 0 IN IP4 Client
s=session
c=IN IP4 Client
t=0 0
m=audio 30684 RTP/AVP 114 111 112 115 116 4 3 8 0 106 97
c=IN IP4 172.29.105.23
a=rtcp:60423
a=label:Audio
a=rtpmap:3 GSM/8000/1
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=ptime:20

------=_NextPart_000_4A6D_01CAB3D6.7519F890
Content-Type: application/pidf+xml
Content-ID: <voip_911_user1@contoso.com>
<?xml version="1.0" encoding="utf-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
  xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
  xmlns:bp="urn:ietf:params:xml:ns:pidf:geopriv10:basicPolicy"
  xmlns:ca="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"
  xmlns:ms="urn:schema:Rtc.LIS.msftE911PidfExtn.2008"
```

```
  entity="sip:voip_911_user1@contoso.com">
  &lt;tuple id="0"&gt;
    &lt;status&gt;
      &lt;gp:geopriv&gt;
        &lt;gp:location-info&gt;
          &lt;ca:civicAddress&gt;
            &lt;ca:country&gt;US&lt;/ca:country&gt;
            &lt;ca:A1&gt;WA&lt;/ca:A1&gt;
            &lt;ca:A3&gt;Redmond&lt;/ca:A3&gt;
            &lt;ca:RD&gt;163rd&lt;/ca:RD&gt;
            &lt;ca:STS&gt;Ave&lt;/ca:STS&gt;
            &lt;ca:POD&gt;NE&lt;/ca:POD&gt;
            &lt;ca:HNO&gt;3910&lt;/ca:HNO&gt;
            &lt;ca:LOC&gt;40/4451&lt;/ca:LOC&gt;
            &lt;ca:NAM&gt;Contoso Corporation &lt;/ca:NAM&gt;
            &lt;ca:PC&gt;98052&lt;/ca:PC&gt;
          &lt;/ca:civicAddress&gt;
        &lt;/gp:location-info&gt;
        &lt;gp:usage-rules&gt;
          &lt;bp:retransmission-allowed&gt;true&lt;/bp:retransmission-allowed&gt;
        &lt;/gp:usage-rules&gt;
      &lt;/gp:geopriv&gt;
      &lt;ms:msftE911PidfExtn&gt;
        &lt;ms:ConferenceUri&gt;sip:+14255550199@contoso.com;user=phone
          &lt;/ms:ConferenceUri&gt;
        &lt;ms:ConferenceMode&gt;twoway&lt;/ms:ConferenceMode&gt;
        &lt;LocationPolicyTagID xmlns="urn:schema:Rtc.Lis.LocationPolicyTagID.2008"&gt;user-tagid
          &lt;/LocationPolicyTagID &gt;
      &lt;/ms:msftE911PidfExtn&gt;
    &lt;/status&gt;
    &lt;timestamp&gt;1991-09-22T13:37:31.03&lt;/timestamp&gt;
  &lt;/tuple&gt;
&lt;/presence&gt;
------=_NextPart_000_4A6D_01CAB3D6.7519F890--
```

5 Security

None.

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: Full Routing Script Preamble Format

Following is the full XML schema for the routing script preamble:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://schemas.microsoft.com/02/2006/sip/routing"
xmlns:tns="http://schemas.microsoft.com/02/2006/sip/routing"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <!-- The following type definations are used in the preamble-->
  <xs:complexType name="target-type">
    <xs:annotation>
      <xs:documentation>At least one of uri or application attributes are required to be
present.</xs:documentation>
    </xs:annotation>
    <xs:attribute name="uri" type="xs:string" use="optional" />
    <xs:attribute name="application" type="xs:string" use="optional" />
    <xs:anyAttribute namespace="##any" processContents="lax" />
  </xs:complexType>
  <xs:complexType name="timezone-date-type">
    <xs:attribute name="name" type="xs:string" use="optional" />
    <xs:attribute name="bias" type="xs:integer" use="required" />
    <xs:attribute name="year" type="xs:short" use="required" />
    <xs:attribute name="month" type="xs:short" use="required" />
    <xs:attribute name="dayofweek" type="xs:short" use="required" />
    <xs:attribute name="day" type="xs:short" use="required" />
    <xs:attribute name="hour" type="xs:short" use="required" />
    <xs:attribute name="minute" type="xs:short" use="required" />
  </xs:complexType>
  <xs:complexType name="timezone-type">
    <xs:annotation>
      <xs:documentation>This type is based of the TIME_ZONE_INFORMATION type from Win32
API.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="standard" type="tns:timezone-date-type" />
      <xs:element name="daylight" type="tns:timezone-date-type" />
    </xs:sequence>
    <xs:attribute name="bias" type="xs:integer" use="required" />
  </xs:complexType>
  <xs:complexType name="period-type">
    <xs:attribute name="dow" type="tns:days-of-week-type" use="required" />
    <xs:attribute name="start" type="tns:minutes-from-midnight-type" use="required" />
    <xs:attribute name="end" type="tns:minutes-from-midnight-type" use="required" />
  </xs:complexType>
  <xs:complexType name="period-array-type">
    <xs:sequence>
      <xs:element name="period" type="tns:period-type" minOccurs="0"
maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="refname-type">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Za-z0-9_]+" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="preamble-member-base-type">
    <xs:attribute name="name" type="tns:refname-type" use="required" />
  </xs:complexType>
  <xs:complexType name="wait-type">
    <xs:complexContent>
      <xs:extension base="tns:preamble-member-base-type">
        <xs:attribute name="seconds" type="xs:nonNegativeInteger" use="required" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="list-type">
```

```
      <xs:complexContent>
        <xs:extension base="tns:preamble-member-base-type">
          <xs:sequence>
            <xs:element name="target" type="tns:target-type" minOccurs="0"
maxOccurs="unbounded" />
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="time-type">
      <xs:complexContent>
        <xs:extension base="tns:preamble-member-base-type">
          <xs:sequence>
            <xs:element name="timezone" type="tns:timezone-type" minOccurs="0"
maxOccurs="1" />
          </xs:sequence>
          <xs:attribute name="range" type="xs:string" use="required" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="time-period-type">
      <xs:complexContent>
        <xs:extension base="tns:preamble-member-base-type">
          <xs:sequence>
            <xs:element name="timezone" type="tns:timezone-type" minOccurs="0"
maxOccurs="1" />
            <xs:element name="periodarray" type="tns:period-array-type" />
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="flags-type">
      <xs:complexContent>
        <xs:extension base="tns:preamble-member-base-type">
          <xs:attribute name="value" type="xs:string" use="required" />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="preamble-type">
      <xs:sequence>
        <xs:choice minOccurs="0" maxOccurs="unbounded">
          <xs:element name="flags" type="tns:flags-type" />
          <xs:element name="time" type="tns:time-type" />
          <xs:element name="timeperiod" type="tns:time-period-type" />
          <xs:element name="wait" type="tns:wait-type" />
          <xs:element name="list" type="tns:list-type" />
        </xs:choice>
      </xs:sequence>
    </xs:complexType>
    <xs:simpleType name="minutes-from-midnight-type">
      <xs:restriction base="xs:integer">
        <xs:minInclusive value="0" />
        <xs:maxInclusive value="1440" />
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="day-of-week-type">
      <xs:restriction base="xs:string">
        <xs:enumeration value="sun" />
        <xs:enumeration value="mon" />
        <xs:enumeration value="tue" />
        <xs:enumeration value="wed" />
        <xs:enumeration value="thu" />
        <xs:enumeration value="fri" />
        <xs:enumeration value="sat" />
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="days-of-week-type">
      <xs:list itemType="tns:day-of-week-type" />
    </xs:simpleType>
```

```
  <!-- The following type definations are used in the script-->
  <xs:simpleType name="criteria-type">
    <xs:restriction base="xs:string">
      <xs:pattern value="!{0,1}dnd" />
      <xs:pattern value="!{0,1}umenabled" />
      <xs:pattern value="!{0,1}class:(primary|secondary)" />
      <xs:pattern value="!{0,1}registered" />
      <xs:pattern value="!{0,1}time:[A-Za-z0-9_]+" />
      <xs:pattern value="!{0,1}flags:[A-Za-z0-9_]+\(.*\)" />
      <xs:pattern value="!{0,1}member:[A-Za-z0-9_]+" />
      <xs:pattern value="!{0,1}workinghours" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="reference-type">
    <xs:attribute name="name" type="tns:refname-type" use="required" />
  </xs:complexType>

  <!-- Root document defintion -->
  <xs:complexType name="routing-type">
    <xs:annotation>
      <xs:documentation>The name and version attributes are both mandatory.
</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="preamble" type="tns:preamble-type" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" />
    <xs:attribute name="version" type="xs:integer" />
    <xs:attribute name="minSupportedClientVersion" type="xs:string" use="optional" />
  </xs:complexType>
  <xs:element name="routing" type="tns:routing-type" />
</xs:schema>
```

7 Appendix B: Full Location Profile Format

Following is the full XML schema for the full location profile:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.microsoft.com/2007/03/LocationProfileDescription"
targetNamespace="http://schemas.microsoft.com/2007/03/LocationProfileDescription">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Service Request for Location Profile Schema
      Microsoft Unified Communications Group
    </xsd:documentation>
  </xsd:annotation>

  <xsd:element name="LocationProfileDescription" type="LocationProfileDescriptionType"/>

  <xsd:element name="Name" type="xsd:string"/>
  <xsd:element name="ExternalAccessPrefix" type="xsd:string"/>
  <xsd:element name="OptimizeDeviceDialing" type="xsd:boolean"/>
  <xsd:complexType name="RuleType">
    <xsd:sequence>
      <xsd:element name="Pattern"   type="xsd:string"/>
      <xsd:element name="Translation" type="xsd:string"/>
    <xsd:element name="InternalEnterpriseExtension" type="xsd:boolean" minOccurs="0"/>
    <xsd:element name="ApplicableForDeviceDialing" type="xsd:boolean" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="LocationProfileDescriptionType">
    <xsd:sequence>
      <xsd:element ref="Name" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="Rule" type="RuleType" minOccurs="1" maxOccurs="unbounded"/>
    <xsd:element ref="ExternalAccessPrefix" minOccurs="0" maxOccurs="0"/>
    <xsd:element ref="OptimizeDeviceDialing" minOccurs="0" maxOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>

</xsd:schema>
```

8 Appendix C: Full Call Context Format

Following is the schema for call context data.

```
<?xml version="1.0"       encoding="UTF-8"?>
<xs:schema version="1.0"
targetNamespace="http://schemas.microsoft.com/2008/03/sip/conversationContext"
xmlns:callctns="http://schemas.microsoft.com/2008/03/sip/conversationContext"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">

  <xs:annotation>
    <xs:documentation>Notes/Context associated with a conversation </xs:documentation>
  </xs:annotation>

  <xs:complexType name="XmlConvContextParticipantType">
    <xs:sequence>
      <xs:element name="uri" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="displayName" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="onBehalfUri" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="onBehalfDisplayName" type="xs:string" minOccurs="0"
maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>


  <xs:complexType name="XmlConvContextParticipantCollectionType">
    <xs:sequence>
      <xs:element name="participant" type="callctns:XmlConvContextParticipantType"
minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>


  <xs:complexType name="XmlConvContextType" >
    <xs:sequence>
      <xs:element name="id" type="xs:token"  minOccurs="1" maxOccurs="1"/>
      <xs:element name="from" type="callctns:XmlConvContextParticipantType" minOccurs="1"
maxOccurs="1"/>
      <xs:element name="to" type="callctns:XmlConvContextParticipantType" minOccurs="1"
maxOccurs="1"/>
      <xs:element name="participants"
type="callctns:XmlConvContextParticipantCollectionType" minOccurs="1" maxOccurs="1" />
      <xs:element name="date" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
      <xs:element name="mode" type="xs:token" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="conversationId" type="xs:token" minOccurs="1" maxOccurs="1"/>
      <xs:element name="dataFormat" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="contextData" type="xs:string" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="XmlConvContext" type="callctns:XmlConvContextType" />

</xs:schema>
```

9 Appendix D: E911 PIDF Extension Format

Following is the full XML schema for the E911 PIDF extension:

```
<xs:schema xmlns:pidftns="urn:schema:Rtc.LIS.msftE911PidfExtn.2008"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="urn:schema:Rtc.LIS.msftE911PidfExtn.2008"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="msftE911PidfExtn" type="pidftns:msftE911PidfExtn" />
  <xs:complexType name="msftE911PidfExtn">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="ConferenceUri" type="xs:anyURI" />
      <xs:element minOccurs="1" maxOccurs="1" name="ConferenceMode"
type="pidftns:ConferenceModeEnum" />
      <xs:any minOccurs="0" maxOccurs="unbounded" namespace="##other" processContents="lax"
/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any" />
  </xs:complexType>
  <xs:simpleType name="ConferenceModeEnum">
    <xs:restriction base="xs:token">
      <xs:enumeration value="oneway" />
      <xs:enumeration value="twoway" />
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

The **msftE911PidfExtn** also contains an extensibility element that contains the value of the **LocationPolicyTagID** property returned in the **LocationPolicy** in-band provisioning group.

```
<LocationPolicyTagID xmlns="urn:schema:Rtc.Lis.LocationPolicyTagID.2008">location-policy-tag-
id-value</LocationPolicyTagID >
```

10 Appendix E: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs.

- Microsoft Office Communications Server 2007
- Microsoft Office Communications Server 2007 R2
- Microsoft Office Communicator 2007
- Microsoft Office Communicator 2007 R2
- Microsoft Lync Server 2010
- Microsoft Lync 2010
- Microsoft Lync Server 2013
- Microsoft Lync Client 2013/Skype for Business

1. Microsoft Skype for Business 2016

2. Microsoft Skype for Business Server 2015

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 2.2.2: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<2> Section 2.2.5: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<3> Section 2.2.8.1: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<4> Section 2.2.8.1: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<5> Section 2.2.8.2: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<6> Section 2.2.8.2: Supported in Office Communications Server 2007 R2, Office Communicator 2007 R2 only.

<7> Section 2.2.9: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<8> Section 2.2.9: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<9> Section 2.2.9.2: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<10> Section 2.2.14: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<11> Section 2.2.17: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<12> Section 2.2.18: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<13> Section 2.2.19: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<14> Section 2.2.20: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<15> Section 2.2.21: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<16> Section 2.2.22: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<17> Section 3.6.3: Supported in Office Communications Server 2007, Office Communicator 2007 only.

<18> Section 3.6.3: Supported in Office Communications Server 2007, Office Communicator 2007 only.

<19> Section 3.6.3: Supported in Office Communications Server 2007, Office Communicator 2007 only.

<20> Section 3.6.3: Supported in Office Communications Server 2007, Office Communicator 2007 only.

<21> Section 3.6.3: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<22> Section 3.6.3: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<23> Section 3.6.5.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<24> Section 3.6.5.1: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<25> Section 3.6.6: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<26> Section 3.6.6: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<27> Section 3.7: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<28> Section 3.7: Office Communications Server 2007, Office Communicator 2007, Office Communicator 2007 R2, Office Communications Server 2007 R2: This behavior is not supported.

<29> Section 3.7.5.2: Office Communicator 2007, Office Communications Server 2007, Office Communicator 2007 R2, Office Communications Server 2007 R2: This behavior is not supported.

<30> Section 3.7.5.3.1: Office Communicator 2007, Office Communications Server 2007, Office Communicator 2007 R2, Office Communications Server 2007 R2: This behavior is not supported.

<31> Section 3.9.2.3: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<32> Section 3.9.2.4: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<33> Section 3.9.5.1: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<34> Section 3.9.5.1: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<35> Section 3.9.5.1.1: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<36> Section 3.9.5.1.2: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<37> Section 3.9.5.1.2: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<38> Section 3.9.5.1.2: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<39> Section 3.9.5.1.2: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<40> Section 3.9.5.1.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<41> Section 3.9.5.1.3: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<42> Section 3.9.5.1.3: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<43> Section 3.9.5.1.3: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<44> Section 3.9.5.1.4: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<45> Section 3.9.5.1.4: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<46> Section 3.9.5.1.4: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<47> Section 3.9.5.1.4: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<48> Section 3.9.5.1.4: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<49> Section 3.9.5.2.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<50> Section 3.9.5.2.2: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<51> Section 3.9.5.2.2: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<52> Section 3.9.5.2.2: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<53> Section 3.9.5.2.2: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<54> Section 3.9.5.2.2: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<55> Section 3.9.5.2.2.2: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<56> Section 3.9.5.2.2.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: The primary user timer is not supported.

<57> Section 3.9.5.2.2.3: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<58> Section 3.9.5.2.2.4: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<59> Section 3.9.5.5: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<60> Section 3.9.5.8: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<61> Section 3.9.5.10: This behavior is not supported in Office Communicator 2007 or Office Communications Server 2007.

<62> Section 3.9.5.10: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<63> Section 3.9.5.10: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: This behavior is not supported.

<64> Section 3.9.5.10: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: This behavior is not supported.

<65> Section 3.9.6.3: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<66> Section 3.9.6.3: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<67> Section 3.9.6.4: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<68> Section 3.10.1.5: The **ms-remote-fqdn** parameter is only available in Lync Server 2013.

<69> Section 3.12: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<70> Section 3.13: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<71> Section 3.14: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<72> Section 3.15: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<73> Section 3.15.4.1: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<74> Section 3.15.5.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<75> Section 3.16: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: Extensions for Agent Anonymity. Ms-Call-Info and P-Agent-On-Behalf-Of are not supported.

<76> Section 3.17: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: E911 message processing is not supported.

<77> Section 4.6: This example does not apply to: Office Communications Server 2007, Office Communicator 2007.

<78> Section 4.7.4: Office Communications Server 2007, Office Communicator 2007. This behavior is not supported.

<79> Section 4.8: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<80> Section 4.11: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<81> Section 4.12.1: This example does not apply to: Office Communications Server 2007, Office Communicator 2007.

<82> Section 4.13: This example does not apply to: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2.

<83> Section 4.14: This example does not apply to Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, and Office Communicator 2007 R2.

11 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- The removal of a document from the documentation set.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the technical content of the document is identical to the last released version.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.
- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated**.

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
10 Appendix E: Product Behavior	Updated the list format for applicable products.	N	Content update.

12 Index

E

F

G

H

I

N

O

P

R

S

T

V