[MS-E911WS]:

Web Service for E911 Support Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks**. The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
3/31/2010	0.1	Major	Initial Availability
4/30/2010	0.2	Editorial	Revised and edited the technical content
6/7/2010	0.3	Editorial	Revised and edited the technical content
6/29/2010	0.4	Editorial	Changed language and formatting in the technical content.
7/23/2010	0.4	No Change	No changes to the meaning, language, or formatting of the technical content.
9/27/2010	1.0	Major	Significantly changed the technical content.
11/15/2010	1.0	No Change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.0	No Change	No changes to the meaning, language, or formatting of the technical content.
3/18/2011	1.0	No Change	No changes to the meaning, language, or formatting of the technical content.
6/10/2011	1.0	No Change	No changes to the meaning, language, or formatting of the technical content.
1/20/2012	2.0	Major	Significantly changed the technical content.
4/11/2012	2.0	No Change	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	2.0	No Change	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	2.0.1	Editorial	Changed language and formatting in the technical content.
2/11/2013	2.0.1	No Change	No changes to the meaning, language, or formatting of the technical content.
7/30/2013	2.1	Minor	Clarified the meaning of the technical content.
11/18/2013	2.1	No Change	No changes to the meaning, language, or formatting of the technical content.
2/10/2014	2.1	No Change	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	2.1	No Change	No changes to the meaning, language, or formatting of the technical content.
7/31/2014	2.1	No Change	No changes to the meaning, language, or formatting of the technical content.
10/30/2014	2.2	Minor	Clarified the meaning of the technical content.
9/4/2015	2.2	No Change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

Web Service for E911 Support Protocol specifies the Web Service for E911 Support Protocol interface that is used by protocol clients to retrieve locations associated with network identifiers, or locations within a city. A location is a civic address with up to room-level granularity. The network identifiers that can be specified are the Wireless Access Point, Received Signal Strength Indication, Media Access Control Address, Chassis, Port, Subnet, and Internet Protocol Address.

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in [RFC2119]. Sections 1.5 and 1.9 are also normative but do not contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

The following terms are specific to this document:

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

binary large object (BLOB): A discrete packet of data that is stored in a database and is treated as a sequence of uninterpreted bytes.

certificate: A certificate is a collection of attributes (1) and extensions that can be stored persistently. The set of attributes in a certificate can vary depending on the intended usage of the certificate. A certificate securely binds a public key to the entity that holds the corresponding private key. A certificate is commonly used for **authentication** and secure exchange of information on open networks, such as the Internet, extranets, and intranets. Certificates are digitally signed by the issuing certification authority (CA) and can be issued for a user, a computer, or a service. The most widely accepted format for certificates is defined by the ITU-T X.509 version 3 international standards. For more information about attributes and extensions, see [RFC3280] and [X509] sections 7 and 8.

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

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.

Hypertext Transfer Protocol (HTTP): An application-level protocol for distributed, collaborative, hypermedia information systems (text, graphic images, sound, video, and other multimedia files) on the World Wide Web.

Hypertext Transfer Protocol Secure (HTTPS): An extension of HTTP that securely encrypts and decrypts web page requests. In some older protocols, "Hypertext Transfer Protocol over Secure Sockets Layer" is still used (Secure Sockets Layer has been deprecated). For more information, see [SSL3] and [RFC5246].

Kerberos: An **authentication** system that enables two parties to exchange private information across an otherwise open network by assigning a unique key (called a ticket) to each user that logs on to the network and then embedding these tickets into messages sent by the users. For more information, see [MS-KILE].

NT LAN Manager (NTLM) Authentication Protocol: A protocol using a challenge-response mechanism for **authentication** in which clients are able to verify their identities without sending a password to the server. It consists of three messages, commonly referred to as Type 1 (negotiation), Type 2 (challenge) and Type 3 (authentication). For more information, see [MS-NLMP].

presence information: A set of metadata for a client device, including IP address, port, and connection status.

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

public safety answering point (PSAP): A call center that is responsible for answering calls to a telephone number for an emergency service and, in some cases, dispatching that service.

Secure Sockets Layer (SSL): A security protocol that supports confidentiality and integrity of messages in client and server applications that communicate over open networks. SSL uses two keys to encrypt data-a public key known to everyone and a private or secret key known only to the recipient of the message. SSL supports server and, optionally, client **authentication** using X.509 **certificates**. For more information, see [X509]. The SSL protocol is precursor to **Transport Layer Security (TLS)**. The TLS version 1.0 specification is based on SSL version 3.0 [SSL3].

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].

SOAP: A lightweight protocol for exchanging structured information in a decentralized, distributed environment. **SOAP** uses XML technologies to define an extensible messaging framework, which provides a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation-specific semantics. SOAP 1.2 supersedes SOAP 1.1. See [SOAP1.2-1/2003].

SOAP body: A container for the payload data being delivered by a **SOAP message** to its recipient. See [SOAP1.2-1/2007] section 5.3 for more information.

SOAP envelope: A container for **SOAP message** information and the root element of a **SOAP** document. See [SOAP1.2-1/2007] section 5.1 for more information.

SOAP message: An XML document consisting of a mandatory **SOAP envelope**, an optional SOAP header, and a mandatory **SOAP body**. See [SOAP1.2-1/2007] section 5 for more information.

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.

Transport Layer Security (TLS): A security protocol that supports confidentiality and integrity of messages in client and server applications communicating over open networks. **TLS** supports server and, optionally, client authentication by using X.509 certificates (as specified in [X509]). **TLS** is standardized in the IETF TLS working group. See [RFC4346].

type-length-value (TLV): A method of organizing data that involves a Type code (16-bit), a specified length of a Value field (16-bit), and the data in the Value field (variable).

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].

Web Services Description Language (WSDL): An XML format for describing network services as a set of endpoints that operate on messages that contain either document-oriented or procedure-oriented information. The operations and messages are described abstractly and are bound to a concrete network protocol and message format in order to define an endpoint.

Related concrete endpoints are combined into abstract endpoints, which describe a network service. WSDL is extensible, which allows the description of endpoints and their messages regardless of the message formats or network protocols that are used.

WSDL message: An abstract, typed definition of the data that is communicated during a WSDL operation [WSDL]. Also, an element that describes the data being exchanged between web service providers and clients.

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.

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.

[IEEE802.1AB] Congdon, P., Ed. and Lane, B., Ed., "Station and Media Access Control Connectivity Discovery", April 2005, http://www.ieee802.org/1/pages/802.1ab.html

[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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.rfc-editor.org/rfc/rfc2616.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

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

[SOAP1.1] Box, D., Ehnebuske, D., Kakivaya, G., et al., "Simple Object Access Protocol (SOAP) 1.1", May 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[WSA1.0] World Wide Web Consortium, "Web Services Addressing 1.0 - WSDL Binding", W3C Candidate Recommendation, May 2006, http://www.w3.org/TR/2006/CR-ws-addr-wsdl-20060529/

[WSDL] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S., "Web Services Description Language (WSDL) 1.1", W3C Note, March 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[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

[MS-NLMP] Microsoft Corporation, "NT LAN Manager (NTLM) Authentication Protocol".

[MS-OCAUTHWS] Microsoft Corporation, "OC Authentication Web Service Protocol".

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, http://www.rfc-editor.org/rfc/rfc2818.txt

[RFC4559] Jaganathan, K., Zhu, L., and Brezak, J., "SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows", RFC 4559, June 2006, http://www.rfc-editor.org/rfc/rfc4559.txt

1.3 Overview

This protocol is used to retrieve the locations based on network identifiers, or based on city.

This protocol specifies a request that contains the network identifiers for which locations need to be retrieved. The response contains the response status and, if the request is processed successfully, the locations that are most appropriate for the network identifiers specified. This protocol also specifies another request that contains the city, state, and country/region for which locations need to be retrieved. The response contains the response status and, if the request is processed successfully, the locations in that city, state, and country/region.

This protocol is defined as a Web service. This protocol specifies the structure of the schema used to construct the body in the request and response messages. This protocol uses **Simple Object Access Protocol (SOAP)**, as described in [SOAP1.1], and **Web Services Description Language (WSDL)**, as described in [WSDL] to describe the structure of the message body. The full WSDL is included in section 6.

1.4 Relationship to Other Protocols

This protocol uses SOAP over **Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)**, as described in [RFC2818], as shown in the following layering diagram.

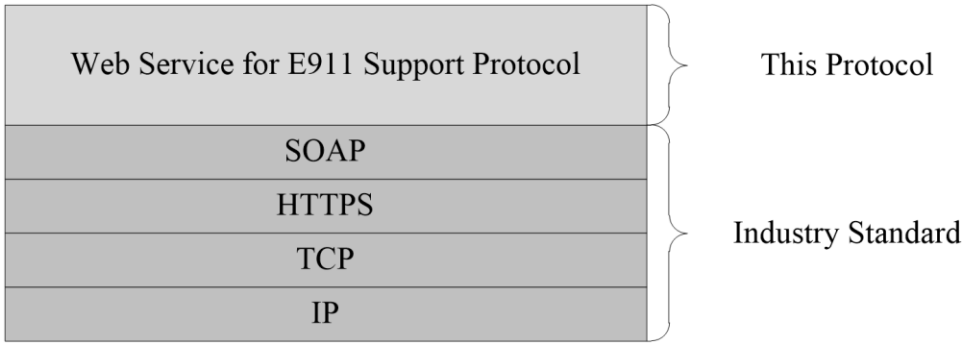

Figure 1: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

For a protocol client that uses this protocol with a protocol server, it is assumed that the protocol server has an operational SOAP1.1/HTTP1.1 /TCP/IP stack. It is also assumed that the protocol client has the **fully qualified domain name (FQDN)** of the protocol server to which the protocol client will connect. The protocol client can obtain the FQDN of the protocol server via a different channel, for example, the **Session Initiation Protocol (SIP)** signaling channel. The protocol server also requires that the protocol client be able to negotiate **Hypertext Transfer Protocol (HTTP)** over **Transport Layer Security (TLS)** to establish the connection.

1.6 Applicability Statement

This protocol is designed so that a client can acquire the location that can be passed on with an E911(Enhanced 911 call with location information in it) call, so that a **public safety answering point (PSAP)** can dispatch emergency help to the correct destination. The locations returned can also be used by the client to publish **presence information**.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

This protocol uses SOAP, as specified in [SOAP1.1], over HTTPS as specified in section 1.4. The protocol uses the security model described in section 5.<1>

2.2 Common Message Syntax

This section contains common definitions that are used by this protocol. The syntax of the definitions uses **XML schema**, as specified in [XMLSCHEMA1] and [XMLSCHEMA2], and WSDL, as specified in [WSDL].

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
soap	http://schemas.xmlsoap.org/wsdl/soap	[SOAP1.1]
xsd	http://www.w3.org/2001/XMLSchema	[XMLSCHEMA1], [XMLSCHEMA2]
http	http://schemas.xmlsoap.org/wsdl/http	[RFC2616]
ca	urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr	[RFC5139]
pidf	urn:ietf:params:xml:ns:pidf	[RFC3863]
wsaw	http://www.w3.org/2006/05/addressing/wsdl	[WSA1.0]
wsdl	http://schemas.xmlsoap.org/wsdl	[WSDL]
tns	urn:schema:Microsoft.Rtc.WebComponent.Lis.2010	

2.2.2 Messages

None.

2.2.3 Elements

The following table summarizes the set of common XML schema element definitions defined by this specification. XML schema element definitions that are specific to a particular operation are described with the operation.

Element	Description
Entity	The **Uniform Resource Identifier (URI)** that the Location Object returned in the **Presence Information Data Format (PIDF)** document references.
ReturnCode	The return code indicating whether the request succeeded or the reason for failure.
presenceList	List of **pidf:presence** elements each containing a **location** object.

.

2.2.3.1 Entity

The **Entity** element contains the Uniform Resource Identifier (URI) that the Location Object returned in the Presence Information Data Format (PIDF) document references. It is expected to be the URI of the user making the Web service request.

The schema for this element is as follows:

```
<xsd:element minOccurs="1" maxOccurs="1" name="Entity" type="tns:restrictedAnyURI" />
```

2.2.3.2 ReturnCode

The **ReturnCode** element contains the return code in the response, indicating whether the request succeeded or the reason for failure.

The schema for this element is as follows:

```
<xsd:element minOccurs="1" maxOccurs="1" name="ReturnCode" type="tns:ReturnCodeType" />
```

The **ReturnCodeType** type and each of the values it can take are described in section 2.2.5.1.

2.2.3.3 presenceList

The **presenceList** element contains the list of **pidf:presence** elements in the response, each of which contains a **location** object.

The schema for this element is as follows:

```
<xsd:element minOccurs="0" maxOccurs="1" name="presenceList" type="tns:presenceListType" />
```

The **presenceListType** type is described in section 2.2.4.1.

2.2.4 Complex Types

The following table summarizes the set of common XML schema complex type definitions defined by this specification. XML schema complex type definitions that are specific to a particular operation are described with the operation.

Complex type	Description
presenceListType	List of **pidf:presence** elements.

2.2.4.1 presenceListType

The **presenceListType** complex type contains a list of **pidf:presence** elements. The **pidf:presence** element, as defined by PIDF [RFC3863], has a GEOPRIV Location Object, as defined 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 defined in [RFC5139]. The client ignores all except the following elements returned in the Civic Address: **country/region**, **A1**, **A3**, **PRD**, **RD**, **STS**, **POD**, **HNO**, **HNS**, **LOC**, **NAM**, **PC**. If the address cannot be trusted to match the network identifiers specified in the location request, the **method** element embedded in the **geopriv** element MUST have the value "Manual".

The schema for this type is as follows:

```
<xsd:complexType name="presenceListType">
    <xsd:sequence>
        <xsd:element minOccurs="0" maxOccurs="unbounded" ref="pidf:presence" />
    </xsd:sequence>
</xsd:complexType>
```

2.2.5 Simple Types

The following table summarizes the set of common XML schema simple type definitions defined by this specification. XML schema simple type definitions that are specific to a particular operation are described with the operation.

Simple type	Description
ReturnCodeType	The return code indicating whether the request succeeded or the reason for failure.
restrictedAnyURI	**s:anyURI** but bounded to length between 1 and 64.

2.2.5.1 ReturnCodeType

The **ReturnCodeType** simple type contains the return code indicating whether the request succeeded or the reason for failure. The return code can be one of the following four values:

- 200=Success
- 400=Bad Request
- 404=Not Found
- 500=Internal Server Error

The schema for this type is as follows:

```
<xsd:simpleType name="ReturnCodeType">
    <xsd:annotation>
        <xsd:documentation>200=Success; 400=Bad Request; 404=Not Found; 500=Internal Server
Error;</xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="200" />
        <xsd:enumeration value="400" />
        <xsd:enumeration value="404" />
        <xsd:enumeration value="500" />
    </xsd:restriction>
</xsd:simpleType>
```

2.2.5.2 restrictedAnyURI

The **restrictedAnyURI** simple type is the same as the **xsd:anyURI** simple type, except that the length of **restrictedAnyURI** is restricted to a value from 1 through 64<2>.

The schema for this type is as follows:

```
<xsd:simpleType name="restrictedAnyURI">
    <xsd:annotation>
        <xsd:documentation>anyURI but bounded to length between 1 and
64.</xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:anyURI">
        <xsd:minLength value="1" />
        <xsd:maxLength value="64" />
    </xsd:restriction>
</xsd:simpleType>
```

2.2.6 Attributes

This specification does not define any common XML schema attribute definitions.

2.2.7 Groups

This specification does not define any common XML schema group definitions.

2.2.8 Attribute Groups

This specification does not define any common XML schema attribute group definitions.

2.2.9 Common Data Structures

This specification does not define any common XML schema data structures.

3 Protocol Details

The client side of this protocol is simply a pass-through. That is, no additional timers or other state is required on the client side of this protocol. Calls made by the higher-layer protocol or application are passed directly to the transport, and the results returned by the transport are passed directly back to the higher-layer protocol or application.

3.1 Server Details

The Location Information Service listens on a port type called ILIService. The interface exposes two operations called **GetLocations** and **GetLocationsInCity**.

3.1.1 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.

The Location Information Service does not retain any state between requests, but conceptually has access to configuration that maps network identifiers such as the following:

- Wireless Access Point Basic Service Set Identifier—mapped as WAPBSSID
- Received Signal Strength Indication—mapped as RSSI
- Media Access Control Address—mapped as MAC
- Chassis—mapped as ChassisID, which is a **binary large object (BLOB)** representing the ChassisID **type-length-value (TLV)**, as defined by [IEEE802.1AB]
- Port—mapped as PortID, which is a BLOB representing the PortID type-length-value (TLV), as defined by [IEEE802.1AB]
- Subnet—mapped as SubnetID
- Internet Protocol Address to Locations—mapped as IP

3.1.2 Timers

None.

3.1.3 Initialization

As part of initialization, the server MUST start listening for incoming requests on an HTTPS **Uniform Resource Locator (URL)**. The client MUST have access to this HTTPS URL and can obtain the URL by a channel that is separate from the HTTPS channel used for retrieving locations, for example, through SIP.

3.1.4 Message Processing Events and Sequencing Rules

The following table summarizes the list of WSDL operations as defined by this specification*:*

Operation	Description
GetLocations	This operation retrieves the most appropriate locations that match the network identifiers specified in a location request.
GetLocationsInCity	This operation retrieves the locations in city, state, and country/region specified in the request.

3.1.4.1 GetLocations

The following excerpt from this protocol's WSDL specifies the messages that constitute this operation.

```
<wsdl:operation name="GetLocations">
  <wsdl:input wsaw:Action="LIService/GetLocations" name="GetLocationsRequest"
message="tns:GetLocationsRequest" />
  <wsdl:output wsaw:Action="LIService/GetLocationsResponse" name="GetLocationsResponse"
message="tns:GetLocationsResponse" />
</wsdl:operation>
```

When a client needs to request a location, a **Transmission Control Protocol (TCP)** connection MUST be made to the server and **Secure Sockets Layer (SSL)** MUST be negotiated. The address of the server that makes the TCP connection can be obtained through a different channel, such as SIP. After successful SSL negotiation, a SOAP HTTP request, **GetLocationsRequest** message, MUST be constructed with a **SOAP body** containing the **GetLocationsRequest** element.

On receiving a **GetLocationsRequest** request, the server queries its repository of locations to get all of the locations that match the network identifiers specified in the request. The order in which these locations are looked up by using the network identifiers is WAPBSSID, ChassisID+PortID, ChassisID, SubnetID, MAC. This order, however, is implementation-specific and depends on what is considered the most-appropriate location match by the server. After obtaining a location match, the server MUST construct the **GetLocationsResponse** message, containing the **GetLocationsResponse** element, and it MUST send the message in the SOAP HTTP response, which is a 2xx response to a SOAP HTTP request. In case of errors, the **GetLocationsResponse** message MUST specify the error that was encountered by the server, and the error MUST be sent in the 2xx SOAP HTTP response. The **GetLocationsResponse** message is specified in section 3.1.4.1.1.2. Errors in protocols and in the security model described in section 2.1 and section 1.4 should be handled per their own specifications.

3.1.4.1.1 Messages

The following table summarizes the set of WSDL message definitions that are specific to this operation.

Message	Definition
GetLocationsRequest	A request from a client to retrieve the locations of the **endpoint**.
GetLocationsResponse	The response from a server after it executes a request to retrieve the locations of an endpoint (5).

3.1.4.1.1.1 GetLocationsRequest

The **GetLocationsRequest** **SOAP message** is a request that is sent from the client to retrieve the locations of the endpoint. This message can be sent just after login and whenever the client endpoint connects to another wireless access point, but it is implementation-specific. The request information MUST be captured in the **GetLocationsRequest** element in the SOAP body of the message. The **GetLocationsRequest** element is specified in section 3.1.4.1.2.1.<3>

3.1.4.1.1.2 GetLocationsResponse

The **GetLocationsResponse** SOAP message is a response that is sent by the server after it executes a request to retrieve the locations of the endpoint. This message contains the locations that match the network identifiers specified in the request. The result is represented in the **GetLocationsResponse** element, which MUST be in the SOAP body of the SOAP message. The **GetLocationsResponse** element is specified in section 3.1.4.1.2.2. If the server is able to successfully match any locations for the network identifiers, the response element contains the locations matched and a **ReturnCodeType** indicating success. These locations are included in the complex type **presenceListType**, which is specified in section 2.2.4. In case of an error, the response element MUST specify the reason for the failure to retrieve locations in the simple type **ReturnCodeType**, which is specified in section 2.2.5. Because no location matches occur in the case of an error, the **presenceListType** complex type will not be present in such a case.

3.1.4.1.2 Elements

The following table summarizes the XML schema element definitions that are specific to this operation.

Element	Definition
GetLocationsRequest	Container in the request to retrieve the locations for network identifiers.
GetLocationsResponse	Container in the response to a request to retrieve the locations for network identifiers.

3.1.4.1.2.1 GetLocationsRequest

The **GetLocationsRequest** element is the overall container of the information that is sent in the SOAP request to retrieve locations for network identifiers. The schema of the request body within the **SOAP envelope** is as follows:<4>

```
<xsd:element name="GetLocationsRequest">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element minOccurs="1" maxOccurs="1" name="Entity"
type="tns:restrictedAnyURI" />
            <xsd:element minOccurs="0" maxOccurs="1" name="WAPBSSID"
type="tns:EnetMacAddressType" />
            <xsd:element minOccurs="0" maxOccurs="1" name="RSSI" type="xsd:unsignedByte"
/>
            <xsd:element minOccurs="0" maxOccurs="1" name="MAC"
type="tns:EnetMacAddressType" />
            <xsd:element minOccurs="0" maxOccurs="1" name="ChassisID"
type="tns:LLDPChassisIDOrPortIDTLVType" />
            <xsd:element minOccurs="0" maxOccurs="1" name="PortID"
type="tns:LLDPChassisIDOrPortIDTLVType" />
            <xsd:element minOccurs="0" maxOccurs="1" name="SubnetID" type="tns:IPAddress"
/>
            <xsd:element minOccurs="0" maxOccurs="1" name="IP" type="tns:IPAddress" />
        </xsd:sequence>
    </xsd:complexType>
```

```
</xsd:element>
```

3.1.4.1.2.2 GetLocationsResponse

The **GetLocationsResponse** element is the overall container in the response to the **GetLocationsRequest** request. **GetLocationsResponse** encapsulates the results of the operation to retrieve locations for network identifiers. It contains an optional **presenceList** element of type **presenceListType**, and one **ReturnCode** element of type **ReturnCodeType**. The schema for this element within the SOAP envelope is as follows.

```
<xsd:element name="GetLocationsResponse">
  <xsd:complexType>
     <xsd:sequence>
        <xsd:element minOccurs="1" maxOccurs="1" name="ReturnCode"
type="tns:ReturnCodeType" />
        <xsd:element minOccurs="0" maxOccurs="1" name="presenceList"
type="tns:presenceListType" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

3.1.4.1.3 Complex Types

None.

3.1.4.1.4 Simple Types

None.

3.1.4.1.5 Attributes

None.

3.1.4.1.6 Groups

None.

3.1.4.1.7 Attribute Groups

None.

3.1.4.2 GetLocationsInCity

The following excerpt from the WSDL for this protocol specifies the messages that constitute this operation.

```
<wsdl:operation name="GetLocationsInCity">
  <wsdl:input wsaw:Action="LIService/GetLocationsInCity" name="GetLocationsInCityRequest"
message="tns:GetLocationsInCityRequest" />
  <wsdl:output wsaw:Action="LIService/GetLocationsInCityResponse"
name="GetLocationsInCityResponse" message="tns:GetLocationsInCityResponse" />
</wsdl:operation>
```

When a client needs to request a location, a TCP connection MUST be made to the server and SSL MUST be negotiated. The address of the server that makes the TCP connection can be obtained through a different channel, such as SIP. After successful SSL negotiation, a SOAP HTTP request,

GetLocationsInCityRequest message, MUST be constructed with a SOAP body containing the **GetLocationsInCityRequest** element.

On receiving a **GetLocationsInCityRequest** request, the server queries its repository of locations to get all of the locations that match the city, state, and country/region specified in the request. After obtaining the locations, the server MUST construct the **GetLocationsInCityResponse** message, containing the **GetLocationsInCityResponse** element, and it MUST send the message in the SOAP HTTP response, which is a 2xx response to a SOAP HTTP request. In case of errors, the **GetLocationsInCityResponse** message MUST specify the error that was encountered by the server, and the error MUST be sent in the 2xx SOAP HTTP response. The **GetLocationsInCityResponse** message is specified in section 3.1.4.2.1.2. Errors in protocols and in the security model described in section 2.1 and section 1.4 should be handled per their own specifications.

3.1.4.2.1 Messages

The following **WSDL message** definitions are specific to this operation.

Message	Definition
GetLocationsInCityRequest	A request from a client to retrieve the locations in a city.
GetLocationsInCityResponse	The response from a server after it executes a request to retrieve the locations in a city.

3.1.4.2.1.1 GetLocationsInCityRequest

The **GetLocationsInCityRequest** SOAP message is a request that is sent from the client, as a result of a user action to retrieve locations in a city. The request information MUST be captured in the **GetLocationsInCityRequest** element in the SOAP body of the message. The **GetLocationsInCityRequest** element is specified in section 3.1.4.2.2.1.

3.1.4.2.1.2 GetLocationsInCityResponse

The **GetLocationsInCityResponse** SOAP message is a response that is sent by the server after it executes a request to retrieve locations in a city. This message contains the result of matching locations in the repository with the city, state, and country/region specified in the request. The result is represented in the **GetLocationsInCityResponse** element, which MUST be in the SOAP body of the SOAP message. The **GetLocationsInCityResponse** element is specified in section 3.1.4.2.2.2. If the server is able to successfully match any locations for the city, state and country/region, the **response** element contains the locations matched and a **ReturnCodeType** indicating success. These locations are included in the complex type **presenceListType**, which is specified in section 2.2.4. In case of an error, the response element MUST specify the reason for the failure to retrieve locations in the simple type **ReturnCodeType**, which is specified in section 2.2.5. Because no location matches occur in the case of an error, the **presenceListType** complex type will not be present in such a case.

3.1.4.2.2 Elements

The following table summarizes the XML schema element definitions that are specific to this operation.

Element	Definition
GetLocationsInCityRequest	Container of the information in a request to retrieve locations in a city, state, and country/region.
GetLocationsInCityResponse	Container of the information in a response to a request

Element	Definition
	to retrieve locations in a city, state, and country/region.

3.1.4.2.2.1 GetLocationsInCityRequest

The **GetLocationsInCityRequest** element is the overall container of the information that is sent in the SOAP request to retrieve locations for city, state, and country/region. The schema of the request body within the SOAP envelope is as follows.

```
<xsd:element name="GetLocationsInCityRequest">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element minOccurs="1" maxOccurs="1" name="Entity"
type="tns:restrictedAnyURI" />
            <xsd:element minOccurs="1" maxOccurs="1" name="Country" type="ca:iso3166a2"
/>
            <xsd:element minOccurs="1" maxOccurs="1" name="State" type="tns:StateType" />
            <xsd:element minOccurs="1" maxOccurs="1" name="City" type="tns:CityType" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

3.1.4.2.2.2 GetLocationsInCityResponse

The **GetLocationsInCityResponse** element is the overall container in the response to the **GetLocationsInCityRequest** request. **GetLocationsInCityResponse** encapsulates the results of the operation to retrieve locations for city, state, and country/region. It contains an optional **presenceList** element of type **presenceListType**, and one **ReturnCode** element of type **ReturnCodeType**. The schema for this element within the SOAP envelope is as follows.

```
<xsd:element name="GetLocationsInCityResponse">
  <xsd:complexType>
     <xsd:sequence>
        <xsd:element minOccurs="1" maxOccurs="1" name="ReturnCode" type="tns:ReturnCodeType"
/>
        <xsd:element minOccurs="0" maxOccurs="1" name="presenceList"
type="tns:presenceListType" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

3.1.4.2.3 Complex Types

None.

3.1.4.2.4 Simple Types

The following table summarizes the XML schema simple type definitions that are specific to this operation.

Simple type	Description
CityType	A string of length between 1 and 64 that represents the **City**.

Simple type	Description
StateType	A string of length 2 that represents the **State**.

3.1.4.2.4.1 CityType

The **CityType** simple type is a string that has a length of from 1 through 64 and that represents the city.

The schema is as follows:

```
<xsd:simpleType name="CityType">
    <xsd:annotation>
        <xsd:documentation>any string of length between 1 and 64.</xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="1" />
        <xsd:maxLength value="64" />
    </xsd:restriction>
</xsd:simpleType>
```

3.1.4.2.4.2 StateType

The **StateType** simple type is a string of length 2 that represents the state.

The schema is as follows:

```
<xsd:simpleType name="StateType">
    <xsd:annotation>
        <xsd:documentation>any string of length 2.</xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="2" />
        <xsd:maxLength value="2" />
    </xsd:restriction>
```

</xsd:simpleType>

3.1.4.2.5 Attributes

None.

3.1.4.2.6 Groups

None.

3.1.4.2.7 Attribute Groups

None.

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

4 Protocol Examples

To retrieve the location for the network identifiers of a client, the protocol client constructs the following WSDL message.

```
<soap:Body>
  <GetLocationsRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <Entity>sip:voip_911_user1@contoscovdomain.com</Entity>
    <!-- <WAPBSSID>string</WAPBSSID> -->
    <RSSI>0</RSSI>
    <MAC>12-22-22-22-22-22</MAC>
    <!-- <ChassisID>base64Binary</ChassisID> -->
    <!-- <PortID>base64Binary</PortID> -->
    <SubnetID>192.168.0.0</SubnetID>
    <IP>192.168.0.244</IP>
  </GetLocationsRequest>
</soap:Body>
```

The protocol server then responds with the following.<5>

```
<soap:Body>
  <GetLocationsResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <ReturnCode>200</ReturnCode>
    <presenceList>
      <presence entity="sip:voip_911_user1@contoscovdomain.com"
xmlns="urn:ietf:params:xml:ns:pidf">
        <tuple id="_LIS:0">
          <status>
            <geopriv xmlns="urn:ietf:params:xml:ns:pidf:geopriv10">
              <location-info>
                <civicAddress xmlns="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr">
                  <country>US</country>
                  <A1>WA</A1>
                  <A3>Redmond</A3>
                  <PRD />
                  <RD>163rd</RD>
                  <STS>Ave</STS>
                  <POD>NE</POD>
                  <HNO>3910</HNO>
                  <HNS />
                  <LOC>30/3351</LOC>
                  <NAM>Microsoft</NAM>
                  <PC>98052</PC>
                </civicAddress>
              </location-info>
            </geopriv>
          </status>
        </tuple>
      </presence>
    </presenceList>
  </GetLocationsResponse>
</soap:Body>
```

To retrieve the locations for the city, state, and country/region specified by a client, the protocol client constructs the following WSDL message.

```
<soap:Body>
  <GetLocationsInCityRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <Entity>sip:voip 911 user1@vcontoscodomain.com</Entity>
    <Country>US</Country>
    <State>WA</State>
    <City>San Francisco</City>
```

```
  </GetLocationsInCityRequest>
</soap:Body>
```

The protocol server then responds with the following.

```
<soap:Body>
  <GetLocationsInCityResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <ReturnCode>404</ReturnCode>
  </GetLocationsInCityResponse>
</soap:Body>
```

5 Security

5.1 Security Considerations for Implementers

This protocol allows HTTP connections only over SSL. Users are authenticated using **Kerberos** v5 and **NT LAN Manager (NTLM) Authentication Protocol authentication** methods. NTLM is described in [MS-NLMP]. Clients can also be authenticated using the SPNEGO-based Kerberos and NTLM HTTP authentication, as described in [RFC4559]. Clients can also be authenticated using custom **certificate**-based authentication, as described in [MS-OCAUTHWS].

5.2 Index of Security Parameters

None.

6 Appendix A: Full WSDL

For ease of implementation, the full WSDL and schema are provided in this appendix.

```
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions name="LIService"
targetNamespace="urn:schema:Microsoft.Rtc.WebComponent.Lis.2010"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:tns="urn:schema:Microsoft.Rtc.WebComponent.Lis.2010"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <wsdl:types>
        <xsd:schema>
          <xsd:import namespace= "urn:schema:Microsoft.Rtc.WebComponent.Lis.2010"
schemaLocation="LIService.xsd" />
        </xsd:schema>
    </wsdl:types>
    <wsdl:message name="GetLocationsRequest">
        <wsdl:part name="parameters" element="tns:GetLocationsRequest" />
    </wsdl:message>
    <wsdl:message name="GetLocationsResponse">
        <wsdl:part name="parameters" element="tns:GetLocationsResponse" />
    </wsdl:message>
    <wsdl:message name="GetLocationsInCityRequest">
        <wsdl:part name="parameters" element="tns:GetLocationsInCityRequest" />
    </wsdl:message>
    <wsdl:message name="GetLocationsInCityResponse">
        <wsdl:part name="parameters" element="tns:GetLocationsInCityResponse" />
    </wsdl:message>
    <wsdl:portType name="ILIService">
        <wsdl:operation name="GetLocations">
            <wsdl:input wsaw:Action="LIService/GetLocations" name="GetLocationsRequest"
message="tns:GetLocationsRequest" />
            <wsdl:output wsaw:Action="LIService/GetLocationsResponse"
name="GetLocationsResponse" message="tns:GetLocationsResponse" />
        </wsdl:operation>
        <wsdl:operation name="GetLocationsInCity">
            <wsdl:input wsaw:Action="LIService/GetLocationsInCity"
name="GetLocationsInCityRequest" message="tns:GetLocationsInCityRequest" />
            <wsdl:output wsaw:Action="LIService/GetLocationsInCityResponse"
name="GetLocationsInCityResponse" message="tns:GetLocationsInCityResponse" />
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="LIServiceSoap" type="tns:ILIService">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
        <wsdl:operation name="GetLocations">
            <soap:operation soapAction="LIService/GetLocations" style="document" />
            <wsdl:input name="GetLocationsRequest">
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output name="GetLocationsResponse">
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="GetLocationsInCity">
            <soap:operation soapAction="LIService/GetLocationsInCity" style="document" />
            <wsdl:input name="GetLocationsInCityRequest">
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output name="GetLocationsInCityResponse">
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
</wsdl:definitions>
```

LIService.xsd referenced in the preceding WSDL is as follows.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ca="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"
xmlns:pidf="urn:ietf:params:xml:ns:pidf"
xmlns="urn:schema:Microsoft.Rtc.WebComponent.Lis.2010"
xmlns:tns="urn:schema:Microsoft.Rtc.WebComponent.Lis.2010"
targetNamespace="urn:schema:Microsoft.Rtc.WebComponent.Lis.2010"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.1">
    <xsd:import schemaLocation="CivicAddress.rfc5139.xsd"
namespace="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr" />
    <xsd:import schemaLocation="Pidf LO.rfc3863.xsd" namespace="urn:ietf:params:xml:ns:pidf"
/>
    <xsd:element name="GetLocationsRequest">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element minOccurs="1" maxOccurs="1" name="Entity"
type="tns:restrictedAnyURI" />
                <xsd:element minOccurs="0" maxOccurs="1" name="WAPBSSID"
type="tns:EnetMacAddressType" />
                <xsd:element minOccurs="0" maxOccurs="1" name="RSSI" type="xsd:unsignedByte"
/>
                <xsd:element minOccurs="0" maxOccurs="1" name="MAC"
type="tns:EnetMacAddressType" />
                <xsd:element minOccurs="0" maxOccurs="1" name="ChassisID"
type="tns:LLDPChassisIDOrPortIDTLVType" />
                <xsd:element minOccurs="0" maxOccurs="1" name="PortID"
type="tns:LLDPChassisIDOrPortIDTLVType" />
                <xsd:element minOccurs="0" maxOccurs="1" name="SubnetID" type="tns:IPAddress"
/>
                <xsd:element minOccurs="0" maxOccurs="1" name="IP" type="tns:IPAddress" />
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="GetLocationsResponse">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element minOccurs="1" maxOccurs="1" name="ReturnCode"
type="tns:ReturnCodeType" />
                <xsd:element minOccurs="0" maxOccurs="1" name="presenceList"
type="tns:presenceListType" />
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="GetLocationsInCityRequest">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element minOccurs="1" maxOccurs="1" name="Entity"
type="tns:restrictedAnyURI" />
                <xsd:element minOccurs="1" maxOccurs="1" name="Country" type="ca:iso3166a2"
/>
                <xsd:element minOccurs="1" maxOccurs="1" name="State" type="tns:StateType" />
                <xsd:element minOccurs="1" maxOccurs="1" name="City" type="tns:CityType" />
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="GetLocationsInCityResponse">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element minOccurs="1" maxOccurs="1" name="ReturnCode"
type="tns:ReturnCodeType" />
                <xsd:element minOccurs="0" maxOccurs="1" name="presenceList"
type="tns:presenceListType" />
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:simpleType name="restrictedAnyURI">
        <xsd:annotation>
```

```
            <xsd:documentation>anyURI but bounded to length between 1 and
64.</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:anyURI">
            <xsd:minLength value="1" />
            <xsd:maxLength value="64" />
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="StateType">
        <xsd:annotation>
            <xsd:documentation>any string of length 2.</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:minLength value="2" />
            <xsd:maxLength value="2" />
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="CityType">
        <xsd:annotation>
            <xsd:documentation>any string of length between 1 and 64.</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:minLength value="1" />
            <xsd:maxLength value="64" />
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="EnetMacAddressType">
        <xsd:annotation>
            <xsd:documentation>an Ethernet MAC address in IEEE 802 standard format human-
readable form. http://en.wikipedia.org/wiki/MAC_address</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="(([a-fA-F0-9]{1,2}-){5})([a-fA-F0-9]{1,2})" />
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="IPAddress">
        <xsd:annotation>
            <xsd:documentation>an IP (v4 or v6) address.</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:minLength value="0" />
            <xsd:maxLength value="39" />
            <xsd:pattern value="((25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9][0-9]|[0-
9])\.){3}(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9][0-9]|[0-9])" />
            <xsd:pattern value="(([0-9a-fA-F]{1,4}:){7})([0-9a-fA-F]{1,4})" />
            <xsd:pattern value="(([0-9a-fA-F]{1,4}:){6})(([0-9]{1,3}\.[0-9]{1,3}\.[0-
9]{1,3}\.[0-9]{1,3}))" />
            <xsd:pattern value="(((?>[0-9a-fA-F]{1,4}):)*((?>[0-9a-fA-F]{1,4})))*(::)(((?>[0-
9a-fA-F]{1,4}):)*((?>[0-9a-fA-F]{1,4})))*" />
            <xsd:pattern value="(((?>[0-9a-fA-F]{1,4}):)*((?>[0-9a-fA-F]{1,4})))*(::)(((?>[0-
9a-fA-F]{1,4}):)*((?>[0-9a-fA-F]{1,4})))*(([0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}))"
/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="LLDPChassisIDOrPortIDTLVType">
        <xsd:annotation>
            <xsd:documentation>a Link Layer Discovery Protocol TLV.</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:base64Binary">
            <xsd:minLength value="0" />
            <xsd:maxLength value="258" />
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="presenceListType">
        <xsd:sequence>
            <xsd:element minOccurs="0" maxOccurs="unbounded" ref="pidf:presence" />
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="ReturnCodeType">
```

```
        <xsd:annotation>
            <xsd:documentation>200=Success; 400=Bad Request; 404=Not Found; 500=Internal
Server Error;</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="200" />
            <xsd:enumeration value="400" />
            <xsd:enumeration value="404" />
            <xsd:enumeration value="500" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

CivicAddress.rfc5139..xsd referenced in the **LIService.xsd** is as follows:

```
<?xml version="1.0"?>
   <xsd:schema
     targetNamespace="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:ca="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     elementFormDefault="qualified" attributeFormDefault="unqualified">

  <xsd:import namespace="http://www.w3.org/XML/1998/namespace"
             schemaLocation="http://www.w3.org/2001/xml.xsd"/>

  <xsd:simpleType name="iso3166a2">
    <xsd:restriction base="xsd:token">
      <xsd:pattern value="[A-Z]{2}"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:complexType name="caType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:token">
        <xsd:attribute ref="xml:lang" use="optional"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:element name="civicAddress" type="ca:civicAddress"/>
  <xsd:complexType name="civicAddress">
    <xsd:sequence>
      <xsd:element name="country" type="ca:iso3166a2" minOccurs="0"/>
      <xsd:element name="A1" type="ca:caType" minOccurs="0"/>
      <xsd:element name="A2" type="ca:caType" minOccurs="0"/>
      <xsd:element name="A3" type="ca:caType" minOccurs="0"/>
      <xsd:element name="A4" type="ca:caType" minOccurs="0"/>
      <xsd:element name="A5" type="ca:caType" minOccurs="0"/>
      <xsd:element name="A6" type="ca:caType" minOccurs="0"/>
      <xsd:element name="PRM" type="ca:caType" minOccurs="0"/>
      <xsd:element name="PRD" type="ca:caType" minOccurs="0"/>
      <xsd:element name="RD" type="ca:caType" minOccurs="0"/>
      <xsd:element name="STS" type="ca:caType" minOccurs="0"/>
      <xsd:element name="POD" type="ca:caType" minOccurs="0"/>
      <xsd:element name="POM" type="ca:caType" minOccurs="0"/>
      <xsd:element name="RDSEC" type="ca:caType" minOccurs="0"/>
      <xsd:element name="RDBR" type="ca:caType" minOccurs="0"/>
      <xsd:element name="RDSUBBR" type="ca:caType" minOccurs="0"/>
      <xsd:element name="HNO" type="ca:caType" minOccurs="0"/>
      <xsd:element name="HNS" type="ca:caType" minOccurs="0"/>
      <xsd:element name="LMK" type="ca:caType" minOccurs="0"/>
      <xsd:element name="LOC" type="ca:caType" minOccurs="0"/>
      <xsd:element name="FLR" type="ca:caType" minOccurs="0"/>
      <xsd:element name="NAM" type="ca:caType" minOccurs="0"/>
      <xsd:element name="PC" type="ca:caType" minOccurs="0"/>
      <xsd:element name="BLD" type="ca:caType" minOccurs="0"/>
```

```
      <xsd:element name="UNIT" type="ca:caType" minOccurs="0"/>
      <xsd:element name="ROOM" type="ca:caType" minOccurs="0"/>
      <xsd:element name="SEAT" type="ca:caType" minOccurs="0"/>
      <xsd:element name="PLC" type="xsd:token" minOccurs="0"/>
      <xsd:element name="PCN" type="ca:caType" minOccurs="0"/>
      <xsd:element name="POBOX" type="ca:caType" minOccurs="0"/>
      <xsd:element name="ADDCODE" type="ca:caType" minOccurs="0"/>
      <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:anyAttribute namespace="##any" processContents="lax"/>
  </xsd:complexType>
</xsd:schema>
```

Pidf_LO.rfc3863.xsd referenced in the **LIService.xsd** is as follows<6>:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Liquid XML Studio 0.9.7.0 (http://www.liquid-technologies.com) -->
<xsd:schema xmlns:tns="urn:ietf:params:xml:ns:pidf" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:ietf:params:xml:ns:pidf"
xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
xmlns:ms="urn:schema:Rtc.LIS.msftE911PidfExtn.2008"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:import schemaLocation="http://www.w3.org/2001/xml.xsd"
namespace="http://www.w3.org/XML/1998/namespace" />
  <xsd:import schemaLocation="Geopriv CA.rfc4119.xsd"
namespace="urn:ietf:params:xml:ns:pidf:geopriv10" />
  <xsd:import schemaLocation="MsftE911PidfExtn.2008.xsd"
namespace="urn:schema:Rtc.LIS.msftE911PidfExtn.2008" />
  <xsd:element name="presence" type="tns:presence" />
  <xsd:complexType name="presence">
    <xsd:sequence>
      <xsd:element minOccurs="0" maxOccurs="unbounded" name="tuple" type="tns:tuple" />
      <xsd:element minOccurs="0" maxOccurs="unbounded" name="note" type="tns:note" />
      <xsd:any minOccurs="0" maxOccurs="unbounded" namespace="##other"
processContents="lax" />
    </xsd:sequence>
    <xsd:attribute name="entity" type="xsd:anyURI" use="required" />
  </xsd:complexType>
  <xsd:complexType name="tuple">
    <xsd:sequence>
      <xsd:element name="status" type="tns:status" />
      <xsd:any minOccurs="0" maxOccurs="unbounded" namespace="##other"
processContents="lax" />
      <xsd:element minOccurs="0" name="contact" type="tns:contact" />
      <xsd:element minOccurs="0" maxOccurs="unbounded" name="note" type="tns:note" />
      <xsd:element minOccurs="0" name="timestamp" type="xsd:dateTime" />
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID" use="required" />
  </xsd:complexType>
  <xsd:complexType name="status">
    <xsd:sequence>
      <xsd:element minOccurs="0" name="basic" type="tns:basic" />
      <!--We pass out the geopriv and msftE911PidfExtn elements as part of PIDF-LO. Added
these explicitly to the schema for ease of use of generated code. We could add more in
the future, so partners should continue to reference the original rfc3863 schema that
allows extensibility-->
      <xsd:element minOccurs="0" ref="gp:geopriv" />
      <xsd:element minOccurs="0" ref="ms:msftE911PidfExtn" />
      <!--xsd:any minOccurs="0" maxOccurs="unbounded" namespace="##other"
processContents="lax" /-->
    </xsd:sequence>
  </xsd:complexType>
  <xsd:simpleType name="basic">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="open" />
      <xsd:enumeration value="closed" />
```

```
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:complexType name="contact">
    <xsd:simpleContent>
      <xsd:extension base="xsd:anyURI">
        <xsd:attribute name="priority" type="tns:qvalue" />
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:complexType name="note">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute ref="xml:lang" />
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:simpleType name="qvalue">
    <xsd:restriction base="xsd:decimal">
      <xsd:pattern value="0(.[0-9]{0,3})?" />
      <xsd:pattern value="1(.0{0,3})?" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:attribute default="0" name="mustUnderstand" type="xsd:boolean">
    <xsd:annotation>
      <xsd:documentation>
         This attribute may be used on any element within an optional
         PIDF extension to indicate that the corresponding element must
         be understood by the PIDF processor if the enclosing optional
         element is to be handled.
         </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
</xsd:schema>
```

MsftE911PidfExtn.2008.xsd referenced in the **Pidf_LO.rfc3863.xsd** is as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<!--Created with Liquid XML Studio 6.1.18.0 (http://www.liquid-technologies.com)-->
<xsd:schema xmlns:tns="urn:schema:Rtc.LIS.msftE911PidfExtn.2008"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="urn:schema:Rtc.LIS.msftE911PidfExtn.2008"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="msftE911PidfExtn" type="tns:msftE911PidfExtn" />
  <xsd:complexType name="msftE911PidfExtn">
    <xsd:sequence>
      <xsd:element minOccurs="1" maxOccurs="1" name="ConferenceUri" type="xsd:anyURI" />
      <xsd:element minOccurs="1" maxOccurs="1" name="ConferenceMode"
type="tns:ConferenceModeEnum" />
      <xsd:any minOccurs="0" maxOccurs="unbounded" namespace="##other"
processContents="lax" />
    </xsd:sequence>
    <xsd:anyAttribute namespace="##any" />
  </xsd:complexType>
  <xsd:simpleType name="ConferenceModeEnum">
    <xsd:restriction base="xsd:token">
      <xsd:enumeration value="oneway" />
      <xsd:enumeration value="twoway" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

Geopriv_CA.rfc4119.xsd referenced in the **Pidf_LO.rfc3863.xsd** is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Liquid XML Studio 0.9.7.0 (http://www.liquid-technologies.com) -->
```

```
<xsd:schema xmlns:gbp="urn:ietf:params:xml:ns:pidf:geopriv10:basicPolicy"
xmlns:tns="urn:ietf:params:xml:ns:pidf:geopriv10" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="urn:ietf:params:xml:ns:pidf:geopriv10"
xmlns:ca="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xsd:import schemaLocation="BasicGeoprivPolicyTypes.rfc4119.xsd"
namespace="urn:ietf:params:xml:ns:pidf:geopriv10:basicPolicy" />
  <xsd:import schemaLocation="CivicAddress.rfc5139.xsd"
namespace="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr" />
  <xsd:import schemaLocation="http://www.w3.org/2001/xml.xsd"
namespace="http://www.w3.org/XML/1998/namespace" />
  <xsd:element name="geopriv" type="tns:geopriv" />
  <xsd:complexType name="geopriv">
    <xsd:sequence>
      <xsd:element minOccurs="1" maxOccurs="1" name="location-info"
type="tns:locInfoType" />
      <xsd:element minOccurs="1" maxOccurs="1" name="usage-rules"
type="gbp:locPolicyType" />
      <xsd:element minOccurs="0" maxOccurs="1" name="method" type="tns:locMethod" />
      <xsd:element minOccurs="0" maxOccurs="1" name="provided-by"
type="tns:locProvidedBy" />
      <xsd:any minOccurs="0" maxOccurs="unbounded" namespace="##other"
processContents="lax" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="locInfoType">
    <xsd:sequence>
      <xsd:element minOccurs="0" maxOccurs="1" ref="ca:civicAddress" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="locMethod">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute ref="xml:lang" />
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:complexType name="locProvidedBy">
    <xsd:sequence>
      <xsd:any minOccurs="1" maxOccurs="unbounded" namespace="##other"
processContents="skip" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

BasicGeoprivPolicyTypes.rfc4119.xsd referenced in the **Geopriv_CA.rfc4119.xsd** is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Liquid XML Studio 0.9.7.0 (http://www.liquid-technologies.com) -->
<xsd:schema xmlns:tns="urn:ietf:params:xml:ns:pidf:geopriv10:basicPolicy"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="urn:ietf:params:xml:ns:pidf:geopriv10:basicPolicy"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:import schemaLocation="http://www.w3.org/2001/xml.xsd"
namespace="http://www.w3.org/XML/1998/namespace" />
  <xsd:complexType name="locPolicyType">
    <xsd:sequence>
      <xsd:element minOccurs="0" maxOccurs="1" name="retransmission-allowed"
type="xsd:boolean" />
      <xsd:element minOccurs="0" maxOccurs="1" name="retention-expiry"
type="xsd:dateTime" />
      <xsd:element minOccurs="0" maxOccurs="1" name="external-ruleset" type="xsd:anyURI"
/>
      <xsd:element minOccurs="0" maxOccurs="1" name="note-well" type="tns:notewell" />
      <xsd:any minOccurs="0" maxOccurs="unbounded" namespace="##other"
processContents="lax" />
    </xsd:sequence>
```

```
    </xsd:complexType>
    <xsd:complexType name="notewell">
      <xsd:simpleContent>
        <xsd:extension base="xsd:string">
          <xsd:attribute ref="xml:lang" />
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>
  </xsd:schema>
```

7 Appendix B: 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 Lync Server 2010
- Microsoft Lync 2010
- Microsoft Lync Server 2013
- Microsoft Lync Client 2013/Skype for Business
- Microsoft Skype for Business 2016
- 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 2.1: Lync Server 2010, Lync 2010: These product versions support only Internet Protocol version 4 (IPv4). They do not support Internet Protocol version 6 (IPv6).

<2> Section 2.2.5.2: Lync Server 2010, Lync 2010: The limit is 1 to 454 characters.

<3> Section 3.1.4.1.1.1: Lync 2010: Lync 2010 populates the elements of the **GetLocationsRequest** request out of order. To interoperate with this client, the server needs to support requests that have the elements out of order. The out-of-order element sequence is specified in the product behavior note in section 3.1.4.1.2.1.

<4> Section 3.1.4.1.2.1: Lync 2010: The Lync 2010 implementation differs from the given schema, in the sense that it sends the **GetLocationsRequest** with a different order sequence of IP and MAC elements. To enable interoperability with Lync 2010, the server needs to use the following schema to validate the request.

```
<xsd:element name="GetLocationsRequest">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element minOccurs="1" maxOccurs="1" name="Entity"
type="tns:restrictedAnyURI" />
            <xsd:element minOccurs="0" maxOccurs="1" name="WAPBSSID"
type="tns:EnetMacAddressType" />
            <xsd:element minOccurs="0" maxOccurs="1" name="RSSI" type="xsd:unsignedByte"
/>
            <xsd:element minOccurs="0" maxOccurs="1" name="IP" type="tns:IPAddress" />
            <xsd:element minOccurs="0" maxOccurs="1" name="MAC"
type="tns:EnetMacAddressType" />
            <xsd:element minOccurs="0" maxOccurs="1" name="ChassisID"
type="tns:LLDPChassisIDOrPortIDTLVType" />
            <xsd:element minOccurs="0" maxOccurs="1" name="PortID"
type="tns:LLDPChassisIDOrPortIDTLVType" />
            <xsd:element minOccurs="0" maxOccurs="1" name="SubnetID" type="tns:IPAddress"
/>
        </xsd:sequence>
    </xsd:complexType>
```

```
</xsd:element>
```

<5> Section 4: Lync Server 2010: Though attribute id of the element tuple, has the type xsd:ID, the Lync Server implementation currently differs in a way that it has a colon (:) character in its value in the server response.

<6> Section 6: Lync Server 2010, Lync 2010: Though attribute **id** of the complex type **tuple** has the type **xsd:ID**, the current implementations differ in that they have a colon (:) character in the **id** value in the response.

8 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

9 Index

T

V

W