Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.

- **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 can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.

- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.

- **Patents.** Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.

- **License Programs.** To see all of the protocols in scope under a specific license program and the associated patents, visit the Patent Map.

- **Trademarks.** The names of companies and products contained in this documentation might 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, email addresses, logos, people, places, and events that are 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 as specifically described above, whether by implication, estoppel, or otherwise.

**Tools.** The Open Specifications documentation does 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 documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

**Support.** For questions and support, please contact dochelp@microsoft.com.
## Revision Summary

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision History</th>
<th>Revision Class</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>3/31/2010</td>
<td>0.1</td>
<td>Major</td>
<td>Initial Availability</td>
</tr>
<tr>
<td>4/30/2010</td>
<td>0.2</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>6/7/2010</td>
<td>0.3</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>6/29/2010</td>
<td>0.4</td>
<td>Editorial</td>
<td>Changed language and formatting in the technical content.</td>
</tr>
<tr>
<td>7/23/2010</td>
<td>0.4</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>9/27/2010</td>
<td>1.0</td>
<td>Major</td>
<td>Significantly changed the technical content</td>
</tr>
<tr>
<td>11/15/2010</td>
<td>1.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>12/17/2010</td>
<td>1.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>3/18/2011</td>
<td>1.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>6/10/2011</td>
<td>1.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>1/20/2012</td>
<td>2.0</td>
<td>Major</td>
<td>Significantly changed the technical content</td>
</tr>
<tr>
<td>4/11/2012</td>
<td>2.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>7/16/2012</td>
<td>2.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>10/8/2012</td>
<td>2.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>1/20/2013</td>
<td>2.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>7/30/2013</td>
<td>2.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>11/18/2013</td>
<td>2.1</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content</td>
</tr>
<tr>
<td>2/10/2014</td>
<td>2.1</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>4/30/2014</td>
<td>2.2</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content</td>
</tr>
<tr>
<td>7/31/2014</td>
<td>2.3</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content</td>
</tr>
<tr>
<td>10/30/2014</td>
<td>2.4</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content</td>
</tr>
<tr>
<td>3/30/2015</td>
<td>3.0</td>
<td>Major</td>
<td>Significantly changed the technical content</td>
</tr>
<tr>
<td>9/4/2015</td>
<td>3.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>7/15/2016</td>
<td>3.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>Date</td>
<td>Revision History</td>
<td>Revision Class</td>
<td>Comments</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
<td>----------------</td>
<td>----------------------------------------------------</td>
</tr>
<tr>
<td>9/14/2016</td>
<td>3.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>6/20/2017</td>
<td>3.1</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>4/27/2018</td>
<td>4.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>7/24/2018</td>
<td>5.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>8/28/2018</td>
<td>6.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>12/11/2018</td>
<td>6.1</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>3/19/2019</td>
<td>6.1</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
</tbody>
</table>
# Table of Contents

## 1 Introduction .................................................................................. 8
1.1 Glossary ......................................................................................... 8
1.2 References ..................................................................................... 11
   1.2.1 Normative References .............................................................. 11
   1.2.2 Informative References ............................................................ 13
1.3 Protocol Overview (Synopsis) .......................................................... 13
   1.3.1 Web Ticket Service .................................................................. 13
   1.3.1.1 Web Service Web Applications ........................................... 14
   1.3.1.2 Non-Web Service Web Applications .................................... 15
   1.3.2 Certificate Provisioning Service ................................................. 16
   1.3.3 Authentication Broker Service .................................................. 16
1.4 Relationship to Other Protocols ...................................................... 16
1.5 Prerequisites/Preconditions ............................................................ 17
1.6 Applicability Statement .................................................................. 18
1.7 Versioning and Capability Negotiation .......................................... 18
1.8 Vendor-Extensible Fields ............................................................... 18
1.9 Standards Assignments .................................................................. 18

## 2 Messages ................................................................................... 19
2.1 Transport ...................................................................................... 19
2.2 Common Message Syntax .............................................................. 19
   2.2.1 Namespaces ............................................................................ 19
   2.2.2 Messages ............................................................................... 20
   2.2.3 Elements ............................................................................... 20
   2.2.4 Complex Types ...................................................................... 20
      2.2.4.1 af:OCSDiagnosticsFaultType ........................................... 20
      2.2.4.2 af:MSWebAuthenticationType ......................................... 21
      2.2.4.3 af:BindingType ................................................................ 22
      2.2.4.4 tns:ErrorInfoType ............................................................ 22
   2.2.5 Simple Types ........................................................................... 22
      2.2.5.1 tns:ResponseClassType .................................................. 22
   2.2.6 Attributes .............................................................................. 23
      2.2.6.1 ResponseClass ................................................................. 23
   2.2.7 Groups ................................................................................... 23
   2.2.8 Attribute Groups ................................................................. 23

## 3 Protocol Details ........................................................................... 24
3.1 Certificate Provisioning Service Server Details ................................ 24
   3.1.1 Abstract Data Model ............................................................... 24
   3.1.2 Timers ................................................................................... 24
   3.1.3 Initialization .......................................................................... 24
   3.1.4 Message Processing Events and Sequencing Rules .................... 24
      3.1.4.1 GetAndPublishCert .......................................................... 25
         3.1.4.1.1 tns:GetAndPublishCertMsg ......................................... 25
         3.1.4.1.2 tns:GetAndPublishCertResponseMsg ........................... 25
         3.1.4.1.2 Elements .................................................................. 25
            3.1.4.1.2.1 tns:GetAndPublishCert ......................................... 26
            3.1.4.1.2.2 tns:GetAndPublishCertResponse ............................ 26
            3.1.4.1.2.3 wst:RequestSecurityToken .................................... 26
            3.1.4.1.2.4 wst:RequestSecurityTokenResponse ....................... 27
      3.1.4.1.3 Complex Types ............................................................... 27
         3.1.4.1.3.1 tns:GetAndPublishCertType ...................................... 28
         3.1.4.1.3.2 tns:GetAndPublishCertResponseType ....................... 28
         3.1.4.1.3.3 tns:GetAndPublishCertErrorInfoType ....................... 28
3.1.4.1.4 Simple Types ........................................... 29
3.1.4.1.4.1 tns:GetAndPublishResponseCodeType ............... 29
3.1.4.1.5 Attributes ........................................... 30
3.1.4.1.5.1 DeviceId .......................................... 30
3.1.4.1.5.2 Entity ............................................. 30
3.1.4.1.6 Groups ................................................ 30
3.1.4.1.7 Attribute Groups ................................... 30
3.1.5 Timer Events ............................................. 31
3.1.6 Other Local Events ....................................... 31
3.2 Web Ticket Service Server Details ........................................... 31
3.2.1 Abstract Data Model ....................................... 33
3.2.2 Timers .................................................... 33
3.2.3 Initialization .............................................. 33
3.2.4 Message Processing Events and Sequencing Rules ................. 33
3.2.4.1 IssueToken ........................................... 33
3.2.4.1.1 Messages ........................................... 35
3.2.4.1.1.1 tns:IWebTicketService_IssueToken_InputMessage .... 35
3.2.4.1.1.2 tns:IWebTicketService_IssueToken_OutputMessage ... 36
3.2.4.1.2 Elements ............................................ 36
3.2.4.1.3 Complex Types ..................................... 36
3.2.4.1.3.1 q1:MessageBody .................................. 36
3.2.4.1.3.2 q2:MessageBody .................................. 36
3.2.4.1.3.3 wst:RequestSecurityTokenMsg ..................... 36
3.2.4.1.3.4 wst:RequestSecurityTokenResponseMsg .......... 37
3.2.4.1.4 Simple Types ....................................... 38
3.2.4.1.5 Attributes .......................................... 38
3.2.4.1.6 Groups .............................................. 38
3.2.4.1.7 Attribute Groups .................................. 38
3.2.5 Timer Events ............................................. 38
3.2.6 Other Local Events ....................................... 38
3.3 Authentication Broker Service Server Details ................................. 38
3.3.1 Abstract Data Model ....................................... 39
3.3.2 Timers .................................................... 40
3.3.3 Initialization .............................................. 40
3.3.4 Message Processing Events and Sequencing Rules ..................... 40
3.3.4.1 CreateAuthBrokerSession .................................. 40
3.3.4.1.1 Messages ........................................... 40
3.3.4.1.1.1 tns:IAuthBroker_CreateAuthBrokerSession_InputMessage ... 41
3.3.4.1.1.2 tns:IAuthBroker_CreateAuthBrokerSession_OutputMessage .... 41
3.3.4.1.2 Elements ............................................ 41
3.3.4.1.2.1 tns:CreateAuthBrokerSession ...................... 41
3.3.4.1.2.2 tns:CreateAuthBrokerSessionResponse ........... 42
3.3.4.1.3 Complex Types ..................................... 42
3.3.4.1.3.1 tns:CreateAuthBrokerSessionResponse .......... 42
3.3.4.1.4 Simple Types ....................................... 42
3.3.4.1.5 Attributes .......................................... 43
3.3.4.1.6 Groups .............................................. 43
3.3.4.1.7 Attribute Groups .................................. 43
3.3.4.2 TerminateAuthBrokerSession .................................. 43
3.3.4.2.1 Messages ........................................... 43
3.3.4.2.1.1 tns:IAuthBroker_TerminateAuthBrokerSession_InputMessage .... 43
3.3.4.2.1.2 tns:IAuthBroker_TerminateAuthBrokerSession_OutputMessage .... 43
3.3.4.2.2 Elements ............................................ 44
3.3.4.2.2.1 tns:TerminateAuthBrokerSession ................... 44
3.3.4.2.2.2 tns:TerminateAuthBrokerSessionResponse ....... 44
3.3.4.2.3 Complex Types ..................................... 44
3.3.4.2.4 Simple Types ....................................... 44
3.3.4.2.5 Attributes .......................................... 44
3.3.4.2.6 Groups ......................................................................................... 45
3.3.4.2.7 Attribute Groups ......................................................................... 45
3.3.4.3 AuthBrokerAcquireCredential ......................................................... 45
3.3.4.3.1 Messages .................................................................................... 45
   3.3.4.3.1.1 tns:IAuthBroker_AuthBrokerAcquireCredential_InputMessage ...... 45
   3.3.4.3.1.2 tns:IAuthBroker_AuthBrokerAcquireCredential_OutputMessage .... 45
3.3.4.3.2 Elements .................................................................................... 45
   3.3.4.3.2.1 tns:AuthBrokerAcquireCredential ........................................ 46
   3.3.4.3.2.2 tns:AuthBrokerAcquireCredentialResponse .......................... 46
3.3.4.3.3 Complex Types ......................................................................... 46
3.3.4.3.4 Simple Types ............................................................................ 46
3.3.4.3.5 Attributes .................................................................................. 46
3.3.4.3.6 Groups ....................................................................................... 47
3.3.4.3.7 Attribute Groups ........................................................................ 47
3.3.4.4 AuthBrokerNegotiateSecurityAssociation ...................................... 47
3.3.4.4.1 Messages .................................................................................... 47
   3.3.4.4.1.1 tns:IAuthBroker_AuthBrokerNegotiateSecurityAssociation_InputMessage ......................................................................................... 47
   3.3.4.4.1.2 tns:IAuthBroker_AuthBrokerNegotiateSecurityAssociation_OutputMessage ......................................................................................... 47
3.3.4.4.2 Elements .................................................................................... 48
   3.3.4.4.2.1 AuthBrokerNegotiateSecurityAssociation ................................. 48
   3.3.4.4.2.2 AuthBrokerNegotiateSecurityAssociationResponse ................ 48
3.3.4.4.3 Complex Types ......................................................................... 49
   3.3.4.4.3.1 tns:NegotiateSaResponse ...................................................... 49
   3.3.4.4.3.2 tns:SAReturnData ................................................................. 49
   3.3.4.4.3.3 tns:AuthReturnValuePair .................................................... 50
3.3.4.4.4 Simple Types ............................................................................ 51
3.3.4.4.5 Attributes .................................................................................. 51
3.3.4.4.6 Groups ....................................................................................... 51
3.3.4.4.7 Attribute Groups ...................................................................... 52
3.3.5 Timer Events ..................................................................................... 52
3.3.6 Other Local Events .......................................................................... 52

4 Protocol Examples .................................................................................. 53
4.1 GetAndPublishCert ............................................................................. 53
   4.1.1 Request .......................................................................................... 53
   4.1.2 Response ....................................................................................... 53
4.2 IssueToken .......................................................................................... 55
   4.2.1 Request .......................................................................................... 55
   4.2.2 Response ....................................................................................... 55
4.3 CreateAuthBrokerSession ................................................................. 57
   4.3.1 Request .......................................................................................... 57
   4.3.2 Response ....................................................................................... 59
4.4 TerminateAuthBrokerSession ............................................................ 59
   4.4.1 Request .......................................................................................... 59
   4.4.2 Response ....................................................................................... 61
4.5 AuthBrokerAcquireCredential ........................................................... 61
   4.5.1 Request .......................................................................................... 61
   4.5.2 Response ....................................................................................... 63
4.6 AuthBrokerNegotiateSecurityAssociation ......................................... 63
   4.6.1 Request .......................................................................................... 63
   4.6.2 Response ....................................................................................... 65

5 Security ................................................................................................. 66
5.1 Security Considerations for Implementers ............................................ 66
5.2 Index of Security Parameters .............................................................. 66
1 Introduction

The OC Authentication Web Service Protocol defines the message formats, server behavior, and client behavior for the purposes of authentication and certificate enrollment.

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:

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

**base64 encoding**: A binary-to-text encoding scheme whereby an arbitrary sequence of bytes is converted to a sequence of printable ASCII characters, as described in [RFC4648].

**certificate**: (1) A certificate is a collection of attributes 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.

(2) When referring to X.509v3 certificates, that information consists of a public key, a distinguished name (DN) of some entity assumed to have control over the private key corresponding to the public key in the certificate, and some number of other attributes and extensions assumed to relate to the entity thus referenced. Other forms of certificates can bind other pieces of information.

**certificate chain**: A sequence of certificates, where each certificate in the sequence is signed by the subsequent certificate. The last certificate in the chain is normally a self-signed certificate.

**certification**: The certificate (1) request and issuance process whereby an end entity first makes itself known to a certification authority (CA) (directly, or through a registration authority) through the submission of a certificate enrollment request, prior to that CA issuing a certificate (1) or certificates (1) for that end entity.

**certification authority (CA)**: A third party that issues public key certificates (1). Certificates serve to bind public keys to a user identity. Each user and certification authority (CA) can decide whether to trust another user or CA for a specific purpose, and whether this trust should be transitive. For more information, see [RFC3280].

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

**fully qualified domain name (FQDN)**: An unambiguous domain name 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 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).
**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].

**Integrated Windows authentication:** A configuration setting that enables negotiation of authentication protocols in Internet Information Services (IIS). Integrated Windows authentication is more secure than Basic authentication, because the user name and password are hashed instead of plaintext.

**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].

**private key:** One of a pair of keys used in public-key cryptography. The private key is kept secret and is used to decrypt data that has been encrypted with the corresponding public key. For an introduction to this concept, see [CRYPTO] section 1.8 and [IEEE1363] section 3.1.

**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 key:** One of a pair of keys used in public-key cryptography. The public key is distributed freely and published as part of a digital certificate. For an introduction to this concept, see [CRYPTO] section 1.8 and [IEEE1363] section 3.1.

**Security Assertion Markup Language (SAML):** The set of specifications that describe security assertions encoded in XML, profiles for attaching assertions to protocols and frameworks, request/response protocols used to obtain assertions, and the protocol bindings to transfer protocols, such as SOAP and HTTP.

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

**security token:** An opaque message or data packet produced by a Generic Security Services (GSS)-style authentication package and carried by the application protocol. The application has no visibility into the contents of the token.

**security token service (STS):** A web service that issues claims and packages them in encrypted security tokens.

**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.

**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 fault: A container for error and status information within a SOAP message. See [SOAP1.2-1/2007] section 5.4 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.

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.

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].

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 application: A software application that uses HTTP as its core communication protocol and delivers information to the user by using web-based languages such as HTML and XML.

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.

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.

web ticket: A security token that is sent by a protocol client to a web application during authentication. The security token can be included in either the body or the header of an HTTP message.

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.

WSDL operation: A single action or function of a web service. The execution of a WSDL operation typically requires the exchange of messages between the service requestor and the service provider.
X.509: An ITU-T standard for public key infrastructure subsequently adapted by the IETF, as specified in [RFC3280].

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 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.


[MS-OAUTH2EX] Microsoft Corporation, "OAuth 2.0 Authentication Protocol Extensions".

[MS-OCER] Microsoft Corporation, "Client Error Reporting Protocol".


[MS-WSTEP] Microsoft Corporation, "WS-Trust X.509v3 Token Enrollment Extensions".


1.3 Protocol Overview (Synopsis)

This protocol can be used to generate a **security token**, which can subsequently be used for **authentication** with other services. This protocol also allows a protocol client to request **X.509 v3 certificates** (2), which can subsequently be used for certificate-based authentication.

This protocol is used by the Web Ticket Service, described in section 1.3.1, by the Certificate Provisioning Service, described in section 1.3.2, and by the Authentication Broker Service, described in section 1.3.3.

1.3.1 Web Ticket Service

The Web Ticket Service is a **security token service (STS)**. The type of credentials that a client presents to the Web Ticket Service is described in section 3.2. The **security token** returned in the response is called a **Web ticket**.

The client presents the Web ticket as its credentials when authenticating to certain **Web applications**. See the individual Web application specifications for details, for example, Lync Autodiscover Web Service described in [MS-OCDISCWS] or Certificate Provisioning Service described in this document. The Web ticket can be presented in the body of the **Hypertext Transfer Protocol (HTTP)** message or in the HTTP header, depending on the type of Web application.
1.3.1.1 Web Service Web Applications

The following figure illustrates this protocol for Web applications that are Web services.

![Diagram of Web Service Web Applications Protocol]

Figure 1: This protocol for Web service Web applications

1. The client requests the metadata for the Web service using WS Metadata Exchange protocol as described in [WS-MetaDataExchange].

2. The Web service metadata is returned. The client discovers the Uniform Resource Locator (URL) of the Web Ticket Service. See details in section 3.2.

3. The client requests the metadata for the Web Ticket Service.

4. The Web Ticket Service metadata is returned. The following authentication types can be associated with the bindings in the metadata: Integrated Windows authentication, OCS-signed certificate authentication, and Live ID authentication. For details, see section 3.2.

5. The client sends an RST (Request Security Token). For details, see section 3.2.4.1.1.1.

6. The Web Ticket Service responds with an RSTR (Request Security Token Response). For details, see section 3.2.4.1.1.2.

7. The client sends a request to the Web service, with the Web ticket attached. For details, see section 3.2.
8. The Web service sends a response.

### 1.3.1.2 Non-Web Service Web Applications

The following figure illustrates this protocol for **Web applications** that are non-Web services.

![Diagram](image)

**Figure 2: This protocol for non-Web service Web applications**

1. The client sends a GET or POST HTTP request to the non-Web service Web application with content defined by the requirements of that application.

2. A response with status code 401 and a HTTP header containing the **URL** of the Web Ticket Service. For details, see section 3.2.

3. The client requests the Web Ticket Service's metadata using WS Metadata Exchange protocol as described in [WS-MetaDataExchange](#).

4. The Web Ticket Service metadata is returned. The following **authentication** types can be associated with the bindings in the metadata: **Integrated Windows authentication**, OCS-signed certificate authentication, and Live ID authentication. For details, see section 3.2.

5. The client sends an **RST** (Request Security Token). For details, see section 3.2.4.1.1.1.

6. The Web Ticket Service responds with a **RSTR** (Request Security Token Response). For details, see section 3.2.4.1.1.2.
7. The client sends a request to the non-Web service Web application, with the **Web ticket** in an HTTP header. For details, see section 3.2.

8. The Web service sends a response.

### 1.3.2 Certificate Provisioning Service

The Certificate Provisioning Service provides an **X.509 v3 certificate** (2) for the authenticated user to the client. The client can use the obtained certificate (2) for **authentication** against other services. One example of an authentication mechanism that uses this certificate (2) can be found in [MS-SIPAE] section 4.4.

### 1.3.3 Authentication Broker Service

The Authentication Broker Service provides a web service-based **TLS** implementation. This is to be used by a client that does not have local support for TLS and wishes to use TLS-DSK **authentication** mechanism with the **SIP** server which is detailed in [MS-SIPAE].

The following diagram illustrates the sequence of events. Details of the call flows are explained in section 3.3.

![Sequence of events for Authentication Broker Service](image)

#### Figure 3: Sequence of events for Authentication Broker Service

### 1.4 Relationship to Other Protocols

The Web Ticket Service and **Web applications** that accept **Web tickets** as client credentials use **Simple Object Access Protocol (SOAP)** over **Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)**, as described in [RFC2818], SOAP 1.1, as described in [SOAP1.1], and **WS-Trust 1.3**, as described in [WS-Trust1.3], as shown in the following figure.
1.5 Prerequisites/Preconditions

This protocol facilitates the issuance of X.509 v3 certificates (2). A server implementation of the protocol requires the functionality of a certification authority (CA), capable of interpreting requests in PKCS#10, as described in [RFC2986], and generating the appropriate certificate (2).

Protocol clients are required to be able to understand PKCS#7 format, as described in [RFC2315] and [RFC5652], and X.509 v3 certificate (2) format, as described in [RFC5280], which are used by the server to send the certificate chain and the certificate (2).

A protocol client needs to retrieve the Web Ticket Service **URL** before using this protocol. The two ways for the client to do so are shown in the figures in section 1.3.1.1. If the client retrieves it from a **Web service**, the URL ought to be read from the metadata document of a participating Web service, from the `wsp:Policy/sp:IssuedToken/sp:Issuer/wsa10:Address` element associated with the service's binding that accepts a Web ticket, as described in [WSSP1.2-2012] and [WS-MetaDataExchange]. If the client retrieves it from a non-Web service, the **Web application** is required to return it in a 401 response in an **HTTP** header extension named **X-MS-WebTicketURL**, as described in [MS-OCDISCWS].

In order to use the Authentication Broker Service, a protocol client needs to retrieve the Internal/External AuthBroker Service URL, which is included as part of the User type in the response of the Lync Autodiscover Web Service [MS-OCDISCWS]. The section below shows a sample response.

```xml
<AutodiscoverResponse AccessLocation="Internal" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <User>
    <Link token="Internal/Autodiscover" href="https://pool1.contoso.com/Autodiscover/AutodiscoverService.svc/root/>
    <Link token="Internal/AuthBroker" href="https://pool1.contoso.com/Reach/sip.svc"/>
  </User>
</AutodiscoverResponse>
```

---

**Where:**

- This protocol
- Industry standard

---

**Figure 4: This protocol in relation to other protocols**
1.6 Applicability Statement

This protocol is applicable when clients require authentication with servers using X.509 v3 certificates (2).

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

This protocol provides extensibility by the use of any and anyAttribute in the schema, as specified in [XMLSCHEMA1/2]. Vendors can choose to include their own elements by taking advantage of this extensibility.

1.9 Standards Assignments

None.
2 Messages

2.1 Transport

This protocol uses the SOAP message protocol for formatting request and response messages, as specified in [SOAP1.2/1] and [SOAP1.2/2]. It transmits those messages using HTTPS, as specified in [RFC2818].

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 [XMLSCHEMA1/2] and [XMLSCHEMA2/2], and WSDL, as specified in [WSDL].

The table in section 2.2.1 lists common namespaces.

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.

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Namespace URI</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>xs</td>
<td><a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a></td>
<td>[XMLSCHEMA1/2]</td>
</tr>
<tr>
<td>xsi</td>
<td><a href="http://www.w3.org/2001/XMLSchema-instance">http://www.w3.org/2001/XMLSchema-instance</a></td>
<td>[XMLSCHEMA1/2]</td>
</tr>
<tr>
<td>xml</td>
<td><a href="http://www.w3.org/XML/1998/namespace">http://www.w3.org/XML/1998/namespace</a></td>
<td>[XMLSCHEMA1/2]</td>
</tr>
<tr>
<td>wst</td>
<td><a href="http://docs.oasis-open.org/ws-sx/ws-trust/200512/">http://docs.oasis-open.org/ws-sx/ws-trust/200512/</a></td>
<td>[WS-Trust1.3]</td>
</tr>
<tr>
<td>tns</td>
<td><a href="http://schemas.microsoft.com/OCS/AuthWebServices/">http://schemas.microsoft.com/OCS/AuthWebServices/</a></td>
<td></td>
</tr>
<tr>
<td>soap</td>
<td><a href="http://schemas.xmlsoap.org/wsd/soap/">http://schemas.xmlsoap.org/wsd/soap/</a></td>
<td>[WSDL]</td>
</tr>
<tr>
<td>wsdl</td>
<td><a href="http://schemas.xmlsoap.org/wsd/">http://schemas.xmlsoap.org/wsd/</a></td>
<td>[WSDL]</td>
</tr>
<tr>
<td>auth</td>
<td><a href="http://schemas.xmlsoap.org/ws/2006/12/authorization">http://schemas.xmlsoap.org/ws/2006/12/authorization</a></td>
<td>[WSFederation]</td>
</tr>
<tr>
<td>wsse</td>
<td><a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssext-secext-1.0.xsd">http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssext-secext-1.0.xsd</a></td>
<td>[WSSE 1.0]</td>
</tr>
<tr>
<td>saml</td>
<td>urn:oasis:names:tc:SAML:1.0:assertion</td>
<td>[SAMLCore]</td>
</tr>
<tr>
<td>af</td>
<td>urn:component::Microsoft.Rtc.WebAuthentication.2010</td>
<td></td>
</tr>
<tr>
<td>wsaw</td>
<td><a href="http://www.w3.org/2006/05/addressing/wsd">http://www.w3.org/2006/05/addressing/wsd</a></td>
<td>[WSA1.0]</td>
</tr>
<tr>
<td>sp</td>
<td><a href="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702</a></td>
<td>[WSSP1.2-2012]</td>
</tr>
<tr>
<td>wsa10</td>
<td><a href="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing</a></td>
<td>[WSA1.0 Core]</td>
</tr>
<tr>
<td>Prefix</td>
<td>Namespace URI</td>
<td>Reference</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>soap12</td>
<td><a href="http://schemas.xmlsoap.org/wsdl/soap12">http://schemas.xmlsoap.org/wsdl/soap12</a></td>
<td>[WSDL SOAP]</td>
</tr>
<tr>
<td>wsu</td>
<td><a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd</a></td>
<td></td>
</tr>
<tr>
<td>wsam</td>
<td><a href="http://www.w3.org/2007/05/addressing/metadata">http://www.w3.org/2007/05/addressing/metadata</a></td>
<td>[WSA1.0 Metadata]</td>
</tr>
<tr>
<td>soapenc</td>
<td><a href="http://schemas.xmlsoap.org/soap/encoding">http://schemas.xmlsoap.org/soap/encoding</a></td>
<td></td>
</tr>
</tbody>
</table>

### 2.2.2 Messages

This specification does not define any common **WSDL** message definitions.

### 2.2.3 Elements

This specification does not define any common **XML schema** element definitions.

### 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.

<table>
<thead>
<tr>
<th>Complex type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>af:OCSDiagnosticsFaultType</td>
<td>Authentication-specific error information in the <strong>SOAP fault</strong> detail. It is returned for some failures during Live ID <strong>authentication</strong> or <strong>Web ticket</strong> verification at a <strong>Web service</strong>.</td>
</tr>
<tr>
<td>af:MSWebAuthenticationType</td>
<td><strong>WS-Policy</strong> assertion that describes the Live ID environment.</td>
</tr>
<tr>
<td>af:BindingType</td>
<td><strong>WS-Policy</strong> assertion that the protocol client can communicate with the associated port. The absence of this assertion means that the client MUST NOT communicate with the associated <strong>WSDL</strong> port.</td>
</tr>
<tr>
<td>tns:ErrorInfoType</td>
<td>The base type of all the types that describe errors in any operation.</td>
</tr>
</tbody>
</table>

#### 2.2.4.1 af:OCSDiagnosticsFaultType

The **af:OCSDiagnosticsFaultType** element is a child element of **s:Fault/s:detail**, as defined in [SOAP1.1].

```xml
<xs:complexType name="OCSDiagnosticsFaultType">
```

[MS-OCAUTHWS] - v20190319
OC Authentication Web Service Protocol
Copyright © 2019 Microsoft Corporation
Release: March 19, 2019
The **af:Ms-Diagnostics-Fault** element is a child element of **af:OCSDiagnosticsFaultType** element. It describes the authentication-specific error information.

The **af:ErrorId** element carries a unique positive integer value for each specific error condition.

The **af:Reason** element carries a string that provides a reason for an explanation of specific error.

Error IDs and reason string used by OC Authentication Web Service are documented in Section 7.22 of [MS-OCER].

### 2.2.4.2 af:MSWebAuthenticationType

The **af:MSWebAuthenticationType** element is a WS-Policy assertion and a child element of the **wsp:Policy** element. It contains policy elements that provide information about a security token service that can issue tokens accepted by OC Authentication Web Service.

The **af:LiveIdEnvironmentType** element is a child element of the **wsp:Policy** element inside **af:MSWebAuthenticationType**. It describes the environment in which the security token service operates.

The "**PRODUCTION**" enumeration value indicates production environment.

The "**PPE**" enumeration value indicates pre-production environment.
The "INT" enumeration value indicates integration environment.

### 2.2.4.3 af:BindingType

The **af:BindingType** element is a WS-Policy assertion and a child element of the **wsp:Policy** element.

```xml
<xs:complexType name="BindingType">
  <xs:sequence>
    <xs:any processContents="lax" namespace="##any" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>
```

### 2.2.4.4 tns:ErrorInfoType

The **tns:ErrorInfoType** type is defined as follows.

```xml
<xs:complexType name="ErrorInfoType">
  <xs:sequence>
    <xs:element name="Description" type="xs:string" minOccurs="0" maxOccurs="1" />
    <xs:element name="AdditionalContext" minOccurs="0" maxOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:any processContents="lax" namespace="##any" minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>
```

tns:Description: Contains a textual description of the error.
tns:AdditionalContext: Can contain any implementation-defined context.

### 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.

<table>
<thead>
<tr>
<th>Simple type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tns:ResponseClassType</td>
<td>Specifies whether the response for an operation is success, warning, or failure.</td>
</tr>
</tbody>
</table>

### 2.2.5.1 tns:ResponseClassType

The **tns:ResponseClassType** type is defined as follows.

```xml
<xs:simpleType name="ResponseClassType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Success" />  
    <xs:enumeration value="Warning" />  
    <xs:enumeration value="Error" />
  </xs:restriction>
</xs:simpleType>
```
The enumeration values have the usual meaning, and are used by the server to represent the class of the response.

### 2.2.6 Attributes

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tns:ResponseClass</td>
<td>An instance of ResponseClassType that specifies the class of Response.</td>
</tr>
</tbody>
</table>

#### 2.2.6.1 ResponseClass

The ResponseClass attribute is defined as follows.

```
<x:simpleType name="ResponseClassType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Request"/>
        <xs:enumeration value="Successful"/>
        <xs:enumeration value="Failed"/>
    </xs:restriction>
</xs:simpleType>
```

This attribute is an instance of type ResponseClassType, which is defined in section 2.2.5.1. It appears as a required attribute in all the responses of the GetAndPublishCert operation, which is defined in section 3.1.4.1.

### 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.
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 Certificate Provisioning Service Server Details

The Certificate Provisioning Service hosts a message endpoint that receives GetAndPublishCert messages. When received, the server uses the certification request, which is part of the message, to generate and sign a certificate (2). It then stores the certificate (2) in an implementation-defined manner, so that it can be used to verify a client certificate (2) presented for authentication. After that, it sends the certificate (2) to the client as part of GetAndPublishCertResponse, as specified in section 3.1.4.1.2.2.

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 server SHOULD keep the following states:

Certificate Issuer: A proxy, with which the server can communicate with a CA, used for generating X.509 v3 certificates (2). The nature of the proxy is implementation-dependent.

Trusted Certificate Authorities: A list of CAs whose certificate chains are required to be trusted by the protocol clients in order for them to create Transport Layer Security (TLS) connections with the server. This list MUST have sufficient data that the certificates (2) in the chain can be located.

3.1.2 Timers

None.

3.1.3 Initialization

The CA that would be used for generating X.509 v3 certificates (2) SHOULD be initialized with at least one public key/private key pair, used for signing the certificates (2).

The certificate (2) issuer proxy SHOULD be constructed and initialized, so that it can communicate with the CA.

The Trusted Certificate Authorities list SHOULD be initialized.

3.1.4 Message Processing Events and Sequencing Rules

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

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetAndPublishCert</td>
<td>A mechanism for clients to get a certificate (2), which can then be used for authentication purposes.</td>
</tr>
</tbody>
</table>
3.1.4.1 GetAndPublishCert

This operation is defined as part of the CertProvisioningService portType.

```
<wsdl:operation name="GetAndPublishCert">
  <wsdl:input message="tns:GetAndPublishCertMsg" />
  <wsdl:output message="tns:GetAndPublishCertResponseMsg" />
</wsdl:operation>
```

GetAndPublishCert generates a X.509 v3 certificate (2) using the PKCS#10 certification request in the request, and then stores the certificate (2) in an implementation-specific manner, so that it can be used to verify client certificates (2) supplied during authentication. If an error occurs during processing, an error response MUST be sent using the ErrorInfo element in GetAndPublishCertResponse, as specified in section 3.1.4.1.2.2. SOAP faults SHOULD NOT be used for error reporting.

3.1.4.1.1 Messages

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

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tns:GetAndPublishCertMsg</td>
<td>The request for certificate provisioning.</td>
</tr>
<tr>
<td>tns:GetAndPublishCertResponseMsg</td>
<td>The response for certificate provisioning.</td>
</tr>
</tbody>
</table>

3.1.4.1.1.1 tns:GetAndPublishCertMsg

The tns:GetAndPublishCertMsg represents the incoming message and is defined as follows.

```
<wsdl:message name="GetAndPublishCertMsg">
  <wsdl:part name="request" element="tns:GetAndPublishCert" />
</wsdl:message>
```

tns:GetAndPublishCert: Refers to the GetAndPublishCert definition in section 3.1.4.1.2.1.

3.1.4.1.1.2 tns:GetAndPublishCertResponseMsg

The tns:GetAndPublishCertResponseMsg represents the outgoing message and is defined as follows.

```
<wsdl:message name="GetAndPublishCertResponseMsg">
  <wsdl:part name="response" element="tns:GetAndPublishCertResponse" />
</wsdl:message>
```

tns:GetAndPublishCertResponse: Refers to the GetAndPublishCertResponse definition in section 3.1.4.1.2.2.

3.1.4.1.2 Elements

The following table summarizes the XML schema element definitions that are specific to this operation.
<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tns:GetAndPublishCert</td>
<td>Container for the client request for certificate provisioning.</td>
</tr>
<tr>
<td>tns:GetAndPublishCertResponse</td>
<td>Container for the response to a request for certificate provisioning.</td>
</tr>
<tr>
<td>wst:RequestSecurityToken</td>
<td>Used to request a security token (for any purpose).</td>
</tr>
<tr>
<td>wst:RequestSecurityTokenResponse</td>
<td>Used to return a security token or response to a security token request.</td>
</tr>
</tbody>
</table>

### 3.1.4.1.2.1 tns:GetAndPublishCert

The `tns:GetAndPublishCert` element contains the client request, and is defined as follows.

```xml
<xsd:element name="GetAndPublishCert" type="tns:GetAndPublishCertType" />
```

**tns:GetAndPublishCertType**: Refers to the `GetAndPublishCertType` definition in section 3.1.4.1.3.1.

### 3.1.4.1.2.2 tns:GetAndPublishCertResponse

The `tns:GetAndPublishCertResponse` element contains the response from server, and is defined as follows.

```xml
<xsd:element name="GetAndPublishCertResponse" type="tns:GetAndPublishCertResponseType" />
```

**tns:GetAndPublishCertResponseType**: Refers to the `GetAndPublishCertResponseType` definition in section 3.1.4.1.3.2.

### 3.1.4.1.2.3 wst:RequestSecurityToken

The `wst:RequestSecurityToken` element is defined in [WS-Trust1.3] section 3.1, and further extended in [MS-WSTEP] section 3.1.4.1.2.5. For this protocol, this element MUST be a child of the `GetAndPublishCert` element and has the following extra restrictions:

1. `/wst:RequestedSecurityToken/wst:RequestType` MUST be "http://docs.oasis-open.org/wss/200512/wst-200512r01/Issue".
2. `/wst:RequestedSecurityToken/wst:TokenType` MUST be "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3".
3. `/wst:RequestedSecurityToken/wsse:BinarySecurityToken` MUST contain a PKCS#10 Certification Signing Request (CSR) ([MS-WCCE] section 2.2.2.6.1) encoded with base64 encoding
4. `/wst:RequestedSecurityToken/wsBinarySecurityToken/@EncodingType` MUST be "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd#base64binary".
5. The `/wst:RequestedSecurityToken/wsBinarySecurityToken/@ValueType` attribute MUST be "http://schemas.microsoft.com/OCS/AuthWebServices.xsd#PKCS10".
Any optional element or attribute not mentioned in this section SHOULD be ignored.

The server SHOULD be able to process **ValidityPeriod** and **ValidityPeriodUnits**, as specified in [MS-WCCE] section 3.1.4.3.1.1.

### 3.1.4.1.2.4 wst:RequestSecurityTokenResponse

The **wst:RequestSecurityTokenResponse** element is defined in [WS-Trust1.3] section 3.2, and is further extended in [MS-WSTEP] section 3.1.4.1.3.4. For this protocol, this element is a child of the **GetAndPublishCertResponse** element.

In case of an error, this element MUST NOT be present in the **GetAndPublishCertResponse**.

In case of success, the following restrictions MUST be adhered to:

1. **/wst:RequestSecurityTokenResponse/wstep:DispositionMessage** MUST be "Issued".
2. **/wst:RequestSecurityTokenResponse /wstep:DispositionMessage/@lang** attribute MUST be "en-US".
3. **/wst:RequestSecurityTokenResponse/wst:TokenType** MUST be "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3".
4. **/wst:RequestSecurityTokenResponse/wst:RequestedSecurityToken** MUST contain **BinarySecurityToken**, which MUST contain the X.509 v3 certificate (2) using base64 encoding.
5. The **Common Name** of the **Subject** (Section 4.1.2.6 of [RFC3280]) in the returned certificate (2) MUST have the same value as the **Entity** attribute in the client request.
6. **SubjectKeyIdentifier** (Section 4.2.1.2 of [RFC3280]) in the returned certificate (2) SHOULD contain the value of the **DeviceId** attribute in the client request.
7. **/wst:RequestSecurityTokenResponse/wst:RequestedSecurityToken/wsse:BinarySecurityToken/@ValueType** MUST be "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3".
9. **/wst:RequestSecurityTokenResponse/wsse:BinarySecurityToken** MUST contain the **BinarySecurityToken** that came as part of the incoming request.

Any element or attribute not mentioned in this section SHOULD be ignored.

### 3.1.4.1.3 Complex Types

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

<table>
<thead>
<tr>
<th>Complex type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tns:GetAndPublishCertType</td>
<td>Describes the client request for certificate provisioning.</td>
</tr>
<tr>
<td>tns:GetAndPublishCertResponseType</td>
<td>Describes the server response to a request for certificate provisioning.</td>
</tr>
<tr>
<td>tns:GetAndPublishCertErrorInfoType</td>
<td>Describes any failure in a <strong>GetAndPublishCert</strong> operation.</td>
</tr>
</tbody>
</table>
### 3.1.4.1.3.1  `tns:GetAndPublishCertType`  

The `tns:GetAndPublishCertType` type describes the client request and is defined as follows.

```xml
<xs:complexType name="GetAndPublishCertType">
  <xs:sequence>
    <xs:element ref="wst:RequestSecurityToken" minOccurs="1" maxOccurs="1" />
    <xs:any namespace="#other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="DeviceId" type="xs:string" use="required" />
  <xs:attribute name="Entity" type="xs:anyURI" use="required" />
  <xs:anyAttribute namespace="#other" processContents="lax" />
</xs:complexType>

`wst:RequestSecurityToken`: Refers to the `RequestSecurityToken`, as defined in section 3.1.4.1.2.3.

`DeviceId`: Refers to the `DeviceId`, as defined in section 3.1.4.1.5.1.

`Entity`: Refers to the `Entity`, as defined in section 3.1.4.1.5.2.

### 3.1.4.1.3.2  `tns:GetAndPublishCertResponseType`  

The `tns:GetAndPublishCertResponseType` type describes the server response and is defined as follows.

```xml
<xs:complexType name="GetAndPublishCertResponseType">
  <xs:sequence>
    <xs:element ref="wst:RequestSecurityTokenResponse" minOccurs="0" maxOccurs="1" />
    <xs:element name="ErrorInfo" type="tns:GetAndPublishCertErrorInfoType" minOccurs="0" maxOccurs="1" />
  </xs:sequence>
  <xs:attribute name="DeviceId" type="xs:string" use="required" />
  <xs:attribute name="Entity" type="xs:anyURI" use="required" />
  <xs:attribute name="ResponseClass" type="tns:ResponseClassType" use="required" />
  <xs:anyAttribute namespace="#other" processContents="lax" />
</xs:complexType>

`wst:RequestSecurityTokenResponse`: Refers to `RequestSecurityTokenResponse` element in section 3.1.4.1.2.4.

`ErrorInfo`: This element contains information about the error that occurred. It MUST be an instance of the `GetAndPublishCertErrorInfoType`, as defined in section 3.1.4.1.3.3.

`DeviceId`: Refers to the `DeviceId` definition in section 3.1.4.1.5.1. This attribute contains the same value as the one contained in the `DeviceId` attribute of the client request.

`Entity`: Refers to the `Entity` definition in section 3.1.4.1.5.2. This attribute contains the same value as the one contained in `Entity` attribute of the client request.

`ResponseClass`: Refers to the `ResponseClass` definition in section 2.2.6.1.

### 3.1.4.1.3.3  `tns:GetAndPublishCertErrorInfoType`  

The `tns:GetAndPublishCertErrorInfoType` type is defined as follows.

```xml
<xs:complexType name="GetAndPublishCertErrorInfoType">
  <xs:complexContent>
    <xs:extension base="ErrorInfoType">
      <xs:attribute name="ResponseCode" type="GetAndPublishCertResponseCodeType" use="required" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

[MS-OCAUTHWS] - v20190319
OC Authentication Web Service Protocol
Copyright © 2019 Microsoft Corporation
Release: March 19, 2019
It is used to describe any failure in a GetAndPublishCert operation. 
**tns:ResponseCode**: It MUST be an instance of a GetAndPublishCertResponseCodeType, as defined in section 3.1.4.1.4.1, and contains a code that describes the failure.

### 3.1.4.1.4 Simple Types

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

<table>
<thead>
<tr>
<th>Simple type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tns:GetAndPublishResponseCodeType</td>
<td>The status of the certificate provisioning request.</td>
</tr>
</tbody>
</table>

#### 3.1.4.1.4.1  tns:GetAndPublishResponseCodeType

The tns:GetAndPublishResponseCodeType type is defined as follows.

```xml
<xs:simpleType name="GetAndPublishCertResponseCodeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="NoError" />
    <xs:enumeration value="InternalError" />
    <xs:enumeration value="InvalidPublicKey" />
    <xs:enumeration value="InvalidValidityPeriod" />
    <xs:enumeration value="InvalidEKU" />
    <xs:enumeration value="InvalidSipUri" />
    <xs:enumeration value="InvalidCSR" />
    <xs:enumeration value="DataStoreUnavailable" />
    <xs:enumeration value="InvalidDeviceId" />
    <xs:enumeration value="RequestMalformed" />
    <xs:enumeration value="AccountDisabled" />
    <xs:enumeration value="UserImproperlyProvisioned" />
  </xs:restriction>
</xs:simpleType>
```

- **NoError**: Indicates success.
- **InternalError**: Indicates an unexpected server error.
- **InvalidPublicKey**: Indicates that the certification request did not contain a valid public key.
- **InvalidValidityPeriod**: Indicates that the CSR contained an invalid or unacceptable validity period.
- **InvalidEKU**: Indicates that the CSR contained invalid Enhanced Key Usage.
- **InvalidSipUri**: Indicates that the Entity, as defined in section 3.1.4.1.5.2, is invalid.
- **InvalidCSR**: Indicates that the CSR is invalid.
- **DataStoreUnavailable**: Indicates that the store where the certificate (2) was supposed to be stored was not available.
- **InvalidDeviceId**: Indicates that the DeviceId, as defined in section 3.1.4.1.5.1, is invalid.
- **RequestMalformed**: Indicates that the wst:RequestSecurityToken, as defined in section 3.1.4.1.2.3, is invalid.
**AccountDisabled:** Indicates that the account of the user operating the client is disabled.

**UserImproperlyProvisioned:** Indicates that the user is not provisioned on a server that supports this protocol.

### 3.1.4.1.5 Attributes

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeviceId</td>
<td>Part of GetAndPublishCertType, as specified in section 3.1.4.1.3.1, and GetAndPublishCertResponseType, as specified in section 3.1.4.1.3.2.</td>
</tr>
<tr>
<td>Entity</td>
<td>Part of GetAndPublishCertType and GetAndPublishCertResponseType.</td>
</tr>
</tbody>
</table>

#### 3.1.4.1.5.1 DeviceId

The DeviceId attribute is part of GetAndPublishCertType and GetAndPublishCertResponseType, and is defined as follows.

```xml
<xs:attribute name="DeviceId" type="xs:string" use="required" />
```

This is an identifier for the device on which the client is operating, and serves to identify a device unique among the various devices that the same user might be using simultaneously. It MUST be unique for each device being used by the same user. DeviceId MUST be convertible to a GUID. If the client uses an identifier for the device with any other service, which uses the certificate (2) retrieved using the GetAndPublishCert operation for authentication, DeviceId and the aforementioned identifier MUST be equal or it MUST be possible for the DeviceId to be generated using the identifier using a deterministic mathematical transformation. This transformation MUST be known to the certificate (2) verification engine.

#### 3.1.4.1.5.2 Entity

The Entity attribute is part of GetAndPublishCertType and GetAndPublishCertResponseType, and is defined as follows.

```xml
<xs:attribute name="Entity" type="xs:anyURI" use="required" />
```

This is an identifier for the user who is using the client. It MUST be same as the Session Initiation Protocol (SIP) Uniform Resource Identifier (URI) for the authenticated user, as specified in [RFC3261] section 19.1, without the "sip:" prefix.

### 3.1.4.1.6 Groups

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

### 3.1.4.1.7 Attribute Groups

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

None.

3.1.6 Other Local Events

None.

3.2 Web Ticket Service Server Details

The Web Ticket Service issues Web tickets using its IssueToken operation, which follows the protocol described in [WS-Trust1.3], except where indicated in section 3.2.4.1.1.1 and section 3.2.4.1.1.2.

Clients MUST authenticate to the Web Ticket Service using one of the following authentication protocols:

- **Integrated Windows authentication**
- OCS-signed certificate authentication
- Live ID authentication
- OAuth2 authentication

Integrated Windows authentication follows the **Kerberos** and the **NT LAN Manager (NTLM) Authentication Protocol**, as specified in [RFC4559]. If Integrated Windows authentication fails, the errors defined in section 3.2.4.1 are returned.

**Certificate (2)** authentication signed by a user agent server (UAS) follows SOAP Message Security 1.1, as specified in [WSS], to validate an **X.509 security token**, as specified in [WSSX509TP]. If OCS-signed certificate (2) authentication fails, the errors defined in section 3.2.4.1 are returned. The certificate signed by the UAS can be obtained from the Certificate Provisioning Service described in section 3.1 of this document.

The Live ID token is presented as a **Security Assertion Markup Language (SAML)** token, as specified in [SAMLCore], and verified using SOAP Message Security 1.1, as specified in [WSS]. The way in which the client retrieves the SAML token is out of the scope of this document. The type of Live ID environment for which the server is configured is specified in the Web service metadata as MSWebAuthentication policy assertion. See section 2.2.4.2 for MSWebAuthentication policy assertion schema. If Live ID authentication fails, the errors defined in section 3.2.4.1 are returned.

The OAuth2 authentication follows the OAuth 2.0 Authorization Protocol described in [IETF-DRAFT-OAuth2.0] with Extensions described in [MS-OAUTH2EX]. The protocol server extracts the OAuth2 token from the Authorization header of the HTTP request and validates that:

- the token carries an actor token that was issued by the Authorization Server that protocol server trusts;
- the actor token is signed by a certificate associated with the Authorization Server that issued the token;
- the actor token nameid (name identifier) claim value matches the issuer claim in the token;
- both the token itself and actor token carry audience claim with a value in the following format: 00000004-0000-0ff1-ce00-0000000000000000@<host_fqdn>@<realm>, where:
  - 00000004-0000-0ff1-ce00-000000000000 is identifier associated with the protocol server described in the document,
- `<host_fqdn>` is a placeholder which represents the **fully qualified domain name (FQDN)** of the protocol server,

- `<realm>` is a place holder which represents a realm value configured for the protocol server;

- the token carries at least one of the following claims: nameid (name identifier), smtp (e-mail address), sip (SIP address) and values in these claims match corresponding values of exactly one user in the UAS database.

If validation of OAuth2 token fails, the errors defined in section 3.2.4.1 are returned.

**Sending the Web Ticket as Credentials to a Web Service Web Application**

After the client receives a Web ticket from the Web Ticket Service, the client MUST attach the Web ticket, as it would a SAML token, to its requests to a participating Web service.

If the Web ticket fails validation, **OCSDiagnosticsFaults**, as described in section 2.2.4.1, SHOULD be returned. The following table describes the relevant **OCSDiagnosticsFaults**.

<table>
<thead>
<tr>
<th>faultcode</th>
<th>ErrorId</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsse:InvalidSecurityToken</td>
<td>28032</td>
<td>The Web ticket is invalid.</td>
</tr>
<tr>
<td>wsse:InvalidSecurityToken</td>
<td>28033</td>
<td>The Web ticket has expired.</td>
</tr>
<tr>
<td>wsse:InvalidSecurityToken</td>
<td>28034</td>
<td>Proof Web tickets are only valid at the same Web server where they were requested.</td>
</tr>
</tbody>
</table>

The Web service MAY also return faults specified in [WSSE 1.0].

The Web ticket can be sent as a signed security token or a proof-of-possession token, as specified in [WS-Trust1.3].

**Sending the Web Ticket as Credentials to a Non-Web Service Web Application**

After the client receives a Web ticket from the Web Ticket Service, the client MUST send the Web ticket in an HTTP header extension in its request to participating non-Web services.

```
X-MS-WebTicket = ticket-data "(";" ticket-extns)
ticket-data = "opaque" "=" base64-ticket
base64-ticket = 1*(ALPHA / DIGIT / "+" / "/") ; base-64 encoded SAML token
ticket-extns: 1*(ALPHA / DIGIT / "+" / "/") ; "=" 1*(ALPHA / DIGIT / "+")
```

The Web ticket, or SAML token, used to construct the **base64-ticket** MUST be a signed security token, as specified in [WS-Trust1.3].

If the Web ticket fails validation, an error response MUST be returned with an HTTP extension header called **X-Ms-diagnostics**, as described in section 3.2.4.1. The following table describes the relevant fault codes.

<table>
<thead>
<tr>
<th>Faultcode</th>
<th>ErrorId</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsse:InvalidSecurityToken</td>
<td>28032</td>
<td>The Web ticket is invalid.</td>
</tr>
<tr>
<td>wsse:InvalidSecurityToken</td>
<td>28033</td>
<td>The Web ticket has expired.</td>
</tr>
</tbody>
</table>
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.

The Web Ticket Service SHOULD keep the following states:

**Fully Qualified Domain Name of the Web Server Farm:** This fully qualified domain name (FQDN) is used to verify the address in the `wst:RequestSecurityToken/wsp:AppliesTo/wsa10:EndpointReference/wsa10:Address` element of the RST. The logic for determining this FQDN is implementation-dependent.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Message Processing Events and Sequencing Rules

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

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IssueToken</td>
<td>Provides a Web ticket given one of the following credentials:</td>
</tr>
<tr>
<td></td>
<td>▪ Integrated Windows authentication</td>
</tr>
<tr>
<td></td>
<td>▪ Live ID</td>
</tr>
<tr>
<td></td>
<td>▪ A certificate (2) signed by a UAS.</td>
</tr>
<tr>
<td></td>
<td>The operation is at the Web Ticket Service.</td>
</tr>
</tbody>
</table>

3.2.4.1 IssueToken

The IssueToken interface provides an operation that returns a Web ticket for a client.

```xml
<wSDL:portType name="IWebTicketService">
  <wSDL:operation name="IssueToken">
  </wSDL:operation>
</wSDL:portType>
```
If there is an error while processing the credentials of the user, then depending on the authentication type used, the response message contains the error details in a custom HTTP header or in a SOAP fault.

**HTTP X-Ms-diagnostics Header**

The X-Ms-diagnostics header is an HTTP header that is returned if Integrated Windows authentication or certificate (2) authentication signed by the UAS fails at the Web Ticket Service for the reasons in this section. The header has the following format:

```
X-Ms-diagnostics = errorId ;" source ";" reason ";" fault
errorId = 1*DIGIT
source = DQUOTE 1*(ALPHA / DIGIT / ";" / ";" / ";" / ";") DQUOTE
; Fully qualified domain name of server
token = DQUOTE 1*( ALPHANUM / ";" / ";" / ";" / ";") DQUOTE
fault = DQUOTE 1*(ALPHA) ";" 1*(ALPHA) DQUOTE
```

The HTTP response code and the details of the X-Ms-diagnostics header are described later for each authentication type.

The following table lists Integrated Windows authentication errors.

<table>
<thead>
<tr>
<th>Type of error</th>
<th>Response code</th>
<th>ErrorId</th>
<th>token</th>
<th>faultcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>The user was authenticated but could not be found in the UAS database.</td>
<td>403</td>
<td>28000</td>
<td>User is not SIP enabled.</td>
<td>wsse:FailedAuthentication</td>
</tr>
<tr>
<td>Some unexpected error occurred in the system.</td>
<td>500</td>
<td>28001</td>
<td>Internal error while processing Integrated Windows authentication or authorization.</td>
<td>wsse:FailedAuthentication</td>
</tr>
</tbody>
</table>

**SOAP Faults**

The following OCSDiagnosticsFaults, as defined in section 2.2.4.1, are returned for Live ID authentication failures, OCS-signed certificate (2) failures, or if there are internal errors processing the RST after Integrated Windows authentication or certificate (2) credentials signed by the UAS are successfully verified. The following table lists SOAP errors.

<table>
<thead>
<tr>
<th>faultcode</th>
<th>ErrorId</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsse:SecurityTokenUnavailable</td>
<td>28028</td>
<td>The Live ID token encryption key cannot be resolved. Check that the token is obtained for this site in the appropriate Live ID environment.</td>
</tr>
<tr>
<td>wsse:SecurityTokenUnavailable</td>
<td>28017</td>
<td>The Live ID token signing key cannot be resolved. Check that the token is obtained from the appropriate Live ID environment.</td>
</tr>
<tr>
<td>wsse:UnsupportedSecurityToken</td>
<td>28018</td>
<td>The Live ID token was produced with the incorrect site policy.</td>
</tr>
<tr>
<td>wsse:FailedAuthentication</td>
<td>28019</td>
<td>The Live ID token identity is not associated with a user account.</td>
</tr>
<tr>
<td>wsse:InvalidSecurity</td>
<td>28020</td>
<td>There is no valid security token.</td>
</tr>
<tr>
<td>wsse:UnsupportedSecurityTokenType</td>
<td>28021</td>
<td>The security token type is unsupported.</td>
</tr>
<tr>
<td>wsse:InvalidSecurityToken</td>
<td>28022</td>
<td>There is no valid subject statement.</td>
</tr>
<tr>
<td>wsse:InvalidSecurity</td>
<td>28023</td>
<td>There is no valid message security.</td>
</tr>
<tr>
<td>wsse:FailedAuthentication</td>
<td>28024</td>
<td>Authentication failed.</td>
</tr>
</tbody>
</table>
The "key cannot be resolved" errors above indicate that protocol server could not locate the key referenced in the token in local or remote stores that it knows about. The "incorrect site policy" error above indicates that Live ID token presented to the protocol server was constructed using policy that the server does not understand.

The following table lists certificate (2) authentication errors while processing the contents of a certificate (2) signed by the UAS.

<table>
<thead>
<tr>
<th>faultcode</th>
<th>ErrorId</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsse:FailedAuthentication</td>
<td>28011</td>
<td>The certificate (2) is expired.</td>
</tr>
<tr>
<td>wsse:FailedAuthentication</td>
<td>28012</td>
<td>The certificate (2) is invalid.</td>
</tr>
<tr>
<td>wsse:FailedAuthentication</td>
<td>28013</td>
<td>The certificate (2) is not found.</td>
</tr>
<tr>
<td>wsse:FailedAuthentication</td>
<td>28014</td>
<td>The user was not found when queried in the database.</td>
</tr>
<tr>
<td>wsse:FailedAuthentication</td>
<td>28015</td>
<td>There was an internal error while processing a certificate (2) authentication or authorization provided by the UAS.</td>
</tr>
</tbody>
</table>

The following table lists internal failures that occur after Integrated Windows authentication and UAS certificate (2) credentials are successfully verified.

<table>
<thead>
<tr>
<th>SubCode</th>
<th>ErrorId</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsse:InvalidSecurity</td>
<td>28025</td>
<td>There is no valid security principal.</td>
</tr>
<tr>
<td>wsse:InvalidSecurity</td>
<td>28026</td>
<td>There is no valid security identity.</td>
</tr>
<tr>
<td>wsse:InvalidSecurity</td>
<td>28027</td>
<td>There is no valid message security.</td>
</tr>
</tbody>
</table>

The following table lists failures that occur while processing the RST.

<table>
<thead>
<tr>
<th>SubCode</th>
<th>ErrorId</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>wst:RequestFailed</td>
<td>28035</td>
<td>The SIP URI in the claim type requirements of the Web ticket request does not match the SIP URI associated with the presented credentials.</td>
</tr>
</tbody>
</table>

### 3.2.4.1.1 Messages

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

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tns:IWebTicketService_IssueToken_InputMessage</td>
<td>A request for a token to be issued.</td>
</tr>
<tr>
<td>tns:IWebTicketService_IssueToken_OutputMessage</td>
<td>The response to a request for a token to be issues.</td>
</tr>
</tbody>
</table>

#### 3.2.4.1.1.1 tns:IWebTicketService_IssueToken_InputMessage

The **tns:IWebTicketService_IssueToken_InputMessage** represents the incoming message and is defined as follows.

```xml
<wsdl:message name="IWebTicketService_IssueToken_InputMessage">
  <wsdl:part name="rst" type="q1:MessageBody"
  xmlns:q1="http://schemas.microsoft.com/Message"/>
</wsdl:message>
```

Refer to the **q1:MessageBody** definition in section 3.2.4.1.3.1.
3.2.4.1.2 tns:IWebTicketService_IssueToken_OutputMessage

The tns:IWebTicketService_IssueToken_OutputMessage represents the incoming message and is defined as follows.

```xml
<wsdl:message name="IWebTicketService_IssueToken_OutputMessage">
  <wsdl:part name="IssueTokenResult" type="q2:MessageBody"
xmlns:q2="http://schemas.microsoft.com/Messag"/>
</wsdl:message>
```

Refer to the q2:MessageBody definition in section 3.2.4.3.2.

3.2.4.1.2 Elements

Elements are defined in the XML schema definition (XSD) associated with [WS-Trust1.3].

3.2.4.1.3 Complex Types

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

<table>
<thead>
<tr>
<th>Complex type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1:MessageBody</td>
<td>Describes the type of tns:IWebTicketService_IssueToken_InputMessage.</td>
</tr>
<tr>
<td>q2:MessageBody</td>
<td>Describes the type of tns:IWebTicketService_IssueToken_OutputMessage.</td>
</tr>
<tr>
<td>wst:RequestSecurityTokenMsg</td>
<td>The alternative type of q1:MessageBody, as defined in section 3.2.4.1.1.1.</td>
</tr>
<tr>
<td>wst:RequestSecurityTokenResponseMsg</td>
<td>The alternative type of q2:MessageBody, as defined in section 3.2.4.1.2.</td>
</tr>
</tbody>
</table>

Other complex types are defined in the XSD associated with [WS-Trust1.3].

3.2.4.1.3.1 q1:MessageBody

The q1:MessageBody type is defined as follows:

```xml
<xs:complexType name="MessageBody">
  <xs:sequence>
    <xs:any namespace="##any" maxOccurs="unbounded" minOccurs="0"/>
  </xs:sequence>
</xs:schema>
```

3.2.4.1.3.2 q2:MessageBody

Refer to the q1:MessageBody definition in section 3.2.4.1.3.1.

3.2.4.1.3.3 wst:RequestSecurityTokenMsg

The wst:RequestSecurityTokenMsg is the alternative type of q1:MessageBody, and is defined in [WS-Trust1.3], with the exception that only the following elements need to be in the message:
The **wst:RequestSecurityTokenMsg** is an incoming message, and is defined in [WS-Trust1.3], with the exception that only the following elements need to be in the message:

/wst:RequestSecurityToken/@Context: A required attribute that MUST be set to a universally unique identifier (UUID).

/wst:RequestSecurityToken/wst:TokenType: A required element that MUST be set to "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1".

/wst:RequestSecurityToken/wst:RequestType: A required element that MUST be set to "http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue".

/wst:RequestSecurityToken/wsp:AppliesTo/wsa:EndpointReference/wsa:Address: A required element that MUST be set to the HTTP URL of the service for which the token is being requested. For example, the element could be set to the HTTP URL of the Certificate Provisioning Web Service. The server MUST validate that this address is part of the server farm.

/wst:RequestSecurityToken/wst:Entropy/wst:BinarySecret: This required element specifies a base64 encoded sequence of cryptographically random octets representing the requestor's entropy. The key size MUST be obtained from the WS-Policy, as specified in [MS-WSPOL], for the Web Ticket Service and SHOULD NOT be less than 128 bits. The entropy size SHOULD be the same size as the key size.

/wst:RequestSecurityToken/wst:KeyType: A required element that MUST be set to "http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey".


/wst:RequestSecurityToken/wst:Claims/auth:ClaimType: An optional element, as specified in [WSFederation], representing a specific claim type. If this element is present, its **Uri** attribute MUST be set to "http://schemas.xmlsoap.org/ws/2005/05/identity/claims(uri)."

/wst:RequestSecurityToken/wst:Claims/auth:ClaimType/auth:Value: An optional element, as specified in [WSFederation], representing the SIP URI of the user for which the Web ticket will be created. If this element is included, the SIP URI MUST match the credentials submitted with the RST. If the element is not included, the server SHOULD use the credentials submitted with the RST to determine the SIP URI. If the SIP URI does not match the credentials, the server SHOULD respond with a fault message carrying fault code **wst:RequestFailed** as described in section 3.2.4.1.

If any one of the above required elements is not supplied or the element syntax does not conform to the syntax requirement specified in this section, the server SHOULD respond with a fault message carrying fault code **wst:InvalidRequest** as described in Section 3 of [WS-Trust1.3].

### 3.2.4.1.3.4  wst:RequestSecurityTokenResponseMsg

The **wst:RequestSecurityTokenResponseMsg** is the alternative type of **q2:MessageBody**, and is defined in [WS-Trust1.3], with the exception that only the following elements need be in the message:

/wst:RequestSecurityTokenResponse/@Context: A required attribute that MUST be set to the value from the corresponding request.

/wst:RequestSecurityTokenResponse/wst:TokenType: A required element that MUST be set to "http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1".

/wst:RequestedSecurityToken/saml:Assertion: A required element that MUST be returned. This element and its contents SHOULD be treated as an opaque XML token by the User Agent.
3.2.4.1.4 Simple Types

Simple types are defined in the XSD associated with [WS-Trust1.3].

3.2.4.1.5 Attributes

Attributes are defined in the XSD associated with [WS-Trust1.3].

3.2.4.1.6 Groups

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

3.2.4.1.7 Attribute Groups

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

3.2.5 Timer Events

None.

3.2.6 Other Local Events

None.

3.3 Authentication Broker Service Server Details

The Authentication Broker Service requires a session to be created using CreateAuthBrokerSession (as specified in section 3.3.4.1) in order to provide the TLS implementation data for authentication.
with the **SIP server**. The service requires a valid **Web Ticket** which can be obtained using the Web Ticket Service (section 3.2). The client is also required to provide a list of client-supported hash algorithms. The response from **CreateAuthBrokerSession** contains the **SessionId** that will be used for remaining requests, as well as the server-supported hash algorithms.

**AuthBrokerAcquireCredential** (as specified in section 3.3.4.3) is called by the client in order to acquire a valid certificate for the user. This is passed the **SessionId** and the **SIPInstance**. The server will need to acquire a new certificate from the Certificate Provisioning Service (section 3.1) or locate a previously obtained certificate.

Once the above two calls are completed, the client will then initiate authentication with the SIP server. When the TLS protocol implementation is required to generate responses, the client will make a call to **AuthBrokerNegotiateSecurityAssociation** (as specified in section 3.3.4.4), passing the target and gssapi-data provided by the server to generate the gssapi-data required for the response.

There are three conditions under which this call will function:

- **client_hello** – **AuthBrokerNegotiateSecurityAssociation** will generate an **SA** and a client_hello handshake message when no **input** element is provided. The result is encoded using the base64 algorithm, and returned in the response.

- **gssapi-data challenge** – The server locates the SA that it created for the client_hello response, decodes the value of the "input" parameter using the base64 algorithm, and passes it along with the security context information stored in the SA to the TLS implementation. The server obtains or locates a previously obtained **certificate (1)** and calls the TLS implementation to generate an output token that carries the TLS certificate, client_key_exchange, certificate_very, change_cipher, _spec, and finished handshake messages. The server then encodes the TLS token and returns it to the protocol client as part of the response.

- **Finished_handshake** – The server locates the SA that it created for the client_hello response, decodes the value of the "input" parameter using the base64 algorithm and passes it, along with security context information stored in the SA, to the TLS implementation for validation. Once information is validated, the server computes, or derives, client and server authentication keys as described in **[MS-SIPAE]** section 3.2.5.1.

After the client is done with the requests, the session is terminated by calling **TerminateAuthBrokerSession** (section 3.3.4.2).

### 3.3.1 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 Authentication Broker Service SHOULD keep the following states:

- The session identifier of the user.
- A certificate store that can be used to save and retrieve certificates
- The store of the challenge that is associated with the first **AuthBrokerNegotiateSecurityAssociation** (section 3.3.4.4) that is used to generate the response for subsequent calls.
3.3.2 Timers

The server keeps track of the session using a session expiration timer that automatically terminates the session after a period of inactivity.

3.3.3 Initialization

None.

3.3.4 Message Processing Events and Sequencing Rules

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

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateAuthBrokerSession</td>
<td>Creates a session to be used as part of the client authentication with the SIP server using TLS-DSK.</td>
</tr>
<tr>
<td>TerminateAuthBrokerSession</td>
<td>Ends the client session with Authentication Broker Service.</td>
</tr>
<tr>
<td>AuthBrokerAcquireCredential</td>
<td>Associates a specific SIPInstance with the session. This can be called once per session.</td>
</tr>
<tr>
<td>AuthBrokerNegotiateSecurityAssociation</td>
<td>Provides gssapi response data for challenges issues by the SIP server. Also provides client and server authentication keys once the finish handshake is received. This can be called multiple times per session.</td>
</tr>
</tbody>
</table>

3.3.4.1 CreateAuthBrokerSession

Creates a session to be used as part of the client authentication with the SIP server using TLS-DSK.

```xml
<wsdl:operation name="CreateAuthBrokerSession">
  <wsdl:input wsaw:Action="http://tempuri.org/IAuthBroker/CreateAuthBrokerSession"
message="tns:IAuthBroker_CreateAuthBrokerSession_InputMessage" />
  <wsdl:output wsaw:Action="http://tempuri.org/IAuthBroker/CreateAuthBrokerSessionResponse"
message="tns:IAuthBroker_CreateAuthBrokerSession_OutputMessage" />
</wsdl:operation>
```

3.3.4.1.1 Messages

The following table summarizes the XML schema message definitions that are specific to this operation.
## 3.3.4.1.1  tns:IAuthBroker_CreateAuthBrokerSession_InputMessage

The request **WSDL message** for the **CreateAuthBrokerSession** WSDL operation.

```xml
<wsdl:message name="IAuthBroker_CreateAuthBrokerSession_Inpu
tMessage">
  <wsdl:part name="parameters" element="tns:CreateAuthBrokerSession" />
</wsdl:message>
```

**CreateAuthBrokerSession**: Refers to the **CreateAuthBrokerSession** definition in section 3.3.4.1.1.1.

## 3.3.4.1.2  tns:IAuthBroker_CreateAuthBrokerSession_OutputMessage

The response **WSDL message** for the **CreateAuthBrokerSession** WSDL operation.

```xml
<wsdl:message name="IAuthBroker_CreateAuthBrokerSession_OutputMessage">
  <wsdl:part name="parameters" element="tns:CreateAuthBrokerSessionResponse" />
</wsdl:message>
```

**CreateAuthBrokerSessionResponse**: Refers to the **CreateAuthBrokerSessionResponse** definition in section 3.3.4.1.2.1.

### 3.3.4.1.2 Elements

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

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tns:CreateAuthBrokerSession</td>
<td>Container for the client request to create a session.</td>
</tr>
<tr>
<td>tns:CreateAuthBrokerSessionResponse</td>
<td>Container for the response to a request to create a session.</td>
</tr>
</tbody>
</table>

## 3.3.4.1.2.1  tns:CreateAuthBrokerSession

The container for the client request to **CreateAuthBrokerSession**.

```xml
<xs:element name="CreateAuthBrokerSession">
  - <xs:complexType>
    - <xs:sequence>
      <xs:element minOccurs="0" name="supportedHashAlgorithms" nillable="true"
                   type="q5:ArrayOfstring" xmlns:q5="http://schemas.microsoft.com/2003/10/Serialization/Arrays" />
    </xs:sequence>
</xs:element>
```
**supportedHashAlgorithms**: An array of the supported hash algorithms made by the requestor.

### 3.3.4.1.2 `tns:CreateAuthBrokerSessionResponse`

The container for the response to a request to `CreateAuthBrokerSession`.

```xml
<xs:element name="CreateAuthBrokerSessionResponse">
  - <xs:complexType>
  - <xs:sequence>
  </xs:sequence>
</xs:complexType>
</xs:element>
```

**CreateAuthBrokerSessionResult**: The result of the request. The type is defined in section 3.3.4.1.3.1.

### 3.3.4.1.3 Complex Types

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

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tns:CreateAuthBrokerSessionResponse</code></td>
<td>Describes the server response for creating a new session.</td>
</tr>
</tbody>
</table>

### 3.3.4.1.3.1 `tns:CreateAuthBrokerSessionResponse`

Describes the server response for creating a new session.

```xml
<xs:complexType name="CreateAuthBrokerSessionResponse">
  <xs:sequence>
    <xs:element minOccurs="0" name="HashAlgorithm" nillable="true" type="xs:string" />
    <xs:element minOccurs="0" name="SessionId" nillable="true" type="xs:string" />
  </xs:sequence>
</xs:complexType>
```

**HashAlgorithm**: The hash algorithm that will be used for the session. It is determined based on the `supportedHashAlgorithms` provided by the caller.

**SessionId**: A value that is unique to the session.

### 3.3.4.1.4 Simple Types

None.
3.3.4.1.5 Attributes
None.

3.3.4.1.6 Groups
None.

3.3.4.1.7 Attribute Groups
None.

3.3.4.2 TerminateAuthBrokerSession
Ends the client session with Authentication Broker Service.

```xml
<wsdl:operation name="TerminateAuthBrokerSession">
  <wsdl:input wsaw:Action="http://tempuri.org/IAuthBroker/TerminateAuthBrokerSession"
    message="tns:IAuthBroker_TerminateAuthBrokerSession_InputMessage" />
  <wsdl:output
    wsaw:Action="http://tempuri.org/IAuthBroker/TerminateAuthBrokerSessionResponse"
    message="tns:IAuthBroker_TerminateAuthBrokerSession_OutputMessage" />
</wsdl:operation>
```

3.3.4.2.1 Messages
The following table summarizes the XML schema message definitions that are specific to this operation.

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tns:IAuthBroker_TerminateAuthBrokerSession_InputMessage</code></td>
<td>The request for <code>TerminateAuthBrokerSession</code>.</td>
</tr>
</tbody>
</table>

3.3.4.2.1.1 `tns:IAuthBroker_TerminateAuthBrokerSession_InputMessage`
The request WSDL message for the `TerminateAuthBrokerSession` WSDL operation.

```xml
<wsdl:message name="IAuthBroker_TerminateAuthBrokerSession_InputMessage">
  <wsdl:part name="parameters" element="tns:TerminateAuthBrokerSession" />
</wsdl:message>
```

3.3.4.2.1.2 `tns:IAuthBroker_TerminateAuthBrokerSession_OutputMessage`
The response WSDL message for the `TerminateAuthBrokerSession` WSDL operation.

```xml
<wsdl:message name="IAuthBroker_TerminateAuthBrokerSession_OutputMessage">
  <wsdl:part name="parameters" element="tns:TerminateAuthBrokerSessionResponse" />
</wsdl:message>
```
3.3.4.2.2 Elements

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

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tns:TerminateAuthBrokerSession</td>
<td>Container for the client request to TerminateAuthBrokerSession.</td>
</tr>
<tr>
<td>tns:TerminateAuthBrokerSessionResponse</td>
<td>Container for the response to the client request to TerminateAuthBrokerSession.</td>
</tr>
</tbody>
</table>

3.3.4.2.2.1 tns:TerminateAuthBrokerSession

The container for the client request to TerminateAuthBrokerSession.

```xml
<xs:element name="TerminateAuthBrokerSession">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="sessionID" nillable="true" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

**sessionID**: The SessionId that was returned from CreateAuthBrokerSessionResponse.

3.3.4.2.2.2 tns:TerminateAuthBrokerSessionResponse

The container for the response to the client request to TerminateAuthBrokerSession.

```xml
<xs:element name="TerminateAuthBrokerSessionResponse">
  <xs:complexType>
    <xs:sequence />
  </xs:complexType>
</xs:element>
```

3.3.4.2.3 Complex Types

None.

3.3.4.2.4 Simple Types

None.

3.3.4.2.5 Attributes
None.

### 3.3.4.2.6 Groups

None.

### 3.3.4.2.7 Attribute Groups

None.

### 3.3.4.3 AuthBrokerAcquireCredential

Associates a specific **SIPInstance** with the session.

```xml
<wsdl:operation name="AuthBrokerAcquireCredential">
  <wsdl:input wsaw:Action="http://tempuri.org/IAuthBroker/AuthBrokerAcquireCredential"
    message="tns:IAuthBroker_AuthBrokerAcquireCredential_InputMessage" />
  <wsdl:output
    wsaw:Action="http://tempuri.org/IAuthBroker/AuthBrokerAcquireCredentialResponse"
    message="tns:IAuthBroker_AuthBrokerAcquireCredential_OutputMessage" />
</wsdl:operation>
```

#### 3.3.4.3.1 Messages

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

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tns:IAuthBroker_AuthBrokerAcquireCredential_InputMessage</td>
<td>The request for AuthBrokerAcquireCredential.</td>
</tr>
<tr>
<td>tns:IAuthBroker_AuthBrokerAcquireCredential_OutputMessage</td>
<td>The response for AuthBrokerAcquireCredential.</td>
</tr>
</tbody>
</table>

#### 3.3.4.3.1.1 tns:IAuthBroker_AuthBrokerAcquireCredential_InputMessage

The request **WSDL message** for the AuthBrokerAcquireCredential **WSDL operation**.

```xml
<wsdl:message name="IAuthBroker_AuthBrokerAcquireCredential_InputMessage">
  <wsdl:part name="parameters" element="tns:AuthBrokerAcquireCredential" />
</wsdl:message>
```

#### 3.3.4.3.1.2 tns:IAuthBroker_AuthBrokerAcquireCredential_OutputMessage

The response **WSDL message** for the AuthBrokerAcquireCredential **WSDL operation**.

```xml
<wsdl:message name="IAuthBroker_AuthBrokerAcquireCredential_OutputMessage">
  <wsdl:part name="parameters" element="tns:AuthBrokerAcquireCredentialResponse" />
</wsdl:message>
```

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

<table>
<thead>
<tr>
<th>Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tns:AuthBrokerAcquireCredential</td>
<td>Container for the client request to AuthBrokerAcquireCredential.</td>
</tr>
<tr>
<td>tns:AuthBrokerAcquireCredentialResponse</td>
<td>Container for the response to the client request to AuthBrokerAcquireCredential.</td>
</tr>
</tbody>
</table>

#### 3.3.4.3.2.1 tns:AuthBrokerAcquireCredential

The container for the client request to AuthBrokerAcquireCredential.

```xml
<xs:element name="AuthBrokerAcquireCredential">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="sessionid" nillable="true" type="xs:string" />
      <xs:element minOccurs="0" name="sipInstance" nillable="true" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

**sessionid**: The SessionId that was returned from CreateAuthBrokerSessionResponse.

**sipInstance**: The SIPInstance that uniquely identifies the endpoint, as defined in [MS-SIPRE] section 4.2.

#### 3.3.4.3.2.2 tns:AuthBrokerAcquireCredentialResponse

The container for the response to the client request to AuthBrokerAcquireCredential.

```xml
<xs:element name="AuthBrokerAcquireCredentialResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="AuthBrokerAcquireCredentialResult" type="xs:long" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

**AuthBrokerAcquireCredentialResult**: The remaining lifetime, in seconds, of the certificate on the server on the server. The value will be zero if the certificate has expired or if obtaining the certificate failed.

### 3.3.4.3.3 Complex Types

None.

### 3.3.4.3.4 Simple Types

None.

### 3.3.4.3.5 Attributes
3.3.4.3.6 Groups
None.

3.3.4.3.7 Attribute Groups
None.

3.3.4.4 AuthBrokerNegotiateSecurityAssociation
Provides gssapi response data for challenges issues by the SIP server and client and server authentication keys once the handshake is complete.

```xml
<wsdl:operation name="AuthBrokerNegotiateSecurityAssociation">
  <wsdl:input
      wsaw:Action="http://tempuri.org/IAuthBroker/AuthBrokerNegotiateSecurityAssociation"
      message="tns:IAuthBroker_AuthBrokerNegotiateSecurityAssociation_InputMessage" />
  <wsdl:output
      wsaw:Action="http://tempuri.org/IAuthBroker/AuthBrokerNegotiateSecurityAssociationResponse"
      message="tns:IAuthBroker_AuthBrokerNegotiateSecurityAssociation_OutputMessage" />
</wsdl:operation>
```

3.3.4.4.1 Messages
The following table summarizes the XML schema message definitions that are specific to this operation.

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
</table>

3.3.4.4.1.1 tns:IAuthBroker_AuthBrokerNegotiateSecurityAssociation_InputMessage
The request WSDL message for the AuthBrokerNegotiateSecurityAssociation WSDL operation.

```xml
<wsdl:message name="IAuthBroker_AuthBrokerNegotiateSecurityAssociation_InputMessage">
  <wsdl:part name="parameters" element="tns:AuthBrokerNegotiateSecurityAssociation" />
</wsdl:message>
```

3.3.4.4.1.2 tns:IAuthBroker_AuthBrokerNegotiateSecurityAssociation_OutputMessage

The response **WSDL message** for the **AuthBrokerNegotiateSecurityAssociation WSDL operation**.

```xml
<wsdl:message name="IAuthBroker_AuthBrokerNegotiateSecurityAssociation_OutputMessage">
    <wsdl:part name="parameters" element="tns:AuthBrokerNegotiateSecurityAssociationResponse" />
</wsdl:message>
```

### 3.3.4.4.2 Elements

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

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
</table>

#### 3.3.4.4.2.1 AuthBrokerNegotiateSecurityAssociation

The container for the client request to AuthBrokerNegotiateSecurityAssociation.

```xml
<xs:element name="AuthBrokerNegotiateSecurityAssociation">
    <xs:complexType>
        <xs:sequence>
            <xs:element minOccurs="0" name="sessionid" nillable="true" type="xs:string" />
            <xs:element minOccurs="0" name="target" nillable="true" type="xs:string" />
            <xs:element minOccurs="0" name="input" nillable="true" type="xs:string" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

**sessionid**: The **SessionId** that was returned from **CreateAuthBrokerSessionResponse**.

**target**: The targetname, as specified in [MS-SIPAE] section 2.2.1, contained in the response from the SIP server.

**input**: The value of the gssapi-data, as specified in [MS-SIPAE] section 2.2.1, contained in the response from the SIP server. Do not set if this is the first message of the handshake.

#### 3.3.4.4.2.2 AuthBrokerNegotiateSecurityAssociationResponse

The container for the response to the client request to AuthBrokerNegotiateSecurityAssociation.

```xml
<xs:element name="AuthBrokerNegotiateSecurityAssociationResponse">
    <xs:complexType>
        <xs:sequence>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```
**AuthBrokerNegotiateSecurityAssociationResult**: A **NegotiateSaResponse**, as defined in section [3.3.4.4.3.1](#), that describes the server response for the **AuthBrokerNegotiateSecurityAssociation** request.

### 3.3.4.4.3 Complex Types

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

<table>
<thead>
<tr>
<th>Complex Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tns:NegotiateSaResponse</td>
<td>Describes the server response for the <strong>AuthBrokerNegotiateSecurityAssociation</strong> request.</td>
</tr>
<tr>
<td>tns:SAReturnData</td>
<td>Describes the SA return data type.</td>
</tr>
<tr>
<td>tns:AuthReturnValuePair</td>
<td>Describes the base for the SA return data type.</td>
</tr>
</tbody>
</table>

### 3.3.4.4.3.1 tns:NegotiateSaResponse

Describes the server response for the **AuthBrokerNegotiateSecurityAssociation** request.

```xml
<xs:complexType name="NegotiateSaResponse">
  <xs:complexContent mixed="false">
    <xs:extension base="tns:SAReturnData">
      <xs:sequence>
        <xs:element minOccurs="0" name="ClientSigningKey" nillable="true" type="xs:string" />
        <xs:element minOccurs="0" name="ServerSigningKey" nillable="true" type="xs:string" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

For more information on tns:SAReturnData see section [3.3.4.4.3.2](#).

**ClientSigningKey**: The key generated to be used by the client as part of [MS-SIPAE].

**ServerSigningKey**: The key generated to be used by the server as part of [MS-SIPAE].

### 3.3.4.4.3.2 tns:SAReturnData

Describes the SA return data type.

```xml
<xs:complexType name="SAReturnData">
  <xs:complexContent mixed="false">
    <xs:extension base="tns:AuthReturnValuePair">
      <xs:sequence>
        <xs:element minOccurs="0" name="MaxSignature" type="xs:unsignedInt" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
For more information on **tns:AuthReturnValuePair** see section 3.3.4.3.3.

**MaxSignature**: The maximum size of the security token in bytes. This value is 0 unless **ClientSigningKey** and **ServerSigningKey** are populated.

### 3.3.4.4.3.3  tns:AuthReturnValuePair

Describes the base for the **SA** return data type.

```xml
<xs:complexType name="AuthReturnValuePair">
  <xs:sequence>
    <xs:element minOccurs="0" name="OutString" nillable="true" type="xs:string" />
    <xs:element minOccurs="0" name="SecurityStatus" type="xs:int" />
  </xs:sequence>
</xs:complexType>
```

**OutString**: The value of the gssapi-data challenge. For more details on how this is computed see section 3.3.

**SecurityStatus**: The status of the request. Once OK is returned, negotiation is complete and **ClientSigningKey**, **ServerSigningKey**, and **MaxSignature** of **NegotiateSaResponse** will be populated. The table below describes possible values of this field.

#### Success/Informational Values

<table>
<thead>
<tr>
<th>SecurityStatus</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00090312</td>
<td>OK</td>
</tr>
<tr>
<td>0x00090313</td>
<td>ContinueNeeded</td>
</tr>
<tr>
<td>0x00090314</td>
<td>CompAndContinue</td>
</tr>
<tr>
<td>0x00090317</td>
<td>ContentExpired</td>
</tr>
<tr>
<td>0x00090320</td>
<td>CredentialsNeeded</td>
</tr>
<tr>
<td>0x00090321</td>
<td>Renegotiate</td>
</tr>
</tbody>
</table>

#### Error Values

<table>
<thead>
<tr>
<th>SecurityStatus</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x80090300</td>
<td>OutOfMemory</td>
</tr>
<tr>
<td>0x80090301</td>
<td>InvalidHandle</td>
</tr>
<tr>
<td>0x80090302</td>
<td>Unsupported</td>
</tr>
<tr>
<td>0x80090303</td>
<td>TargetUnknown</td>
</tr>
<tr>
<td>0x80090304</td>
<td>InternalError</td>
</tr>
<tr>
<td>0x80090305</td>
<td>PackageNotFound</td>
</tr>
<tr>
<td>0x80090306</td>
<td>NotOwner</td>
</tr>
<tr>
<td>0x80090307</td>
<td>CannotInstall</td>
</tr>
<tr>
<td>SecurityStatus</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>0x80090308</td>
<td>InvalidToken</td>
</tr>
<tr>
<td>0x80090309</td>
<td>CannotPack</td>
</tr>
<tr>
<td>0x8009030A</td>
<td>QopNotSupported</td>
</tr>
<tr>
<td>0x8009030B</td>
<td>NoImpersonation</td>
</tr>
<tr>
<td>0x8009030C</td>
<td>LogonDenied</td>
</tr>
<tr>
<td>0x8009030D</td>
<td>UnknownCredentials</td>
</tr>
<tr>
<td>0x8009030E</td>
<td>NoCredentials</td>
</tr>
<tr>
<td>0x8009030F</td>
<td>MessageAltered</td>
</tr>
<tr>
<td>0x80090310</td>
<td>OutOfSequence</td>
</tr>
<tr>
<td>0x80090311</td>
<td>NoAuthenticatingAuthority</td>
</tr>
<tr>
<td>0x80090318</td>
<td>IncompleteMessage</td>
</tr>
<tr>
<td>0x80090320</td>
<td>IncompleteCredentials</td>
</tr>
<tr>
<td>0x80090321</td>
<td>BufferNotEnough</td>
</tr>
<tr>
<td>0x80090322</td>
<td>WrongPrincipal</td>
</tr>
<tr>
<td>0x80090324</td>
<td>TimeSkew</td>
</tr>
<tr>
<td>0x80090325</td>
<td>UntrustedRoot</td>
</tr>
<tr>
<td>0x80090326</td>
<td>IllegalMessage</td>
</tr>
<tr>
<td>0x80090327</td>
<td>CertUnknown</td>
</tr>
<tr>
<td>0x80090328</td>
<td>CertExpired</td>
</tr>
<tr>
<td>0x80090331</td>
<td>AlgorithmMismatch</td>
</tr>
<tr>
<td>0x80090332</td>
<td>SecurityQosFailed</td>
</tr>
<tr>
<td>0x8009033E</td>
<td>SmartcardLogonRequired</td>
</tr>
<tr>
<td>0x80090343</td>
<td>UnsupportedPreauth</td>
</tr>
</tbody>
</table>

### 3.3.4.4.4 Simple Types
None.

### 3.3.4.4.5 Attributes
None.

### 3.3.4.4.6 Groups
None.
3.3.4.4.7 Attribute Groups
None.

3.3.5 Timer Events
None.

3.3.6 Other Local Events
None.
4 Protocol Examples

4.1 GetAndPublishCert

This section contains an example of a request and response for a GetAndPublishCert operation.

4.1.1 Request

The following example is a request in a GetAndPublishCert operation.

```xml
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <To s:mustUnderstand="1" xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">https://server.contoso.com/CertProv/CertProvisioningService.svc</To>
  </s:Header>
    <GetAndPublishCert DeviceId="{161CCE75-E0C7-5F60-BDD1-054099725B0B}" Entity="alice@contoso.com" xmlns="http://schemas.microsoft.com/OCS/AuthWebServices/">
      <RequestSecurityToken xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
        <TokenType>http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3</TokenType>
        <RequestType>http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-sec-1.0#Basic</RequestType>
          <Subject>
            <X509v3CertificateThumbprint>-----BEGIN CERTIFICATE REQUEST-----MIIDmjCCAwMCAQAwGDEWMBQGA1UEAwwNdGVzdEB0ZXN0LmNvbTCbnzANBgkqhkiG9w0BAQEFAOAOBgcwwwHwYDVR0PBBcgugSm3OaG/n4zDvQ26aXwFv3HbU0zGmH81g6jH9zK5bqM42B2u1eL5jyk5CIA extravagant code snippet

      ------END CERTIFICATE REQUEST-----</X509v3CertificateThumbprint>
      </BinarySecurityToken>
    </RequestSecurityToken>
  </GetAndPublishCert>
</s:Body>
</s:Envelope>
```

-----BEGIN NEW CERTIFICATE REQUEST-----

MIIDmjCCAwMCAQAwGDEWMBQGA1UEAwwNdGVzdEB0ZXN0LmNvbTCbnzANBgkqhkiG9w0BAQEFAOAOBgcwwwHwYDVR0PBBcgugSm3OaG/n4zDvQ26aXwFv3HbU0zGmH81g6jH9zK5bqM42B2u1eL5jyk5CIA extravagant code snippet

------END CERTIFICATE REQUEST-----

-----BEGIN NEW CERTIFICATE REQUEST-----
4.1.2 Response

The following example is a response in a GetAndPublishCert operation.

```xml
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Action s:mustUnderstand="1" xmlns="http://schemas.microsoft.com/wsc/2005/02/addressing/none">
      http://schemas.microsoft.com/OCS/AuthWebServices/GetAndPublishCertResponse
    </Action>
  </s:Header>
    <GetAndPublishCertResponse ResponseClass="Success" DeviceId="{161CCE75-E0C7-5F60-BDD1-054099725B0B}" Entity="alice@contoso.com" xmlns="http://schemas.microsoft.com/OCS/AuthWebServices/">
      <RequestSecurityTokenResponse xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
      <BinarySecurityToken EncryptionType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-security-secect-1.0.xsd#base64binary" />
    </GetAndPublishCertResponse>
  </s:Body>
</s:Envelope>
```
4.2 IssueToken

This section contains an example of a request and response for an IssueToken operation.

4.2.1 Request

The following example is a request in an IssueToken operation.

```xml
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:doc="http://docs.oasis-open.org/wss/2004/01/oauth决胜">
  <s:Body>
    <RequestSecurityToken Context="2fdf3b92-4341-4eeb-b898-44ef4994cd55">
      <TokenType>
        http://docs.oasis-open.org/wss/2004/01/oauth决胜/saml-token-profile-1.1#SAMLV1.1
      </TokenType>
      <RequestType>http://schemas.xmlsoap.org/ws/2005/02/trust/Issue</RequestType>
      <AppliesTo xmlns="http://schemas.xmlsoap.org/ws/2004/09/policy">
        <EndpointReference xmlns="http://www.w3.org/2005/08/addressing">
          <Address>https://pool0.vdomain.com/GroupExpansion/Service.svc</Address>
        </EndpointReference>
      </AppliesTo>
      <Entropy>
        <BinarySecret>pE1grLu4aRHhp9XKcKdS3hnHi+6sXChE3iqPomYgk</BinarySecret>
      </Entropy>
      <KeyType>http://docs.oasis-open.org/wss/2004/01/oauth决胜/saml-token-profile-1.1#SAMLV1.1</KeyType>
    </RequestSecurityToken>
  </s:Body>
</s:Envelope>
```

4.2.2 Response

The following example is a response in an IssueToken operation.

```xml
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:doc="http://docs.oasis-open.org/wss/2004/01/oauth决胜">
  <s:Body>
    <RequestSecurityTokenResponseCollection xmlns="http://docs.oasis-open.org/wss/2004/01/oauth决胜/saml-token-profile-1.1#SAMLV1.1">
      <RequestSecurityTokenResponse Context="2fdf3b92-4341-4eeb-b898-44ef4994cd55">
        <TokenType>
          http://docs.oasis-open.org/wss/2004/01/oauth决胜/saml-token-profile-1.1#SAMLV1.1
        </TokenType>
        <RequestedSecurityToken>
        </RequestedSecurityToken>
      </RequestSecurityTokenResponse>
    </RequestSecurityTokenResponseCollection>
  </s:Body>
</s:Envelope>
```
4.3 CreateAuthBrokerSession

This section contains an example of a request and response for a CreateAuthBrokerSession operation.

4.3.1 Request

The following example is a request in a CreateAuthBrokerSession operation.

```xml
<s:Envelope xmlns:a="http://www.w3.org/2005/08/addressing"
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://tempuri.org/IAuthBroker/CreateAuthBrokerSession</a:Action>
    <a:MessageID>urn:uuid:70de6ed0-5279-44db-956a-84109a5a1a95</a:MessageID>
    <a:To s:mustUnderstand="1">https://webpoolbl20d10.infra.contoso.com/Reach/Sip.svc/AuthBroker</a:To>
    <o:Security xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <Timestamp xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        <Created>2014-08-26T19:35:03.868Z</Created>
        <Expires>2014-08-27T03:11:37.343Z</Expires>
      </Timestamp>
      <saml:Assertion MajorVersion="1" MinorVersion="1" AssertionID="SamlSecurityToken-5b1c2eaf5806-4f8c-840d-e1254ddff134"
Issuer="https://bl20d10fes05.infra.contoso.com:4443/54aae93a-2d06-573b-b07b-7683bal1fc56"
        </saml:Conditions>
      </saml:Assertion>
    </o:Security>
  </s:Header>
  <s:Body>
  </s:Body>
</s:Envelope>
```
<saml:Conditions>
<saml:AuthenticationStatement>
<AuthenticationMethod urn:oasis:names:tc:SAML:1.0:am:unspecified />
<AuthenticationInstant>2014-08-26T19:39:38.343Z</AuthenticationInstant>
</saml:AuthenticationStatement>
</saml:Conditions>
<saml:RestrictionCondition>
<saml:AudienceRestrictionCondition>
<saml:Conditions/>
</saml:AudienceRestrictionCondition>
</saml:RestrictionCondition>
<saml:NameIdentifier Format="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier">sip:user1@contoso.com</saml:NameIdentifier>
</saml:NameIdentifier>
<saml:SubjectConfirmation>
<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
<EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
<EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256"/>
</EncryptedKey>
<KeyName>54aae93a-2d06-573b-b07b-768c3ba1fc56:8d18f6a2ac8b8c8</KeyName>
</KeyInfo>
</saml:SubjectConfirmation>
</saml:Subject>
</saml:AuthenticationStatement>
</saml:Conditions>
</saml:Assertion>
</o:Security>
</s:Body>
</s:Envelope>
4.3.2 Response

The following example is a response in a **CreateAuthBrokerSession** operation.

```xml
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/
 xmlns:a="http://www.w3.org/2005/08/addressing"
 xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://tempuri.org/IAuthBroker/CreateAuthBrokerSessionResponse</a:Action>
    <a:RelatesTo>urn:uuid:70de6ed0-5279-44db-956a-84109a5a95</a:RelatesTo>
  </s:Header>
  <s:Body>
    <CreateAuthBrokerSessionResponse xmlns="http://tempuri.org/">
                                    xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <b:HashAlgorithm>SHA1</b:HashAlgorithm>
        <b:SessionId>79b37b28-26e4-49e0-9df1-2aed7f5c4f2a</b:SessionId>
      </CreateAuthBrokerSessionResult>
    </CreateAuthBrokerSessionResponse>
  </s:Body>
</s:Envelope>
```

4.4 TerminateAuthBrokerSession

This section contains an example of a request and response for a **TerminateAuthBrokerSession** operation.

4.4.1 Request

The following example is a request in a **TerminateAuthBrokerSession** operation.

```xml
<s:Envelope xmlns:a="http://www.w3.org/2005/08/addressing"
 xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://tempuri.org/IAuthBroker/TerminateAuthBrokerSession</a:Action>
    <a:MessageID>urn:uuid:13cd6f39-7fb9-4421-9951-5bf76f0f3547</a:MessageID>
    <a:To s:mustUnderstand="1">https://webpoolbl20d10.infra.contoso.com/Reach/Sip.svc/AuthBroker</a:To>
    <a:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <Timestamp xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        <Created>2014-08-26T19:35:08.088Z</Created>
        <Expires>2014-08-26T19:40:08.088Z</Expires>
      </Timestamp>
      <saml:Assertion MajorVersion="1" MinorVersion="1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
        <Created>2014-08-26T19:35:08.088Z</Created>
        <Expires>2014-08-26T19:40:08.088Z</Expires>
      </saml:Assertion>
    </a:Security>
  </s:Header>
</s:Envelope>
```
    <saml:AudienceRestrictionCondition>
    </saml:AudienceRestrictionCondition>
    </saml:Conditions>
    <saml:Subject>
      <saml:NameIdentifier Format="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/uri">sip:user1@lyncnadbr.com</saml:NameIdentifier>
      <saml:SubjectConfirmation>
        <saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:holder-of-key</saml:ConfirmationMethod>
        <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
          <e:EncryptedKey xmlns:e="http://www.w3.org/2001/04/xmlenc#">
            <e:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256"></e:EncryptionMethod>
            <KeyInfo>
              <KeyName>54aae93a-2d06-573b-b07b-768c3ba1fc56:8d18f6a2ac9b8b8c8</KeyName>
            </KeyInfo>
            <e:CipherData>
              <e:CipherValue>8UIhCyHsgQ4jBTxA5Hj++xCls28GcPgtFE+8sTukXXUpJ3oAvwS3Y+Q==</e:CipherValue>
            </e:CipherData>
          </e:EncryptedKey>
        </KeyInfo>
      </saml:SubjectConfirmation>
    </saml:Subject>
    <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
      <SignedInfo>
        <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></CanonicalizationMethod>
        <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"></SignatureMethod>
        <Reference URI="#SamlSecurityToken-5b1c2ef5-8006-4f8c-840d-e1254d4ff134">
          <Transforms>
            <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"></Transform>
          </Transforms>
          <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"></DigestMethod>
          <DigestValue>wO4at1m6CXhFb6epvW1m12RtgeD5t5G90TQ82k4riu4</DigestValue>
        </Reference>
        <SignatureValue>
          LN+jAhAgKDFX3F1Qkz/vq0PJIXW+i1j0+EliNDhqoz49mP6z2tRnZXxfGbKNIhbwJEdtNgx2EwG
          KccDko2BIZAZw1Ro26IwQYk10hs80Uy4wByVJo9/aLebrNvYHSX1daidpaxUeCunjLrZtG0ULH6ElvY1Myk8
          qoXX5jHH9AM/9bnhX1x34rWj/bbVbhe76oqEe/y4P291E3dZ0y6Kay70UJtZ6C0p047iigdKMKWj0WKKig/2aJ5g
          6a3MBH=MPlQqsKgDRkAAdnJzffFrP5Y1ljjyAlKg94dcrfWVBjZPq6w6RFPLrMVQF5IMXy0XW/8TCZAM+iEn0Q
          QQ==</SignatureValue>
        <KeyInfo>
          <o:SecurityTokenReference>
            <o:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#ThumbprintSHA1">dLYUC56Gpszp8aEwelSagq2YY==</o:KeyIdentifier>
          </o:SecurityTokenReference>
        </KeyInfo>
      </SignedInfo>
    </Signature>
  </s:Assertion>
</o:Security>
</s:Body>
4.4.2 Response

The following example is a response in a `TerminateAuthBrokerSession` operation.

```xml
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:a="http://www.w3.org/2005/08/addressing">
  <s:Header>
    <a:Action
      s:mustUnderstand="1">http://tempuri.org/IAuthBroker/TerminateAuthBrokerSessionResponse</a:Action>
    <a:RelatesTo>urn:uuid:13cd6f39-7fb9-4421-9951-5bf76f0f3547</a:RelatesTo>
  </s:Header>
  <s:Body>
    <TerminateAuthBrokerSessionResponse xmlns="http://tempuri.org/"/>
  </s:Body>
</s:Envelope>
```

4.5 AuthBrokerAcquireCredential

This section contains an example of a request and response for an `AuthBrokerAcquireCredential` operation.

4.5.1 Request

The following example is a request in an `AuthBrokerAcquireCredential` operation.

```xml
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:a="http://www.w3.org/2005/08/addressing">
  <s:Header>
    <a:Action
      s:mustUnderstand="1">http://tempuri.org/IAuthBroker/AuthBrokerAcquireCredential</a:Action>
    <a:MessageID>urn:uuid:8cabe6b3-10e1-437f-adf1-208d0231064e</a:MessageID>
    <a:To
      s:mustUnderstand="1">https://webpoolbl20d10.infra.contoso.com/Reach/Sip.svc/AuthBroker</a:To>
    <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <Timestamp xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        <Created>2014-08-26T19:35:05.322Z</Created>
        <Expires>2014-08-26T19:40:05.322Z</Expires>
      </Timestamp>
    <saml:Assertion
      MajorVersion="1" MinorVersion="1" AssertionID="SamlSecurityToken-5b1c2eaf-5806-4f8c-840d-e1254d4f134"
      Issuer="https://bl20d10fes05.infra.contoso.com:4443/54aae93a-2d06-573b-b07b-768c3b1fc56"
      IssueInstant="2014-08-26T19:39:38.343Z"
      xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
      <saml:Conditions
        <saml:AudienceRestrictionCondition/>
        <saml:Condition>
          <saml:Audience>
            https://webpoolbl20d10.infra.contoso.com</saml:Audience>
          <saml:AudienceRestrictionCondition/>
        </saml:Condition>
      </saml:Conditions>
    </saml:Assertion>
  </o:Security>
</s:Envelope>
```
<saml:AuthenticationStatement
  AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:unspecified"
  AuthenticationInstant="2014-08-26T19:39:38.343Z">
  <saml:Subject>
    <saml:NameIdentifier>
      <saml:NameIdentifier Format="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/uri">sip:user1@lyncnadbr.com</saml:NameIdentifier>
    </saml:NameIdentifier>
    <saml:SubjectConfirmation>
      <saml:ConfirmationMethod>
        <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
          <e:EncryptedKey xmlns:e="http://www.w3.org/2001/04/xmlenc#">
            <e:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256"/>
            <KeyInfo>
              <KeyName>54aae93a-2d06-573b-b07b-7683d3af6d4a2ac8b8c8</KeyName>
            </KeyInfo>
            <CipherData>
              <CipherValue>EUIhCyHsgQ4jBTxASHJ9+xCtS28GcPgtFE+8sTukXUpJ30aWWS3Y+Q==</CipherValue>
            </CipherData>
          </e:EncryptedKey>
        </KeyInfo>
        <KeyInfo>
          <o:SecurityTokenReference>
            <o:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#ThumbprintSHA1">diYUC56GspzpizzEaeweSAqzdYY==</o:KeyIdentifier>
          </o:SecurityTokenReference>
        </KeyInfo>
      </saml:ConfirmationMethod>
    </saml:SubjectConfirmation>
  </saml:Subject>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      <Reference URI="#SamlSecurityToken-5b1c2eaf-5806-4f8c-840d-e1254d4f134">
        <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
        <DigestValue>wO4at1m6CXhFb6epvWlm2RtgeDt5G090TQ82k4riu4==</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>
      LW+jhAgKDFKX3f1Qks/vgQPJiWp1jO+EliNhqyo40mPe6ItRnZXKfGbKNlwbJE1tNgyX2EwG
      KccDk2o2BIZAawHrwO26iWQY01hc80t4wY3Jxu9/ADbBrN7SIxidla1pahvUicEcjkJLT/GZUHkL6Elq1Myk8
      gqXXzjH9kAN/9nhnX1kzj24nJ/bV5nQ167fOe/4yte1Jd620y6kay7OHJzaca1o+50ri1dkVXkKwWk1g/2a75g
      63hMjxMP1Qa1kGdTR4AdnxJfFnp5y1juyAlKqy9gdcrfwVGBJzFqw6nRPNLrMvFQ5MYyOxW/8TcZAM+efnOq
      QQ==</SignatureValue>
    </Signature>
  </Signature>
</saml:AuthenticationStatement>

<AuthBrokerAcquireCredential xmlns="http://tempuri.org/">
  <sessionid>79b37b2b-26e4-49e0-9df1-2ade7f5c4f2a</sessionid>
  <sipInstance>71F297D2-197D-545B-AA97-5701FSB5134D</sipInstance>
</AuthBrokerAcquireCredential>
4.5.2 Response

The following example is a response in an **AuthBrokerAcquireCredential** operation.

```xml
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:a="http://www.w3.org/2005/08/addressing">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://tempuri.org/IAuthBroker/AuthBrokerAcquireCredentialResponse</a:Action>
    <a:RelatesTo>urn:uuid:8cabe6b3-10e1-437f-adf1-208d0231064e</a:RelatesTo>
  </s:Header>
  <s:Body>
    <AuthBrokerAcquireCredentialResponse xmlns="http://tempuri.org/">
      <AuthBrokerAcquireCredentialResult>9898</AuthBrokerAcquireCredentialResult>
    </AuthBrokerAcquireCredentialResponse>
  </s:Body>
</s:Envelope>
```

4.6 AuthBrokerNegotiateSecurityAssociation

This section contains an example of a request and response for an **AuthBrokerNegotiateSecurityAssociation** operation.

4.6.1 Request

The following example is a request in an **AuthBrokerNegotiateSecurityAssociation** operation.

```xml
<s:Envelope xmlns:a="http://www.w3.org/2005/08/addressing"
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://tempuri.org/IAuthBroker/AuthBrokerNegotiateSecurityAssociation</a:Action>
    <a:MessageID>urn:uuid:3e89dde2-fa4f-4246-b965-07b0c453b5a2</a:MessageID>
    <a:To s:mustUnderstand="1">https://webpoolbl20d10.infra.contoso.com/Reach/Sip.svc/AuthBroker</a:To>
    <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <Timestamp xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        <Created>2014-08-26T19:35:07.994Z</Created>
        <Expires>2014-08-26T19:40:07.994Z</Expires>
      </Timestamp>
      <saml:Assertion MajorVersion="1" MinorVersion="1" AssertionID="SamlSecurityToken-5b1c2eaf-5806-4f8c-840d-e1254d4ff134"
Issuer="https://bl20d10fes05.infra.contoso.com:4443/54aae93a-2d06-573b-b07b-768c3ba1fc56"
          <saml:AudienceRestrictionCondition>
          </saml:AudienceRestrictionCondition>
          <saml:NameIdentifier Format="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier">sip:user1@lyncnadbr.com</saml:NameIdentifier>
        </saml:Conditions>
      </saml:Assertion>
    </o:Security>
  </s:Header>
</s:Envelope>
```
<saml:SubjectConfirmation>
  <saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:holder-of-key</saml:ConfirmationMethod>
  <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
    <e:EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
      <e:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes256"></e:EncryptionMethod>
      <KeyInfo>
        <KeyInfo>
          <KeyName>54aae93a-2d06-573b-b07b-768c3ba1fc56:8d18f6a2ac8b8c8</KeyName>
        </KeyInfo>
      </e:EncryptedKey>
    </KeyInfo>
  </saml:SubjectConfirmation>
  </saml:Subject>
  <saml:AuthenticationStatement>
    <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
      <SignedInfo>
        <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></CanonicalizationMethod>
        <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"></SignatureMethod>
        <Reference URI="#SamlSecurityToken-5b1c2eaf-5806-4f8c-840d-e125d4dfff134">
          <Transforms>
            <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></Transform>
            <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></Transform>
          </Transforms>
          <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"></DigestMethod>
          <DigestValue>wO4at1m6CXhFb6epvW1mi2RtgeD5G9O9TQ82k4riu4</DigestValue>
        </Reference>
      </SignedInfo>
      <SignatureValue>LW+jhAgKDFKX3FlQkz/vq0PJiWp+1jO+EL1N6hqoz49mP6ZtRnZXxfGbKNIhbwJE1tNgx2EWGKccDko2BIZAzw1Ro261/wQYX01Ho8U04wByVJJu9/sAEbrNyTHSIXidlaDabxU1cEcjKLT/GZUHL6ElqV1Myk8qgXXzjHd0A9N/9bhnX1kj24Nj/bbVbji767Boe/y4PZSlbJdeZoY6Kay7QHJtZacPo77iQidKVKKj0WKKiq/2aJSG6a3MW/MPflaIKgUDmT4GAdnxxffFnp5Y1jyy4gdcKrv6VBgJzFqw6nRPNLrMVFQ5IMXY0XW/8TCZAM+efnoQQQ</SignatureValue>
      <KeyInfo>
        <o:SecurityTokenReference>
          <o:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-1.1#ThumbprintSHA1">diYUC56Gpspzzp8aEwelSRqzdYy</o:KeyIdentifier>
        </o:SecurityTokenReference>
      </KeyInfo>
    </Signature>
  </saml:AuthenticationStatement>
  <s:Header>
    <s:Body>
      <AuthBrokerNegotiateSecurityAssociation xmlns="http://tempuri.org/">
        <sessionid>79b37b28-26e4-49e0-9df1-2aed7f5c4f2a</sessionid>
        <target>BL20D10FES04.infra.contoso.com</target>
      </AuthBrokerNegotiateSecurityAssociation>
    </s:Body>
  </s:Header>
</s:Envelope>
4.6.2 Response

The following example is a response in an AuthBrokerNegotiateSecurityAssociation operation.

```xml
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/
xmlns:a="http://www.w3.org/2005/08/addressing">
  <s:Header>
    <a:Action s:mustUnderstand="1">http://tempuri.org/IAuthBroker/AuthBrokerNegotiateSecurityAssociationResponse</a:Action>
    <a:RelatesTo>urn:uuid:3e89dde2-4a5f-4246-b965-07b0c450b5a2</a:RelatesTo>
  </s:Header>
  <s:Body>
    <AuthBrokerNegotiateSecurityAssociationResponse xmlns="http://tempuri.org/">
        <b:OutString/>
        <b:SecurityStatus>0</b:SecurityStatus>
        <b:MaxSignature>20</b:MaxSignature>
        <b:ClientSigningKey>xCl4+dGsjmcx9YNqp7JXryLQqc0vpieoWUolJ4Mc+IY=/</b:ClientSigningKey>
        <b:ServerSigningKey>HPIGUkkObg7/fxq+vlF66ks2bwt7zäh43t1DHiz/56b0=/</b:ServerSigningKey>
      </AuthBrokerNegotiateSecurityAssociationResult>
    </AuthBrokerNegotiateSecurityAssociationResponse>
  </s:Body>
</s:Envelope>
```
5 Security

5.1 Security Considerations for Implementers
None.

5.2 Index of Security Parameters
None.
6   Appendix A: Full WSDL

<table>
<thead>
<tr>
<th>WSDL name</th>
<th>Prefix</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://schemas.microsoft.com/OCS/AuthWebServices/">http://schemas.microsoft.com/OCS/AuthWebServices/</a></td>
<td>tns</td>
<td>section 6.1</td>
</tr>
<tr>
<td><a href="http://tempuri.org/">http://tempuri.org/</a></td>
<td>tns</td>
<td>section 6.2</td>
</tr>
<tr>
<td><a href="http://tempuri.org/">http://tempuri.org/</a></td>
<td>tns</td>
<td>section 6.3</td>
</tr>
</tbody>
</table>

For ease of implementation, the full WSDLs are provided in the following sections.

6.1 Certificate Provisioning Service WSDL

```xml
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions
 xmlns:wsdl="http://schemas.xmlsoap.org/wsd1/"
 xmlns:tns="http://schemas.microsoft.com/OCS/AuthWebServices/"
 xmlns:xm="http://www.w3.org/2001/XMLSchema"
 xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512/"
targetNamespace="http://schemas.microsoft.com/OCS/AuthWebServices/">

<wsdl:types>
 <xs:schema id="ocsauth"
targetNamespace="http://schemas.microsoft.com/OCS/AuthWebServices/"
elementFormDefault="qualified">
  <xs:import namespace="http://docs.oasis-open.org/ws-sx/ws-trust/200512/
  schemaLocation="http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3.xsd" />
  <xs:simpleType name="ResponseClassType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Success" />
      <xs:enumeration value="Warning" />
      <xs:enumeration value="Error" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="ErrorInfoType">
    <xs:sequence>
      <xs:element name="Description" type="xs:string" minOccurs="0" maxOccurs="1" />
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="GetAndPublishCert" type="tns:GetAndPublishCertType" />
  <xs:complexType name="GetAndPublishCertType">
    <xs:sequence>
      <xs:element ref="wst:RequestSecurityToken" minOccurs="1" maxOccurs="1" />
      <xs:element ref="wst:IssueSecurityToken" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
</wsdl:types>
</wsdl:definitions>
```

[MS-OCAUTHWS] - v20190319
OC Authentication Web Service Protocol
Copyright © 2019 Microsoft Corporation
Release: March 19, 2019
6.2 Web Ticket Service WSDL

<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions name="WebTicketService" targetNamespace="http://tempuri.org/">
  <wsdl:documentation>
    <msc:Documentation>
      <msc:body>
        OC Authentication Web Service Protocol
      </msc:body>
    </msc:Documentation>
  </wsdl:documentation>

  <wsdl:types>
    <wsdl:message name="GetWebTicketResponse"/>
    <wsdl:message name="GetWebTicketRequest"/>

    <wsdl:message name="NegotiateAuthenticationResponse">
    </wsdl:message>
    <wsdl:message name="NegotiateAuthenticationRequest">
    </wsdl:message>

    <wsdl:messages>
      <wsdl:message name="GetWebTicket">
        <wsdl:part name="Body" element="GetWebTicketRequest"/>
        <wsdl:part name="Response" element="GetWebTicketResponse"/>
      </wsdl:message>
    </wsdl:messages>

    <wsdl:bindings>
      <wsdl:binding name="WebTicketService Binding" type="http://tempuri.org/GetWebTicket">
        <wsdl:operation name="GetWebTicket" message="GetWebTicket" />
      </wsdl:binding>
    </wsdl:bindings>

    <wsdl:portTypes>
      <wsdl:portType name="WebTicketService">
        <wsdl:op name="GetWebTicket" message="GetWebTicket"/>
        <wsdl:op name="NegotiateAuthentication" message="NegotiateAuthentication"/>
      </wsdl:portType>
    </wsdl:portTypes>

    <wsdl:services>
      <wsdl:service name="WebTicketService">
        <wsdl:operation name="GetWebTicket" message="GetWebTicket" />
        <wsdl:operation name="NegotiateAuthentication" message="NegotiateAuthentication"/>
      </wsdl:service>
    </wsdl:services>
  </wsdl:types>
</wsdl:definitions>
<sp:X509Token
sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/AlwaysToRecipient"
>
  <wsp:Policy
  >
  <sp:RequireThumbprintReference/>
  <sp:X509Token
  >
  <sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/AlwaysToRecipient"
  >
  </sp:IncludeToken>
  </sp:X509Token>
  </sp:Policy>
</wsp:Policy>
</sp:X509Token>
</wsp:Policy>
</sp:SignedParts>
</sp:EndorsingSupportingTokens>
  <wsp:Policy>
    <sp:MustSupportRefKeyIdentifier/>
    <sp:MustSupportRefIssuerSerial/>
    <sp:MustSupportRefThumbprint/>
    <sp:MustSupportRefEncryptedKey/>
  </wsp:Policy>
</sp:Wss11>
  <wsp:Policy>
    <sp:MustSupportIssuedTokens/>
    <sp:RequireClientEntropy/>
    <sp:RequireServerEntropy/>
  </wsp:Policy>
</sp:Trust10>
<wsaw:UsingAddressing/>
</wsp:All>
</wsp:ExactlyOne>
</wsp:Policy>
<wsp:Policy wsu:Id="WebTicketServicePin_policy">
  <wsp:ExactlyOne>
    <wsp:All>
      <http:BasicAuthentication
        <wsp:Policy>
          <sp:TransportToken>
            <wsp:Policy
            >
            <sp:HttpsToken RequireClientCertificate="false"/>
            </wsp:Policy>
          </sp:TransportToken>
          <sp:AlgorithmSuite>
            <wsp:Policy
            >
            <sp:Basic256/>
            </wsp:Policy>
          </sp:AlgorithmSuite>
          <sp:Layout>
            <wsp:Policy
            >
            <sp:Strict/>
            </wsp:Policy>
          </sp:Layout>
          </sp:TransportBinding>
        </wsp:Policy>
      </sp:TransportBinding>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
<wsp:Policy wsu:Id="WebTicketServiceAuth_policy">
  <wsp:ExactlyOne>
    <wsp:All>
      <af:FormsAuthentication
        <wsp:Policy

<sp:TransportToken>
   <wsp:Policy>
      <sp:HttpsToken RequireClientCertificate="false"/>
   </wsp:Policy>
</sp:TransportToken>
<sp:AlgorithmSuite>
   <wsp:Policy>
      <sp:Basic256/>
   </wsp:Policy>
</sp:AlgorithmSuite>
<sp:Layout>
   <wsp:Policy>
      <sp:Lax/>
   </wsp:Policy>
</sp:Layout>
<sp:IncludeTimestamp/>
</wsp:Policy>
</sp:TransportBinding>
<sp:SignedSupportingTokens
   xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
   <wsp:Policy>
      <sp:UsernameToken
         sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">
         <wsp:Policy>
            <wssUsernameToken10/>
         </wsp:Policy>
      </sp:UsernameToken>
   </wsp:Policy>
</sp:SignedSupportingTokens>
   <wsp:Policy>
      <sp:MustSupportRefKeyIdentifier/>
      <sp:MustSupportRefIssuerSerial/>
   </wsp:Policy>
</sp:Wss10>
</wsp:Policy>
</wsp:All>
</wsp:ExactlyOne>
</wsp:Policy>
<sp:WssUsernameToken10/>
</wsp:Policy>
</sp:UsernameToken>
</wsp:Policy>
</sp:SignedSupportingTokens>
<sp:TransportBinding
   xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
   <wsp:Policy>
      <sp:TransportToken>
         <wsp:Policy>
            <sp:HttpsToken RequireClientCertificate="false"/>
         </wsp:Policy>
      </sp:TransportToken>
   </wsp:Policy>
</sp:TransportBinding>
<sp:SignedSupportingTokens
   xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
   <wsp:Policy>
<sp:UsernameToken
    sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">
    <wsp:Policy>
        <sp:WssUsernameToken/>
    </wsp:Policy>
</sp:UsernameToken>

<wsp:Policy>
    <sp:SignedSupportingTokens>
    </sp:SignedSupportingTokens>
</wsp:Policy>

<xsd:schema targetNamespace="http://tempuri.org/Imports">
    <xsd:import
        schemaLocation="https://server.vdomain.com/WebTicket/WebTicketService.svc/mex?xsd=xsd0"
        namespace="http://schemas.microsoft.com/Message"/>
</xsd:schema>

<wsdl:types>
    <wsdl:message name="IWebTicketService_IssueToken_InputMessage">
        <wsdl:part name="rst" type="q1:MessageBody" xmlns:q1="http://schemas.microsoft.com/Message"/>
    </wsdl:message>
    <wsdl:message name="IWebTicketService_IssueToken_OutputMessage">
        <wsdl:part name="IssueTokenResult" type="q2:MessageBody" xmlns:q2="http://schemas.microsoft.com/Message"/>
    </wsdl:message>
    <wsdl:portType name="IWebTicketService">
        <wsdl:operation name="IssueToken">
                <wsdl:input/>
                <wsdl:output/>
            </soap:operation>
        </wsdl:operation>
    </wsdl:portType>
</wsdl:types>
<wsdl:binding name="WebTicketServicePin" type="tns:IWebTicketService">
  <soap:PolicyReference URI="#WebTicketServicePin_policy"/>
  <soap:operation name="IssueToken">
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </soap:operation>
</wsdl:binding>

<wsdl:binding name="WebTicketServiceAuth" type="tns:IWebTicketService">
  <soap:PolicyReference URI="#WebTicketServiceAuth_policy"/>
  <soap:operation name="IssueToken">
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </soap:operation>
</wsdl:binding>

<wsdl:binding name="WebTicketServiceAnon" type="tns:IWebTicketService">
  <soap:PolicyReference URI="#WebTicketServiceAnon_policy"/>
  <soap:operation name="IssueToken">
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </soap:operation>
</wsdl:binding>
</wsdl:definitions>

6.3 Authentication Broker Service WSDL

<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions name="RemoteService" targetNamespace="http://tempuri.org/">
  <wsdl:service name="WebTicketService" />
  <wsdl:binding name="WSSecurityWebTicketService" type="tns:RemoteService">
    <soap:PolicyReference URI="#WSSecurityWebTicketService_policy"/>
    <soap:operation name="IssueToken">
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </soap:operation>
  </wsdl:binding>
</wsdl:definitions>
<sp:TransportBinding xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
<sp:TransportToken />
<spHttpsToken />
<sp:TransportToken>
<sp:AlgorithmSuite>
<sp:Policy>
<sp:Basic256 />
</sp:Policy>
</sp:AlgorithmSuite>
<sp:Layout>
<sp:Policy>
<sp:Strict />
</sp:Policy>
</sp:Layout>
</sp:TransportPolicy>
<sp:SignedSupportingTokens xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
<sp:Policy>
<sp:RequireInternalReference />
</sp:Policy>
</sp:SignedSupportingTokens>
<sp:Wss11 xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
<sp:Policy />
</sp:Wss11>
<sp:Trust13 xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
<sp:Policy />
</sp:Trust13>
<sp:MustSupportIssuedTokens />
<sp:RequireClientEntropy />
</sp:Policy>
</sp:TransportBinding>
<sp:RequireServerEntropy />
</wsp:Policy>
<wsaw:UsingAddressing />
</wsp:ExactlyOne>
</wsp:Policy>
<wsdl:message name="IAuthBroker_CreateAuthBrokerSession_InputMessage">
    <wsdl:part name="parameters" element="tns:CreateAuthBrokerSession" />
</wsdl:message>
<wsdl:message name="IAuthBroker_CreateAuthBrokerSession_OutputMessage">
    <wsdl:part name="parameters" element="tns:CreateAuthBrokerSessionResponse" />
</wsdl:message>
<wsdl:message name="IAuthBroker_TerminateAuthBrokerSession_InputMessage">
    <wsdl:part name="parameters" element="tns:TerminateAuthBrokerSession" />
</wsdl:message>
<wsdl:message name="IAuthBroker_TerminateAuthBrokerSession_OutputMessage">
    <wsdl:part name="parameters" element="tns:TerminateAuthBrokerSessionResponse" />
</wsdl:message>
<wsdl:message name="IAuthBroker_AuthBrokerAcquireCredential_InputMessage">
    <wsdl:part name="parameters" element="tns:AuthBrokerAcquireCredential" />
</wsdl:message>
<wsdl:message name="IAuthBroker_AuthBrokerAcquireCredential_OutputMessage">
    <wsdl:part name="parameters" element="tns:AuthBrokerAcquireCredentialResponse" />
</wsdl:message>
<wsdl:message name="IAuthBroker_AuthBrokerNegotiateSecurityAssociation_InputMessage">
    <wsdl:part name="parameters" element="tns:AuthBrokerNegotiateSecurityAssociation" />
</wsdl:message>
<wsdl:message name="IAuthBroker_AuthBrokerNegotiateSecurityAssociation_OutputMessage">
    <wsdl:part name="parameters" element="tns:AuthBrokerNegotiateSecurityAssociationResponse" />
</wsdl:message>
<wsdl:portType name="IAuthBroker">
    <wsdl:operation name="CreateAuthBrokerSession">
    </wsdl:operation>
    <wsdl:operation name="TerminateAuthBrokerSession">
    </wsdl:operation>
    <wsdl:operation name="AuthBrokerAcquireCredential">
    </wsdl:operation>
    <wsdl:operation name="AuthBrokerNegotiateSecurityAssociation">
    </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="WS2007FedHttpBinding_WebTicketBearerTokenAuth_IAuthBroker" type="tns:IAuthBroker">
</wsdl:binding>
</wsdl:service>
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
<soap:operation name="CreateAuthBrokerSession" style="document" />
<soap:input />
</wsdl:port>
<wsdl:port name="WS2007FedHttpBinding WebTicketBearerTokenAuth_IAuthBroker"
binding="tns:WS2007FedHttpBinding WebTicketBearerTokenAuth_IAuthBroker">
<soap:address location="https://999dtk5150we2.exchange.corp.microsoft.com/Reach/Sip.svc/AuthBroker" />
<wsa10:EndpointReference>
</wsa10:EndpointReference>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
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 updates to those products.

- 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
- Microsoft Skype for Business 2019
- Microsoft Skype for Business Server 2019

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates 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.
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

A

Abstract data model
- certificate provisioning service 24
- server (section 3.1.1 24, section 3.2.1 33, section
  3.3.1 39)
- web ticket service 33

Applicability 18

Attribute groups 23

Attributes 23

AuthBrokerAcquireCredential operation
example 61
- request 61
- response 63

AuthBrokerNegotiateSecurityAssociation operation
example 63
- request 63
- response 65

AuthBrokerNegotiateSecurityAssociation operation
example 63
- request 63
- response 65

GetAndPublishCert operation 53
- request 53
- response 53

IssueToken operation 55
- request 55
- response 55

TerminateAuthBrokerSession operation 59
- request 59
- response 61

C

Capability negotiation 18

Certificate provisioning service
- abstract data model 24
- full WSDL 67
- initialization 24
- local events 31

message processing
- GetAndPublishCert operation 25
- overview 16
- sequencing rules 24
- server 24
- timer events 31
- timers 24

Change tracking 79

Complex types 20
- af:BindingType 22
- af:MSWebAuthenticationType 21
- af:OCSDiagnosticsFaultType 20
- tns:ErrorInfoType 22

CreateAuthBrokerSession operation
example 57
- request 57
- response 59

D

Data model - abstract
- certificate provisioning service 24
- server (section 3.1.1 24, section 3.2.1 33, section
  3.3.1 39)
- web ticket service 33

E

Events
- local - server (section 3.1.6 31, section 3.2.6 38,
  section 3.6 52)

Examples
- AuthBrokerAcquireCredential operation 61
  - request 61
  - response 63
- AuthBrokerNegotiateSecurityAssociation operation
  - request 63
  - response 65
- CreateAuthBrokerSession operation 57
  - request 57
  - response 59
- GetAndPublishCert operation 53
  - request 53
  - response 53
- IssueToken operation 55
  - request 55
  - response 55
- TerminateAuthBrokerSession operation 59
  - request 59
  - response 61

F

Fields - vendor-extensible 18

Full WSDL 67
- Authentication Broker Service WSDL 73
- Certificate Provisioning Service 67
- Certificate Provisioning Service WSDL 67
- Web Ticket Service 68
- Web Ticket Service WSDL 68

G

GetAndPublishCert operation 25
- example 53
- request 53
- response 53

Glossary 8

Groups 23

I

Implemeneter - security considerations 66

Index of security parameters 66

Informative references 13

Initialization
- certificate provisioning service 24
- server (section 3.1.3 24, section 3.2.3 33, section
  3.3.5 52)
- web ticket service 33

Introduction 8

IssueToken operation 33
- example 55
- request 55
- response 55

L

Local events
- certificate provisioning service 31