[MS-DTAG]:

Device Trust Agreement Protocol

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.

Copyright © 2017 Microsoft Corporation

Revision Summary

Date	Revision History	Revision Class	Comments
11/6/2009	0.1	Major	First Release.
12/18/2009	0.1.1	Editorial	Changed language and formatting in the technical content.
1/29/2010	0.2	Minor	Clarified the meaning of the technical content.
3/12/2010	0.2.1	Editorial	Changed language and formatting in the technical content.
4/23/2010	1.0	Major	Updated and revised the technical content.
6/4/2010	1.0.1	Editorial	Changed language and formatting in the technical content.
7/16/2010	2.0	Major	Updated and revised the technical content.
8/27/2010	2.0	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2010	2.0	None	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	2.0	None	No changes to the meaning, language, or formatting of the technical content.
1/7/2011	2.0	None	No changes to the meaning, language, or formatting of the technical content.
2/11/2011	2.0	None	No changes to the meaning, language, or formatting of the technical content.
3/25/2011	2.0	None	No changes to the meaning, language, or formatting of the technical content.
5/6/2011	2.0	None	No changes to the meaning, language, or formatting of the technical content.
6/17/2011	2.1	Minor	Clarified the meaning of the technical content.
9/23/2011	2.1	None	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	3.0	Major	Updated and revised the technical content.
3/30/2012	3.0	None	No changes to the meaning, language, or formatting of the technical content.
7/12/2012	3.0	None	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	3.0	None	No changes to the meaning, language, or formatting of the technical content.
1/31/2013	3.0	None	No changes to the meaning, language, or formatting of the technical content.
8/8/2013	4.0	Major	Updated and revised the technical content.
11/14/2013	4.1	Minor	Clarified the meaning of the technical content.
2/13/2014	4.1	None	No changes to the meaning, language, or formatting of the technical content.

Copyright © 2017 Microsoft Corporation

Date	Revision History	Revision Class	Comments
5/15/2014	4.1	None	No changes to the meaning, language, or formatting of the technical content.
6/30/2015	4.1	None	No changes to the meaning, language, or formatting of the technical content.
10/16/2015	4.1	None	No changes to the meaning, language, or formatting of the technical content.
7/14/2016	4.1	None	No changes to the meaning, language, or formatting of the technical content.
6/1/2017	4.1	None	No changes to the meaning, language, or formatting of the technical content.

Copyright © 2017 Microsoft Corporation

Table of Contents

Copyright © 2017 Microsoft Corporation

Copyright © 2017 Microsoft Corporation

Copyright © 2017 Microsoft Corporation

1 Introduction

This document specifies the Device Trust Agreement Protocol (DTAG).

DTAG enables two UPnP endpoints to securely exchange certificates over an unsecure network and to establish a trust relationship by means of a simple, one-time shared secret.

DTAG is compliant with UPnP architecture and is implemented as a UPnP service [UPNPARCH1]. Therefore, this protocol does not have a specific WSDL declaration.

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:

action: A command exposed by a **service** which takes one or more input or output arguments and which may have a return value. For more information, see [UPNPARCH1.1] sections 2 and 3.

authenticator: A large value (160 bits), which is generated from the payload, a shared secret, and a nonce; and which 1) reveals nothing of the payload, shared secret, or nonce; and 2) is impractical to generate from any other payload, shared secret, or nonce.

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: 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.

control point: A control point retrieves device and service descriptions, sends actions to **services**, polls for **service** state variables, and receives events from services.

device: A logical device and/or a container that can embed other logical devices and that embeds one or more services and advertises its presence on network(s). For more information, see [UPNPARCH1.1] sections 1 and 2.

endpoint: In the context of a web service, a network target to which a **SOAP** message can be addressed. See [WSADDR].

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

nonce: A number that is used only once. This is typically implemented as a random number large enough that the probability of number reuse is extremely small. A nonce is used in authentication protocols to prevent replay attacks. For more information, see [RFC2617].

one-time password (OTP): A simple secret shared by two **endpoints** and delivered out-of-band by some means outside of the Device Trust Agreement Protocol (typically, via user input).

Copyright © 2017 Microsoft Corporation

service: A logical functional unit that represents the smallest units of control and that exposes actions and models the state of a physical device with state variables. For more information, see [UPNPARCH1.1] section 3.

service description: A formal definition of a logical **service**, expressed in the **UPnP** Template language and written in **XML** syntax. A **service description** is specified by a **UPnP** vendor by filling in any placeholders in a **UPnP** Service Template (was SCPD). For more information, see [UPNPARCH1.1] section 2.6.

service type: Denoted by "urn:schemas-upnp-org:service:" followed by a unique name assigned by a **UPnP** forum working committee, a colon, and an integer version number. A **service type** has a one-to-one relationship with **UPnP** Service Templates. **UPnP** vendors can specify additional **services**; these are denoted by "urn:domain-name:service: " followed by a unique name assigned by the vendor, a colon, and a version number, where domain-name is a Vendor Domain Name. For more information, see [UPNPARCH1.1] section 2.

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

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 action: The HTTP request header field used to indicate the intent of the **SOAP** request, using a URI value. See [SOAP1.1] section 6.1.1 for more information.

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

SOAP 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.

state variable: A single facet of a model of a physical **service** that is exposed by a **service** and which has a name, data type, optional default value, optional constraints values, and which can trigger events when its value changes. For more information, see [UPNPARCH1.1] sections 2 and 3.

Universal Plug and Play (UPnP): A set of computer network protocols, published by the UPnP Forum [UPnP], that allow devices to connect seamlessly and that simplify the implementation of networks in home (data sharing, communications, and entertainment) and corporate environments. UPnP achieves this by defining and publishing UPnP device control protocols built upon open, Internet-based communication standards.

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

Copyright © 2017 Microsoft Corporation

UTF-8: A byte-oriented standard for encoding Unicode characters, defined in the Unicode standard. Unless specified otherwise, this term refers to the UTF-8 encoding form specified in [UNICODE5.0.0/2007] section 3.9.

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.

XML: The Extensible Markup Language, as described in [XML1.0].

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 (XSD): A language that defines the elements, attributes, namespaces, and data types for **XML** documents as defined by [XMLSCHEMA1/2] and [W3C-XSD] standards. An XML schema uses **XML** syntax for its language.

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.

[RFC2104] Krawczyk, H., Bellare, M., and Canetti, R., "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997, http://www.ietf.org/rfc/rfc2104.txt

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

[RFC3629] Yergeau, F., "UTF-8, A Transformation Format of ISO 10646", STD 63, RFC 3629, November 2003, http://www.ietf.org/rfc/rfc3629.txt

[SOAP1.1] Box, D., Ehnebuske, D., Kakivaya, G., et al., "Simple Object Access Protocol (SOAP) 1.1", W3C Note, May 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[UPNPARCH1.1] UPnP Forum, "UPnP Device Architecture 1.1", October 2008, http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf

[UPNPARCH1] UPnP Forum, "UPnP Device Architecture 1.0", October 2008, http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0.pdf

Copyright © 2017 Microsoft Corporation

[WSASB] Gudgin, M., Hadley, M., and Rogers, T., Eds., "Web Services Addressing 1.0 - SOAP Binding", W3C Recommendation, May 2006, http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/

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

[XMLNS-2ED] World Wide Web Consortium, "Namespaces in XML 1.0 (Second Edition)", August 2006, http://www.w3.org/TR/2006/REC-xml-names-20060816/

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

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

1.2.2 Informative References

[MSDN-XDR] Microsoft Corporation, "XDR Schema Data Types Reference", http://msdn.microsoft.com/en-us/library/ms256049(v=VS.85).aspx

1.3 Overview

A common method for establishing a trust relationship between one **device** and another unknown device is for the devices to exchange and verify each other's **certificate**. However, if the devices are connected over an unsecure network, the success of this method is challenged by the fact that the exchanged information can be exposed to a third party or could even be tampered with. DTAG is designed to ensure the integrity of the **SOAP message** and to enable the establishment of a trust relationship between networked devices by means of a simple, one-time shared secret. The shared secret, called a **one-time password (OTP)**, is transferred in an out-of-band manner, such as through user interaction.

DTAG is implemented as a **UPnP service** consisting of four actions that are performed in the following order:

1. **Exchange**: The two **endpoints** exchange certificates and endpoint identifiers.
2. **Commit**, then **Validate**: The two endpoints perform a series of authentications based on the OTP, the OTP substrings, the endpoint identifiers, and the certificates.
3. **Confirm**: The two endpoints finalize the trust agreement process and store each other's certificate in secure storage.

Each **action** results in a pair of **SOAP** request and response messages in the network, as specified in [UPNPARCH1.1] section 3.1.1. The following diagram illustrates the flow of DTAG messages between the devices and **control points** until the trust agreement is established successfully.

Copyright © 2017 Microsoft Corporation

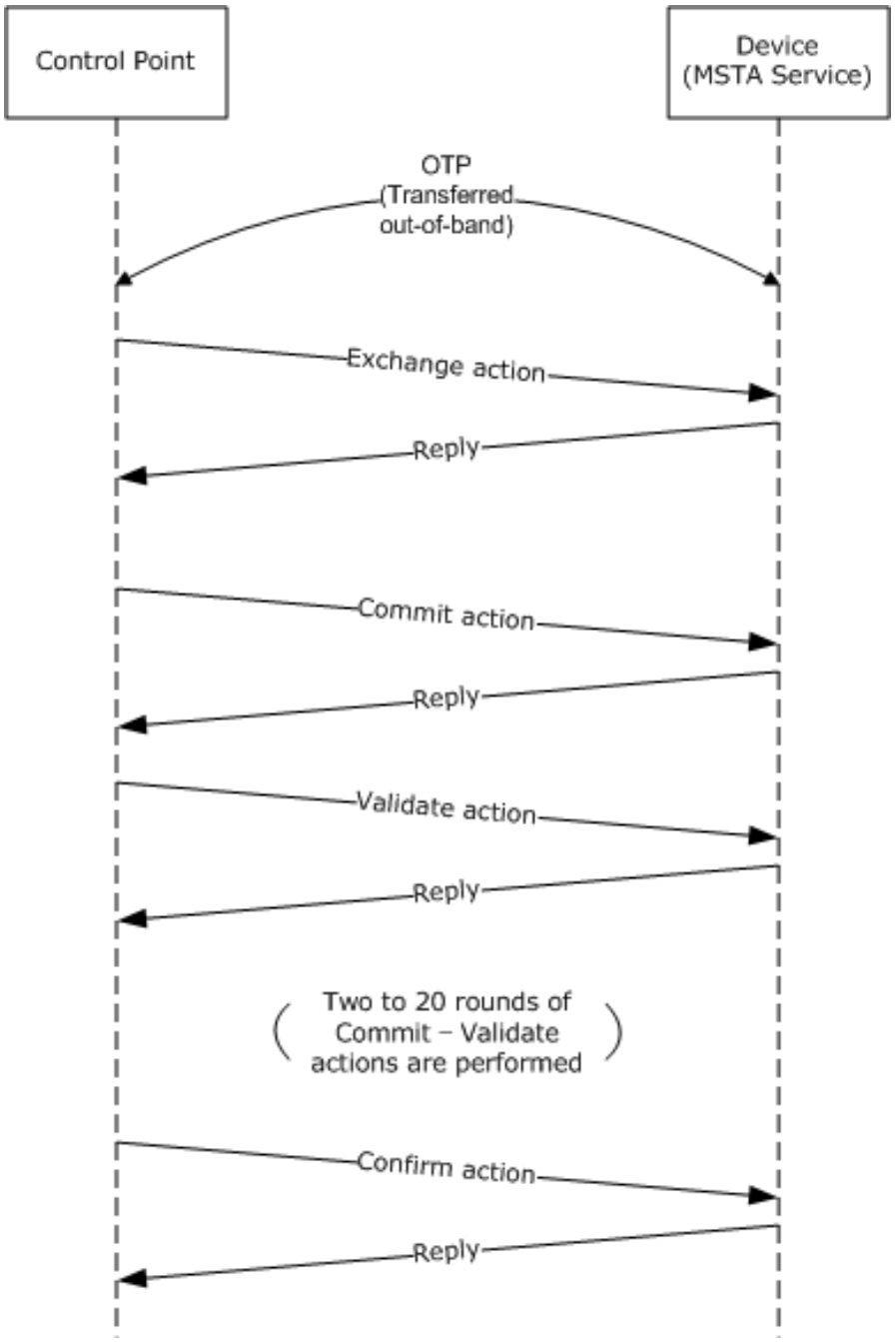

Figure 1: DTAG message sequence to establish trust agreement

1.4 Relationship to Other Protocols

DTAG is a **UPnP service** over **SOAP**/HTTP as shown in the following diagram:

Copyright © 2017 Microsoft Corporation

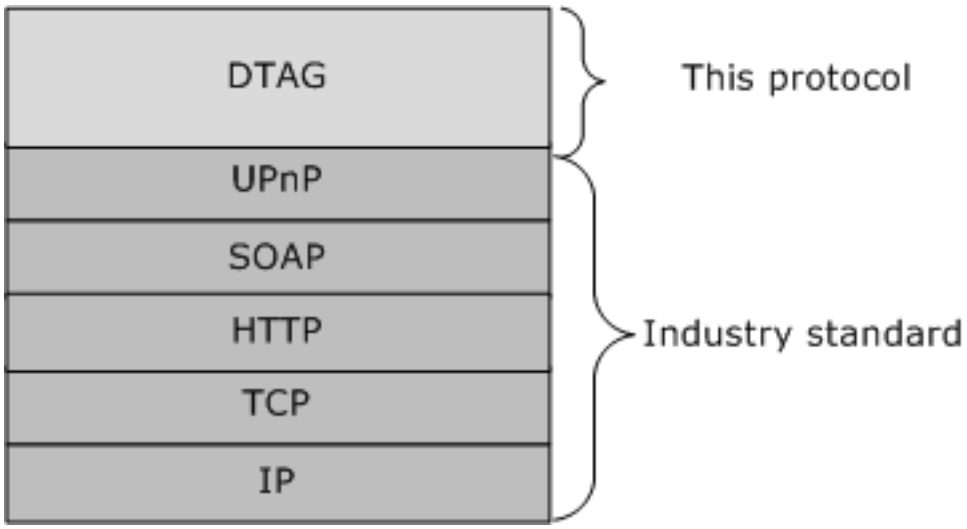

Figure 2: Relationship of DTAG to other protocols

DTAG is built on UPnP architecture version 1.0 [UPNPARCH1] and version 1.1 [UPNPARCH1.1]. For the purposes of this specification, descriptions of the **XML** and SOAP schema are provided via references to UPnP architecture version 1.1 [UPNPARCH1.1].

1.5 Prerequisites/Preconditions

DTAG requires support for storing trusted **certificates** in a tamper-proof manner.

DTAG requires the support of a **UPnP** stack on **device** and **control point**. The device is required to have the **service description** for DTAG. The full UPnP service description of DTAG is provided in section 8. The device description is also required to include the information about the DTAG **service**, for which the **service type** is "mstrustagreement", the service identifier is "MSTA", and the version number is as specified in section 1.7. The protocol server **endpoint** is formed by appending "/_vti_bin/pptws.asmx".

Before DTAG can be used, all of the necessary, initial UPnP operations are required to be completed, including discovery of devices and publication of service/device descriptions as specified in [UPNPARCH1.1].

1.6 Applicability Statement

Use of DTAG is suitable when the **UPnP** device and **control point** are required to ensure the secure exchange of **certificates** over an unsecure network where the messages can be exposed to a third party or even tampered with.

1.7 Versioning and Capability Negotiation

This document specifies DTAG version 1. The version number is recommended to be included where DTAG **service** information is presented in a **device** description, as specified in [UPNPARCH1.1] section 2.3.

This protocol does not have a specific **WSDL** declaration.

1.8 Vendor-Extensible Fields

None.

Copyright © 2017 Microsoft Corporation

1.9 Standards Assignments

None.

Copyright © 2017 Microsoft Corporation

2 Messages

2.1 Transport

DTAG is implemented as a **UPnP service** and does not specify any transport details beyond what is specified by [UPNPARCH1] section 3.

2.2 Common Message Syntax

This section contains common definitions used by this protocol. The syntax of the definitions uses **XML schema (XSD)** as defined in [XMLSCHEMA1] and [XMLSCHEMA2], and **Web Services Description Language (WSDL)** as defined in [WSDL].

2.2.1 Namespaces

This specification defines and references various **XML namespaces** using the mechanisms specified in [XMLNS-2ED]. Although this specification associates a specific XML namespace prefix for each XML namespace that is used, the choice of any particular XML namespace prefix is implementation-specific and not significant for interoperability.

Prefix	Namespace URI	Reference
s, SOAP-ENV	http://schemas.xmlsoap.org/soap/envelope/	[SOAP1.1]
m	urn:schemas-microsoft-com:service:mstrustagreement:1	
dt	urn:schemas-microsoft-com:datatypes	[MSDN-XDR]

2.2.2 Messages

The following table summarizes the set of common **SOAP message** definitions defined by this specification. SOAP message definitions that are specific to a particular operation are described with the operation.

Message	Description
UPnP Error	Sends a **UPnP** error message using a **SOAP** 1.1 UPnP profile, as specified in [UPNPARCH1.1] section 3.1.

2.2.2.1 UPnP Error

DTAG error messages MUST be expressed in **XML** using a **SOAP** 1.1 **UPnP** profile, as specified in [UPNPARCH1.1] section 3.1. For the purpose of this specification, this section specifies the **SOAP fault** message that is used to support UPnP error reporting.

All SOAP faults defined in this specification MUST be sent as described in [WSASB] section 6. For compatible UPnP error reporting, the values of the SOAP fault elements MUST be set as follows.

SOAP Fault Element	Value
<faultcode>	s:Client
<faultstring>	UPnPError
<detail>	<UPnPError> element (section 2.2.3.1)

Copyright © 2017 Microsoft Corporation

2.2.3 Elements

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

Element	Description
<UPnPError>	A wrapper used to support the **UPnP** error reporting format.
<HostID>	The unique identifier of the **control point**.
<Iteration>	The iteration number of the current **Validate action**.
<IterationsRequired>	The number of rounds of **Validate** actions.

2.2.3.1 UPnPError

DTAG error messages MUST be expressed in **XML** using a **SOAP** 1.1 **UPnP** profile, as specified in [UPNPARCH1.1] section 3.1. For this expression, the <UPnPError> element can be defined as follows and included as part of the <detail> element of the **SOAP fault** message, as specified in section 2.2.2.1.

```
<xs:element name="UPnPError">
    <xs:complexType>
        <xs:sequence>
            <xs: element name="ErrorCode" type="xs:integer"/>
            <xs: element name="ErrorDescription" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

The following table lists the possible values of the <ErrorCode> and <ErrorDescription> elements. If an **action** results in multiple errors, the most specific error MUST be returned.

ErrorCode	ErrorDescription	Explanation
401	Invalid Action	See the description of Control in [UPNPARCH1.1] section 3.
402	Invalid Args	Parameters are missing, extra, or are invalid for this action. See the description of Control in [UPNPARCH1.1] section 3.
403	Out of Sync	See the description of Control in [UPNPARCH1.1] section 3.
501	Action Failed	The **service** was not able to process this action, or the action is not allowed in the current state. See the description of Control in [UPNPARCH1.1] section 3.
801	Invalid Endpoint	The parameter, <HostID>, has an invalid format or is inconsistent with previous usage.
802	Invalid Certificate	The parameter, <HostCertificate>, has an invalid format or does not reference the <HostID>.
803	Invalid Nonce	The authentication process failed.

2.2.3.2 HostID

This element contains the **state variable** **_HostID**, described in section 3.1.1, which is the unique identifier information specific to the **control point**.

Copyright © 2017 Microsoft Corporation

```
<xs:element name="HostID" type="A_ARG_TYPE_EndpointID"/>
```

2.2.3.3 Iteration

This element contains the **state variable** **Iter**, described in section 3.1.1, indicating the iteration number of the current **Validate** **action**.

```
<xs:element name="Iteration" type="A_ARG_TYPE_Iteration"/>
```

2.2.3.4 IterationsRequired

This element contains the **state variable** **N**, described in section 3.1.1, which is used to negotiate the number of rounds of **Validate** actions to complete the trust agreement process.

```
<xs:element name="IterationsRequired" type="A_ARG_TYPE_Rounds"/>
```

2.2.4 Complex Types

This specification does not define any common XML schema complex type definitions.

2.2.5 Simple Types

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

Simple type	Description
<A_ARG_TYPE_Rounds>	The number of rounds required for the **Validate** **action**.
<A_ARG_TYPE_Iteration>	The iteration number for the current **Validate** action.
<A_ARG_TYPE_EndpointID>	The **UUID** of the **device** (or the **control point**).
<A_ARG_TYPE_Authenticator>	A 20-octet authentication code.
<A_ARG_TYPE_Nonce>	An array of 20 octets (for a total of 160 bits) that contains cryptographically strong random values.
<A_ARG_TYPE_Certificate>	The **certificate** of the device (or the control point).

2.2.5.1 A_ARG_TYPE_Rounds

This type of element is used to negotiate the number of rounds required for **Validate** actions to be performed by the protocol.

```
<xs:simpleType name="A_ARG_TYPE_Rounds" >
    <xs:restriction base="xs:unsignedByte" >
        <xs:minInclusive value="2"/>
        <xs:maxInclusive value="20"/>
    </xs:restriction>
</xs:simpleType>
```


Copyright © 2017 Microsoft Corporation

The number of rounds MUST be in the range of 2 to 20, inclusive.

2.2.5.2 A_ARG_TYPE_Iteration

This type of element is limited to the values between 1 and 20. These values correspond to each of the **N** times that the **Commit** and **Validate** actions are called.

```
<xs:simpleType name="A_ARG_TYPE_Iteration" >
    <xs:restriction base="xs:unsignedByte" >
        <xs:minInclusive value="1"/>
        <xs:maxInclusive value="20"/>
    </xs:restriction>
</xs:simpleType>
```

2.2.5.3 A_ARG_TYPE_EndpointID

This type of element is a string that uniquely identifies an **endpoint**. It has to remain stable for the lifetime of the **device**.

```
<xs:simpleType name="A_ARG_TYPE_EndpointID" >
    <xs:restriction base="xs:string"/>
</xs:simpleType>
```

2.2.5.4 A_ARG_TYPE_Authenticator

This type of element is an **authenticator** and is the 160-bit (20-octet) result of the **HMAC-SHA-1** message authentication code [RFC2104], as specified in section 3.1.1, encoded as a **base64** string.

```
<xs:simpleType name="A_ARG_TYPE_Authenticator" >
    <xs:restriction base="xs:string"/>
</xs:simpleType>
```

2.2.5.5 A_ARG_TYPE_Nonce

This type of element is an array of 20 octets (160 bits) that contains cryptographically-strong random values, encoded as a **base64** string.

```
<xs:simpleType name="A_ARG_TYPE_Nonce" >
    <xs:restriction base="xs:string"/>
</xs:simpleType>
```

2.2.5.6 A_ARG_TYPE_Certificate

This type of element is a string and contains a **certificate** encoded as a **base64** string.

```
<xs:simpleType name="A_ARG_TYPE_Certificate" >
    <xs:restriction base="xs:string"/>
</xs:simpleType>
```


Copyright © 2017 Microsoft Corporation

2.2.6 Attributes

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

2.2.7 Groups

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

2.2.8 Attribute Groups

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

Copyright © 2017 Microsoft Corporation

3 Protocol Details

The operations of the **device** and **control point** are almost symmetric because they examine each other using the same types of information.

3.1 Common Details

This section describes protocol details that are common between the **device** and **control point**.

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 **device** and **control point** start the trust agreement process when a **one-time password (OTP)** is made available to the two **endpoints**. Throughout the trust agreement process, the device and control point MUST synchronize the state to perform each **action**.

The following diagram provides an overview of the state machine common to the device and control point.

Copyright © 2017 Microsoft Corporation

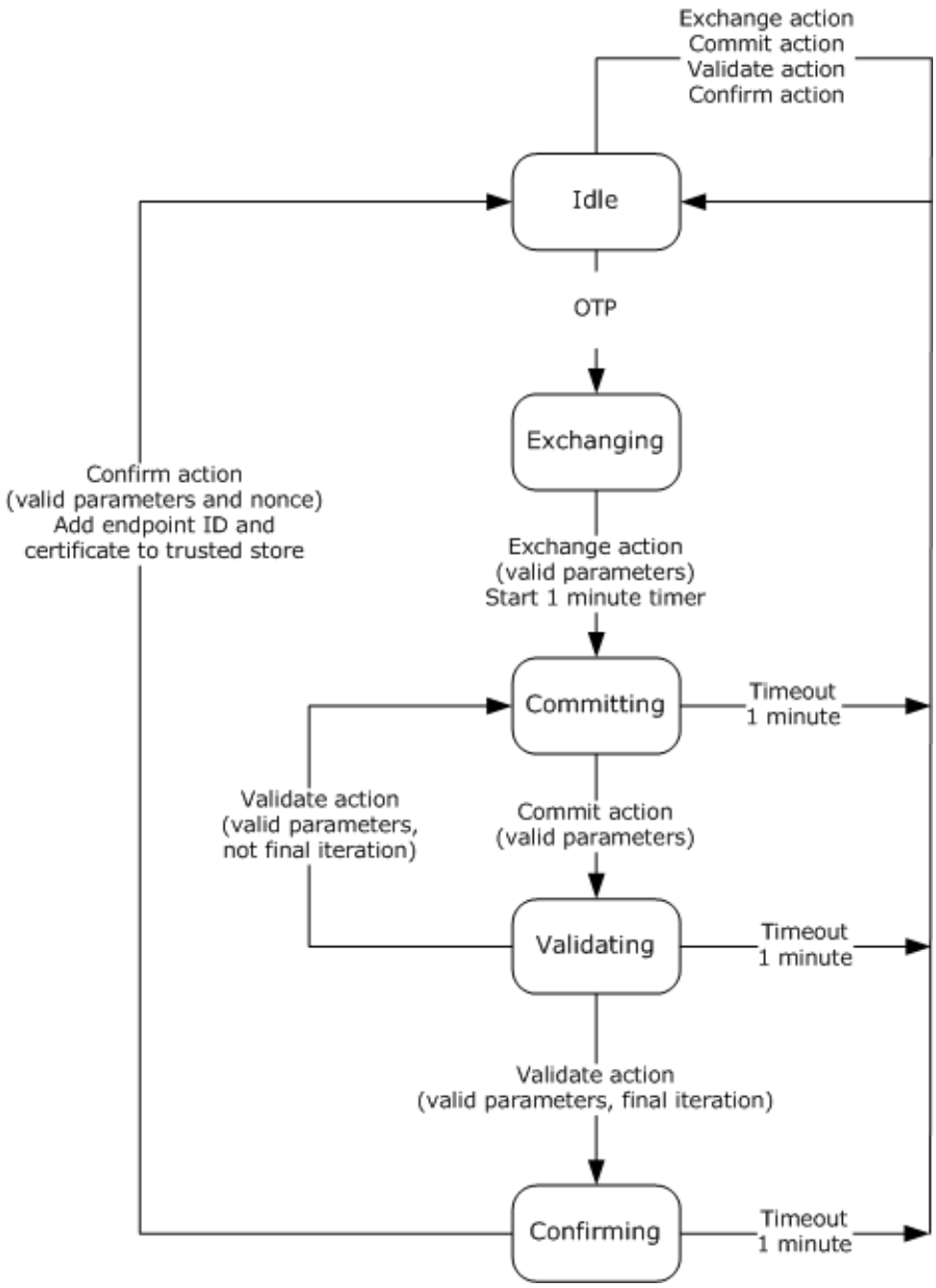

Figure 3: DTAG message sequence to establish trust agreement

TrustState: The current setting of the service's state machine. The following states are specified for this **state variable**.

TrustState	State	Description
0	*Idle*	The trust agreement process is not started. The device and control point wait for a one-time password (OTP) event.
1	*Exchanging*	The device and control point exchange **certificates** and endpoint identifiers, along with the authentication code based on the entire **OTP** string. This authentication code will be examined in the last *Confirming* state. The

Copyright © 2017 Microsoft Corporation

TrustState	State	Description
		Exchange action is processed in this state.
2	*Committing*	The device and control point exchange the authentication code based on **OTP** substrings. The **Commit** action and timeout event are processed in this state.
3	*Validating*	The device and control point validate the authentication code exchanged on the previous **Commit** action. The **Validate** action and timeout event are processed in this state.
4	*Confirming*	The device and control point finalize the validation of the authentication code obtained in the **Exchange** action. The **Confirm** action and timeout event are processed in this state.

N: The number of rounds required for the **Commit**-**Validate** actions that will be perfomed by the protocol. The value of this state variable is selected at run-time.

Iter: The current iteration number at which **Commit**-**Validate** actions are performed. This state variable is only valid up to **N**.

OTP: The one-time password (OTP).

OTPIter: The substring **OTP** for the indicated iteration.

The **OTP** and its substrings are obtained by the following rule.

The **OTP** is divided up into **N** substrings. These substrings are denoted as OTP_1, OTP_2, ... OTP_n. The rule for generating the substring **OTPs** from the **OTP** is as follows:

- Individual characters in an **OTP** are not broken up.
- The number of characters in the **OTP** MUST be greater than or equal to the number of rounds specified in the state variable **N**.
- If *L* is the number of characters in the **OTP**, then each substring will be either *L div***N** or *L div***N**+1 characters long. The last *L mod***N** substrings will have *L div***N**+1 characters. All of the other substrings will have *L div***N** characters.
- The characters of the **OTP** are broken up in order into their substrings.

For example, if the value of **N** is 4 and the value of the **OTP** is "ThatCat", then the first substring, OTP1 would be "T", the second, OTP2 would be "ha", the third, OTP3 would be "tC", and the fourth, OTP4 would be "at".

_DeviceCertificate: The certificate of the device that is associated with the **_DeviceID** state variable and which MUST remain stable for the lifetime of the device.

_DeviceConfirmAuthenticator: The authentication code made by the device for the **Exchange** and **Confirm** actions.

_DeviceConfirmNonce: A 20-octet **nonce** made by the device for the **Exchange** and **Confirm** actions.

_DeviceID: The **UUID** of the device.

_DeviceValidateAuthenticatorIter: The authentication code of the device for the indicated iteration of **Commit**-**Validate** actions.

_DeviceValidateNonceIter: A 20-octet nonce of the device for the indicated iteration of **Commit**-**Validate** actions.

_HostCertificate: The certificate of the control point that is associated with the **_HostID** state variable and which MUST remain stable for the lifetime of the control point.

Copyright © 2017 Microsoft Corporation

_HostConfirmAuthenticator: The authentication code of the control point for the **Exchange** and **Confirm** actions.

_HostConfirmNonce: A 20-octet nonce of the control point for the **Exchange** and **Confirm** actions.

_HostID: The UUID of the control point.

_HostValidateAuthenticatorIter: The authentication code of the control point for the indicated iteration of the **Commit**-**Validate** action.

_HostValidateNonceIter: A 20-octet nonce of the control point for the indicated iteration of the **Commit**-**Validate** action.

The **_DeviceValidateAuthenticatorIter**, **_DeviceConfirmAuthenticator**, **_HostValidateAuthenticatorIter**, and **_HostConfirmAuthenticator** are the 160-bit (20-octet) result of the **HMAC-SHA-1** message authentication code [RFC2104]. The HMAC-SHA-1 function takes two parameters, a 20-octet key and some variable-length text, and returns a 20-octet message authentication code.

The HMAC-SHA-1 function key is a nonce.

The HMAC-SHA-1 function text is the **UTF-8** representation [RFC3629] of the concatenation of the following items in the order presented:

- **N** (or **Iter**), encoded as a decimal number string
- An **OTP** string (or **OTPIter** substring)
- The endpoint identifier
- A certificate, encoded as a **base64** string

Therefore, the HMAC-SHA-1 results are denoted in this specification as:

_DeviceConfirmAuthenticator

= **HMAC**(**_DeviceConfirmNonce**, UTF-8(**N** + **OTP** + **_DeviceID** + **_DeviceCertificate**)

_HostConfirmAuthenticator

= HMAC(**_HostConfirmNonce**, UTF-8(**N** + **OTP** + **_HostID** + **_HostCertificate**)

_DeviceValidateAuthenticatorIter

= HMAC(**_DeviceValidateNonceIter**, UTF-8(**IterIter** + **OTPIter** + **_DeviceID** + **_DeviceCertificate**)

_HostValidateAuthenticatorIter

= HMAC(**_HostValidateNonceIter**, UTF-8(**IterIter** + **OTPIter** + **_HostID** + **_HostCertificate**)

3.1.2 Timers

None.

3.1.3 Initialization

Before startup, the **device** and **control point** keep the **TrustState** **state variable** set to 0 (*Idle*). In this state, any of the service's actions MUST NOT be called, and invoking any one of them MUST return an error.

Copyright © 2017 Microsoft Corporation

3.1.4 Message Processing Events and Sequencing Rules

3.1.4.1 One-time Password (OTP) Event

An **OTP** event outside of the scope of this specification (for example, human interaction) triggers the start of the trust agreement process. When the trigger event occurs, the **service** MUST initiate the process as follows:

1. The **device** and **control point** MUST terminate any ongoing DTAG process, discarding all locally saved **OTPs**, **nonces**, **endpoint** identifiers, and **certificates**.
2. The device and control point MUST acquire and locally save the endpoint identifier (in other words, **_DeviceID** and **_HostID**, respectively).
3. The device and control point MUST acquire and locally save the certificate (in other words, **_DeviceCertificate** and **_HostCertificate**, respectively).
4. The device and control point MUST acquire and locally save the **OTP**, and generate the substrings, as described in section 3.1.1.
5. The device and control point MUST change **TrustState** from 0 (*Idle*) to 1 (*Exchanging*), as described in section 3.1.1.

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

3.2 Device Details

In addition to the protocol details specified in section 3.1, the following details are also applied to the **device**.

3.2.1 Abstract Data Model

None.

3.2.2 Timers

The **device** and **control point** each have a 1 minute timer "TimeOut" for the maximum interval allowed for the transition between actions.

3.2.3 Initialization

None.

3.2.4 Message Processing Events and Sequencing Rules

On each **action**, the **control point** sends a request message to the **device**, and the device returns a response or error message to the control point, as specified in [UPNPARCH1.1] section 3.1.

Copyright © 2017 Microsoft Corporation

3.2.4.1 Exchange Action

In order to perform the **Exchange action**, the **control point** MUST attach an <Exchange> body to the DTAG **SOAP message** that contains the <HostID>, <HostCertificate>, <IterationsRequired>, and <HostConfirmAuthenticator> elements. This action is supported only when **TrustState** is 1 (*Exchanging*).

On this action, the following checks MUST be performed:

1. The <HostID>, <HostCertificate>, <IterationsRequired>, and <HostConfirmAuthenticator> elements MUST be syntactically validated.
2. The <HostCertificate> (**_HostCertificate**) MAY be validated as per any vendor-defined rules.
3. The <IterationsRequired> (**N**) MAY additionally be checked per vendor-defined rules.

If successful, the **device**:

1. MUST locally save the values of **_HostID**, **_HostCertificate**, and **_HostConfirmAuthenticator**.
2. MUST generate and locally save **_DeviceConfirmNonce**.
3. MUST change **TrustState** from 1 (*Exchanging*) to 2 (*Committing*).
4. MUST start a one-minute timer.
5. MUST set the following elements and return with status 200 (success):
   - The <DeviceID>, the **UUID** of the enclosing **UPnP** device, as specified in A_ARG_Type_EndpointID (section 2.2.5.3), and as acquired in section 3.1.4.1.
   - The <DeviceCertificate>, as specified in A_ARG_TYPE_Certificate (section 2.2.5.6), and as acquired in section 3.1.4.1.
   - The <DeviceConfirmAuthenticator>, an **HMAC** as specified in section 3.1.1, calculated as:

     Base64 (HMAC(**_DeviceConfirmNonce**, **UTF-8** (**N** + **OTP** + **_DeviceID** + **_DeviceCertificate**)).

If this action fails, the device MUST find the appropriate error code from the table in section 2.2.3.1 and send a **SOAP fault** message to the control point, as specified in section 2.2.2.1.

3.2.4.1.1 Messages

Message	Description
Exchange	Contains the request for the **Exchange action**.
Exchange Response	Contains the response of the **Exchange** action.

3.2.4.1.1.1 Exchange Message

The HTTP header MUST specify the SOAPACTION for the **Exchange action** as follows:

SOAPACTION: "urn:schemas-microsoft-com:service: mstrustagreement:1#Exchange"

Where "urn:schemas-microsoft-com:service: mstrustagreement:1" is the **service type** that comes from the **device** description, as specified in section 7, and "#Exchange" is the **SOAP action**.

The following **XML** session shows the <HostId>, <HostCertificate>, <IterationsRequired>, and <HostConfirmAuthenticator> elements in a **SOAP** Exchange message.

Copyright © 2017 Microsoft Corporation

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:Exchange xmlns:m="urn:schemas-microsoft-com:service:mstrustagreement:1">
      <HostID xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="string">
        Control point identifier
      </HostID>
      <HostCertificate xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="string">
        Host Certificate payload
      </HostCertificate>
      <IterationsRequired xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="ui1">
        Number of iterations required
      </IterationsRequired>
      <HostConfirmAuthenticator xmlns:dt="urn:schemas-microsoft-com:datatypes"
dt:dt="string">
        HostConfirmAuthenticator payload
      </HostConfirmAuthenticator>
    </m:Exchange>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

HostID: The <HostID> element, as specified in section 2.2.3.2.

HostCertificate: The <HostCertificate> element, as specified in section 3.2.4.1.2.2.

IterationsRequired: The <IterationsRequired> element, as specified in section 2.2.3.4.

HostConfirmAuthenticator: The <HostConfirmAuthenticator> element, as specified in section 3.2.4.1.2.4.

3.2.4.1.1.2 Exchange Response Message

The **device** MUST reply with an ExchangeResponse SOAP response message, which contains the <DeviceID>, <DeviceCertificate>, and <DeviceConfirmAuthenticator> elements.

```
<s:Envelope xmlns:s=http://schemas.xmlsoap.org/soap/envelope/
  s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:ExchangeResponse xmlns:u="urn:schemas-microsoft-com:service:mstrustagreement:1">
      <DeviceID>
        Device identifier
      </DeviceID>
      <DeviceCertificate>
        Device Certificate payload
      </DeviceCertificate>
      <DeviceConfirmAuthenticator>
        DeviceConfirmAuthenticator payload
      </DeviceConfirmAuthenticator>
    </u:ExchangeResponse>
  </s:Body>
</s:Envelope>
```

DeviceID: The <DeviceID> element, as specified in section 3.2.4.1.2.1.

DeviceCertificate: The <DeviceCertificate> element, as specified in section 3.2.4.1.2.3.

DeviceConfirmAuthenticator: The <DeviceConfirmAuthenticator> element, as specified in section 3.2.4.1.2.5.

3.2.4.1.2 Elements

Copyright © 2017 Microsoft Corporation

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

Element	Description
<DeviceID>	A string that contains the **_DeviceID**, as described in section 3.1.1.
<HostCertificate>	A **base64** encoded string that contains the **_HostCertificate**, as described in section 3.1.1.
<DeviceCertificate>	A base64 encoded string that contains the **_DeviceCertificate**, as described in section 3.1.1.
<HostConfirmAuthenticator>	A base64 encoded string that contains the **_HostConfirmAuthenticator**, as described in section 3.1.1.
<DeviceConfirmAuthenticator>	A base64 encoded string that contains the **_DeviceConfirmAuthenticator**, as described in section 3.1.1.

3.2.4.1.2.1 DeviceID

This element provides the unique identifier information specific to the **device** (**_DeviceID**). This element is contained in the **SOAP body** of the ExchangeResponse message.

```
<xs:element name="DeviceID" type="A_ARG_TYPE_EndipointID"/>
```

3.2.4.1.2.2 HostCertificate

This element provides the **control point certificate** (**_HostCertificate**). This element is contained in the **SOAP body** of the ExchangeResponse message and is encoded as a **base64** string.

```
<xs:element name="HostCertificate" type="A_ARG_TYPE_Certificate"/>
```

3.2.4.1.2.3 DeviceCertificate

This element provides the **certificate** of the **device** associated with the <DeviceID> (**_DeviceCertificate**) encoded as a **base64** string. This element is contained in the **SOAP body** of the ExchangeResponse message and is encoded as a base64 string.

```
<xs:element name="HostCertificate" type="A_ARG_TYPE_Certificate"/>
```

3.2.4.1.2.4 HostConfirmAuthenticator

This element is an **authenticator** that provides the 160-bit (20-octet) authentication code for the **control point** (**_HostConfirmAuthenticator**). This element is contained in the **SOAP body** of the Exchange message that and is encoded as a **base64** string.

```
<xs:element name="HostConfirmAuthenticator" type="A_ARG_TYPE_Authenticator"/>
```

3.2.4.1.2.5 DeviceConfirmAuthenticator

This element is an **authenticator** that provides the 160-bit (20-octet) authentication code made by the **device** (**_DeviceConfirmAuthenticator**), specified in section 3.2.4.1. This element is contained in the **SOAP body** of the ExchangeResponse message and is encoded as a **base64** string.

Copyright © 2017 Microsoft Corporation

```
<xs:element name="HostConfirmAuthenticator" type="A_ARG_TYPE_Authenticator"/>
```

3.2.4.2 Commit Action

In order to perform the **Commit** **action**, the **control point** MUST attach a <Commit> body to the DTAG SOAP message that contains the <HostID>, <Iteration>, and <HostValidateAuthenticator> elements. This action is only supported when **TrustState** is 2 (*Committing*).

On this action, the following checks MUST be performed:

1. The <HostID>, <Iteration>, and <HostValidateAuthenticator> (**_HostValidateAuthenticatorIter**) elements MUST be syntactically validated.

2. The <HostID> MUST match the value of the **_HostID** obtained in the **Exchange** action.

If successful, the **service**:

1. MUST change **TrustState** from 2 (*Committing*) to 3 (*Validating*).

2. MUST generate **_DeviceValidateNonceIter**.

3. MUST set the following element and return with status 200 (success).

   - The <DeviceValidateAuthenticator>, an **HMAC** as specified in section 3.1.1, calculated as:

     Base64 (HMAC(**_DeviceValidateNonceIter**, **UTF-8**(**Iter** + **OTPIter** + **_DeviceID** + **_DeviceCertificate**)).

If this action fails, the **device** MUST find the appropriate error code from the table in section 2.2.3.1 and send a SOAP fault message to the control point, as specified in section 2.2.2.1.

3.2.4.2.1 Messages

Message	Description
Commit	Contains the request for the **Commit** **action**.
Commit Response	Contains the response of the **Commit** action.

3.2.4.2.1.1 Commit Message

The HTTP header MUST specify the SOAPACTION for the **Commit** **action** as follows:

SOAPACTION: "urn:schemas-microsoft-com:service: mstrustagreement:1#Commit"

Where "urn:schemas-microsoft-com:service: mstrustagreement:1" is the **service type** which comes from the **device** description as specified in section 7 and "#Commit" is the **SOAP action**.

The following **XML** session shows the <HostId>, <Iteration>, and <HostValidateAuthenticator> elements in a SOAP Commit message.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:Commit xmlns:m="urn:schemas-microsoft-com:service:mstrustagreement:1">
      <HostID xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="string">
        Host identifier
      </HostID>
```


Copyright © 2017 Microsoft Corporation

```
      <Iteration xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="ui1">
        Current iteration
      </Iteration>
      <HostValidateAuthenticator xmlns:dt="urn:schemas-microsoft-com:datatypes"
dt:dt="string">
        HostValidateAuthenicator payload
      </HostValidateAuthenticator>
    </m:Commit>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

HostID: The <HostID> element, as specified in section 2.2.3.2.

Iteration: The <Iteration> element, as specified in section 2.2.3.3.

HostValidateAuthenticator: The <HostValidateAuthenticator> element, as specified in section 3.2.4.2.2.1.

3.2.4.2.1.2 Commit Response Message

The server MUST reply with a CommitResponse **SOAP** response message that containis the <DeviceValidateAuthenticator> element.

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
  s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:CommitResponse xmlns:u="urn:schemas-microsoft-com:service:mstrustagreement:1">
      <DeviceValidateAuthenticator>
        DeviceValidateAuthenticator payload
      </DeviceValidateAuthenticator>
    </u:CommitResponse>
  </s:Body>
</s:Envelope>
```

DeviceValidateAuthenticator: The <DeviceValidateAuthenticator> element, as specified in section 3.2.4.2.2.2.

3.2.4.2.2 Elements

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

Element	Description
<HostValidateAuthenticator>	A **base64** encoded string that contains the **_HostValidateAuthenticatorIter**, as described in section 3.1.1.
<DeviceValidateAuthenticator>	A base64 encoded string that contains the **_DeviceValidateAuthenticatorIter**, as described in section 3.1.1.

3.2.4.2.2.1 HostValidateAuthenticator

This element is an **authenticator** that provides the 160-bit (20-octet) authentication code for the **control point** (**_HostValidateAuthenticatorIter**). This element is contained in the **SOAP body** of the Commit message and is encoded as a **base64** string.

Copyright © 2017 Microsoft Corporation

```
<xs:element name="HostValidateAuthenticator" type="A_ARG_TYPE_Authenticator"/>
```

3.2.4.2.2.2 DeviceValidateAuthenticator

This element is an **authenticator** that provides the 160-bit (20-octet) authentication code for the **device** (**_DeviceValidateAuthenticatorIter**), as specified in section 3.2.4.2. This element is contained in the **SOAP body** of the CommitResponse message and is encoded as a **base64** string.

```
<xs:element name="DeviceValidateAuthenticator" type="A_ARG_TYPE_Authenticator"/>
```

3.2.4.3 Validate Action

In order to perform the **Validate action**, the **control point** MUST attach a <Validate> body to the DTAG SOAP message that contains the <HostID>, <Iteration>, and <HostValidateNonce> elements. This action is only valid if **TrustState** is 3 (*Validating*).

On this action, the following checks MUST be performed:

1. The <HostID>, <Iteration>, and <HostValidateNonce> (**_HostValidateNonceIter**) elements MUST be syntactically validated.
2. The <HostID> MUST match the value of the **_HostID** obtained in the **Exchange** action.
3. The <Iteration> number MUST be equal to the device's current iteration number, **Iter**.
4. The value of HMAC(**_HostValidateNonceIter**, **UTF-8**(**Iter** + **OTPIter** + **_HostID** + **_HostCertificate**)) calculated as specified in section 3.1.1, MUST match the **_HostValidateAuthenticatorIter** obtained in the **Commit** action.

If successful, the **service**:

1. MUST increment the iteration number, **Iter**.
2. MUST change **TrustState** from 3 (*Validating*) to 4 (*Confirming*), if this is the last iteration, or to 2 (*Committing*) if this is not the last iteration.
3. MUST set the following element and return with status 200 (success).
   - <DeviceValidateNonce>, a **base64** encoded string of **_DeviceValidateNonceIter**, which is the 20-octet random number acquired in section 3.2.4.2.

If this action fails, the **device** MUST find the appropriate error code from the table in section 2.2.3.1 and send a SOAP fault message to the control point, as specified in section 2.2.2.1.

3.2.4.3.1 Messages

Message	Description
Validate	Contains the request for the **Validate action**.
Validate Response	Contains the response of the **Validate** action.

3.2.4.3.1.1 Validate Message

The HTTP header MUST specify the SOAPACTION for the **Validate action** as follows:

Copyright © 2017 Microsoft Corporation

SOAPACTION: "urn:schemas-microsoft-com:service: mstrustagreement:1#Validate"

Where "urn:schemas-microsoft-com:service: mstrustagreement:1" is the **service type**, which comes from the **device** description, as specified in section 7 and "#Validate" is the **SOAP action**.

The following **XML** session shows the <HostID>, <Iteration>, and <HostValidateNonce> elements in a **SOAP** Validate message.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:Validate xmlns:m="urn:schemas-microsoft-com:service:mstrustagreement:1">
      <HostID xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="string">
        Host identifier
      </HostID>
      <Iteration xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="ui1">
        Current iteration
      </Iteration>
      <HostValidateNonce xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="string">
        HostValidateNonce payload
      </HostValidateNonce>
    </m:Validate>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

HostID: The <HostID> element, as specified in section 2.2.3.2.

Iteration: The <Iteration> element, as specified in section 2.2.3.3.

HostValidateNonce: The <HostValidateNonce> element, as specified in section 3.2.4.3.2.1.

3.2.4.3.1.2 Validate Response Message

The server MUST reply with a ValidateResponse **SOAP** response message that contains the <DeviceValidateNonce> element.

```
<s:Envelope.. xmlns:s=http://schemas.xmlsoap.org/soap/envelope/
  s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:ValidateResponse xmlns:u="urn:schemas-microsoft-com:service:mstrustagreement:1">
      <DeviceValidateNonce>
        DeviceValidateNonce payload
      </DeviceValidateNonce>
    </u:ValidateResponse>
  </s:Body>
</s:Envelope>
```

DeviceValidateNonce: The <DeviceValidateNonce> element, as specified in section 3.2.4.3.2.2.

3.2.4.3.2 Elements

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

Element	Description
<HostValidateNonce>	A **base64** encoded string that contains the **_HostValidateNonceIter**, as described in section 3.1.1.
<DeviceValidateNonce>	A base64 encoded string that contains the **_DeviceValidateNonceIter**, as described

Copyright © 2017 Microsoft Corporation

Element	Description
	in section 3.1.1.

3.2.4.3.2.1 HostValidateNonce

This element provides the **nonce** of the **control point** for the indicated iteration of the **Commit-Validate action** (**_HostValidateNonceIter**). This element is contained in the **SOAP body** of the Validate message and is encoded as a **base64** string.

```
<xs:element name="HostValidateNonce" type="A_ARG_TYPE_Nonce"/>
```

3.2.4.3.2.2 DeviceValidateNonce

This element provides the **nonce** of the **device** for the indicated iteration of the **Commit-Validate action** (**_DeviceValidateNonceIter**). This element is contained in the **SOAP body** of the ValidateResponse message and is encoded as a **base64** string.

```
<xs:element name="DeviceValidateNonce" type="A_ARG_TYPE_Nonce"/>
```

3.2.4.4 Confirm Action

In order to perform the **Confirm action**, the **control point** MUST attach a <Confirm> body to the DTAG SOAP message that contains the <HostID>, <IterationsRequired>, and <HostConfirmNonce> elements. This action is only valid if **TrustState** is 4 (*Confirming*).

On this action, the following checks MUST be performed:

The <HostID>, <HostConfirmNonce>, and <IterationsRequired> (**N**) MUST be syntactically validated.

The <HostID> MUST match the value of the **_HostID** obtained in the **Exchange** action.

The value of **HMAC**(**_HostConfirmNonce**, **UTF-8** (**N** + **OTP** + **_HostID** + **_HostCertificate**)), as specified in section 3.1.1, MUST match the **_HostConfirmAuthenticator** acquired in the **Exchange** action.

If successful, then trust has been established, and the **service**:

MUST store **_HostID** and **_HostCertificate** in a tamper-proof, persistent store.

MUST change **TrustState** from 4 (*Confirming*) to 0 (*Idle*).

MUST set the following element and return with status 200 (success):

<DeviceConfirmNonce>, a **base64** encoded string of **_DeviceConfirmNonce**, which is the 20 octet random number acquired in section 3.1.4.1.

If this action fails, the **device** MUST find the appropriate error code from the table in section 2.2.3.1 and send a **SOAP fault** message to the control point, as specified in section 3.1.1.

3.2.4.4.1 Messages

Message	Description
Confirm	Contains the request for the **Confirm action**.

Copyright © 2017 Microsoft Corporation

Message	Description
Confirm Response	Contains the response of the **Confirm** action.

3.2.4.4.1.1 Confirm Message

The HTTP header MUST specify the SOAPACTION for the **Confirm** **action** as follows:

SOAPACTION: "urn:schemas-microsoft-com:service: mstrustagreement:1#Confirm"

Where "urn:schemas-microsoft-com:service: mstrustagreement:1" is the **service type** that comes from the **device** description, as specified in section 7, and "#Confirm" is the **SOAP action**.

The following **XML** session shows the <HostID>, <IterationsRequired>, and <HostConfirmNonce> in a **SOAP** Confirm message.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:Confirm xmlns:m="urn:schemas-microsoft-com:service:mstrustagreement:1">
      <HostID xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="string">
        Host identifier
      </HostID>
      <IterationsRequired xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="ui1">
        Number of iterations requested
      </IterationsRequired>
      <HostConfirmNonce xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="string">
        HostConfirmationNonce payload
      </HostConfirmNonce>
    </m:Confirm>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

HostID: The <HostID> element, as specified in section 2.2.3.2.

IterationsRequired: The <IterationsRequired> element, as specified in section 2.2.3.4.

HostConfirmNonce: The <HostConfirmNonce> element, as specified in section 3.2.4.4.2.1.

3.2.4.4.1.2 Confirm Response Message

The server MUST reply with a ConfirmResponse **SOAP** response message that contains the <DeviceConfirmNonce> element.

```
<s:Envelope xmlns:s=http://schemas.xmlsoap.org/soap/envelope/
  ..s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:ConfirmResponse xmlns:u="urn:schemas-microsoft-com:service:mstrustagreement:1">
      <DeviceConfirmNonce>
        DeviceConfirmNonce payload
      </DeviceConfirmNonce>
    </u:ConfirmResponse>
  </s:Body>
</s:Envelope>
```

DeviceConfirmNonce: The <DeviceConfirmNonce> element, as specified in section 3.2.4.4.2.2

Copyright © 2017 Microsoft Corporation

3.2.4.4.2 Elements

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

Element	Description
<HostConfirmNonce>	A **base64** encoded string that contains the **_HostConfirmNonce**,as described in section 3.1.1.
<DeviceConfirmNonce>	A base64 encoded string that contains the **_DeviceConfirmNonce**, as described in section 3.1.1.

3.2.4.4.2.1 HostConfirmNonce

This element provides the **nonce** of the **control point** for the indicated iteration of the **Exchange** and **Confirm** actions (**_HostConfirmNonce**). This element is contained in the **SOAP body** of the Confirm message and is encoded as a **base64** string.

```
<xs:element name="HostConfirmNonce" type="A_ARG_TYPE_Nonce"/>
```

3.2.4.4.2.2 DeviceConfirmNonce

This element provides the **nonce** of the **device** for the indicated iteration of the **Exchange** and **Confirm** actions (**_DeviceConfirmNonce**). This element is contained in the **SOAP body** of the ConfirmResponse message and is encoded as a **base64** string.

```
<xs:element name="DeviceConfirmNonce" type="A_ARG_TYPE_Nonce"/>
```

3.2.5 Timer Events

After sending each response message, if the **device** does not receive the message of the next **action** within one minute, the device MUST stop DTAG and reset **TrustState** to 0 (*Idle*).

3.2.6 Other Local Events

None.

3.3 Control Point (Host) Details

In addition to the protocol details specified in section 3.1, the following details are applied to the **control point**.

3.3.1 Abstract Data Model

None.

3.3.2 Timers

None.

Copyright © 2017 Microsoft Corporation

3.3.3 Initialization

None.

3.3.4 Message Processing Events and Sequencing Rules

3.3.4.1 Exchange Response

This response is supported only when **TrustState** is 1 (*Exchanging*). On this response, the following checks MUST be performed:

1. The <DeviceID>, <DeviceCertificate>, and <DeviceConfirmAuthenticator> elements MUST be syntactically validated.
2. The <DeviceCertificate> (**_DeviceCertificate**) MAY be validated as per any vendor-defined rules.

If successful, the **control point**:

1. MUST locally save the values of **_DeviceID**, **_DeviceCertificate**, and **_DeviceConfirmAuthenticator**.
2. MUST set **Iter** to 1.
3. MUST generate **_HostValidateNonceIter**.
4. MUST change **TrustState** from 1 (*Exchanging*) to 2 (*Committing*).
5. MUST set the following elements and send them in a Commit Message (section 3.2.4.2.1.1):
   - <HostID>, as acquired in section 3.1.4.1.
   - <Iteration>, as of the current **Iter** value.
   - <HostValidateAuthenticator>, an **HMAC**, as specified in section 3.1.1, calculated as:

     Base64(HMAC(**_HostValidateNonceIter**, **UTF-8**(**Iter** + **OTPIter** + **_HostID** + **_HostCertificate**)).

If this **action** fails, the control point MUST change **TrustState** to 0 (*Idle*), cancel the DTAG protocol, and report an error to the control point user of this protocol.

3.3.4.2 Commit Response

This response is supported only when **TrustState** is 2 (*Committing*). On this response, the following checks MUST be performed:

1. The <DeviceValidateAuthenticator> element MUST be syntactically validated.

If successful, the **service**:

1. MUST change **TrustState** from 2 (*Committing*) to 3 (*Validating*).
2. MUST set the following element and send them in a Validate Message (section 3.2.4.3.1.1):
   - <HostID>, acquired as specified in section 3.1.4.1.
   - <Iteration>, as the current **Iter** value.
   - <HostValidateNonce>, as the current **_HostValidateNonceIter** value.

Copyright © 2017 Microsoft Corporation

If this **action** fails, the **control point** MUST change **TrustState** to 0 (*Idle*), cancel the DTAG protocol, and report an error to the control point user of this protocol.

3.3.4.3 Validate Response

This **action** is supported only when **TrustState** is 3 (*Validating*). On this action, the following checks MUST be performed:

1. The <DeviceValidateNonce> (**_DeviceValidateNonceIter**) element MUST be syntactically validated.
2. The <Iteration> number MUST be equal to the device's current iteration number, **Iter**.
3. The value of **HMAC**(**_DevicetValidateNonceIter**, **UTF-8**(**Iter** + **OTPIter** + **_DeviceID** + **_DeviceCertificate**)), calculated as specified in section 3.1.1, MUST match the **_DeviceValidateAuthenticatorIter** obtained in the **Commit** response.

If successful, and this is *not* the last iteration, the **service**:

1. MUST increment the iteration number, **Iter**.
2. MUST change **TrustState** from 3 (*Validating*) to 2 (*Committing*).
3. MUST set the following elements and send them in a Commit Message (section 3.2.4.2.1.1):
   - <HostID>, as acquired in section 3.1.4.1.
   - <Iteration>, as the new **Iter** value.
   - <HostValidateAuthenticator>, an HMAC, as specified in section 3.1.1, calculated as:

     Base64(HMAC(**_HostValidateNonceIter**, UTF-8 (**Iter** + **OTPIter** + **_HostID** + **_HostCertificate**)).

If successful and this is the last iteration, the service:

1. MUST increment the iteration number, **Iter**.
2. MUST change **TrustState** from 3 (*Validating*) to 4 (*Committing*).
3. MUST set the following elements and send them in a Confirm Message (section 3.2.4.4.1.1):
   - <HostID>, as acquired in section 3.1.4.1.
   - <IterationsRequired>, as acquired in section 3.1.4.1.
   - <HostConfirmNonce>, as used in section 3.2.4.4.2.1.

If this action fails, the **control point** MUST change **TrustState** to 0 (*Idle*), cancel the DTAG protocol, and report an error to the control point user of this protocol.

3.3.4.4 Confirm Response

This **action** is supported only when **TrustState** is 4 (*Confirming*). On this action, the following checks MUST be performed:

1. The <DeviceConfirmNonce> (**_DeviceConfirmNonce**) element MUST be syntactically validated.
2. The value of **HMAC**(**_DeviceConfirmNonce**, **UTF-8**(**N** + **OTP** + **_DeviceID** + **_DeviceCertificate**)), calculated as specified in section 3.1.1, MUST match the **_DeviceValidateAuthenticator** obtained in the **Exchange** response.

Copyright © 2017 Microsoft Corporation

If successful then trust has been established, and the **service**:

1. MUST store **_DeviceID** and **_DeviceCertificate** in a tamper-proof, persistent store.
2. MUST change **TrustState** from 4 (*Confirming*) to 0 (*Idle*).
3. MUST report the success to the **control point** user of this protocol.

If this action fails, the control point MUST change **TrustState** to 0 (*Idle*), cancel the DTAG protocol, and report an error to the control point user of this protocol.

3.3.4.5 One-time Password (OTP) Event

In addition to the local events specified in section 3.1.4.1, the **control point**:

1. MUST generate **_HostConfirmNonce**.
2. MUST set the following elements and send them in an Exchange Message (section 3.2.4.1.1.1):
   - <HostID>, as acquired in section 3.1.4.1.
   - <HostCertificate>, as acquired in section 3.1.4.1.
   - <HostConfirmAuthenticator>, an **HMAC** as specified in section 3.1.1, calculated as:

     Base64(HMAC(**_HostConfirmNonce**, **UTF-8** (**N**+ **OTP**+ **_HostID**+ **_HostCertificate**)).

3.3.5 Timer Events

None.

3.3.6 Other Local Events

None.

Copyright © 2017 Microsoft Corporation

4 Protocol Examples

4.1 Trust Channel Establishment

This subsection demonstrates a sequence of DTAG messages for a successful exchange of **certificates** between the **device** and the **control point** when the out-of-band one-time password (OTP) is "7495".

4.1.1 Exchange Action Message

Upon the receipt of an out-of-band **one-time password (OTP)**, the **control point** sends the Exchange **SOAP** request message to the **device**. The following example demonstrates an Exchange message where the <HostCertificate> and <HostConfirmAuthenticator> elements are encoded in a Multipurpose Internet Mail Extensions (MIME) **base64** scheme and the requested iteration for the **Commit**-**Validate** **action** is four:

```
<?xml version="1.0"?>
  <SOAP-ENV:Envelope xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
      <m:Exchange xmlns:m="urn:schemas-microsoft-com:service:mstrustagreement:1">
        <HostID xmlns:dt="urn:schemas-upnp-org:service-1-0" dt:dt="string">
          uuid:fe8a7384-68fe-40fd-8996-ff49e24d7e9d
        </HostID>
        <HostCertificate xmlns:dt="urn:schemas-upnp-org:service-1-0" dt:dt="string">

AAABAANiMIIDXjCCAkagAwIBAgIQbrzUER96qKVGrYOAysWLpzANBgkqhkiG9w0BAQUFADA3MTUwMwYDVQQD
EyxNaWNyb3NvZnQgV2luZG93cyBNZWRpYSBDZW50ZXIgRXh0ZW5kZXIgSG9zdDAeFw0wOTA5MTAxNzIzNTZa
Fw0zOTA5MTAyMzQ2NTlaMDcxNTAzBgNVBAMTLE1pY3Jvc29mdCBXaW5kb3dzIE1lZGlhIENlbnRlciBFeHRl
bmRlciBIb3N0MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAow8imH/kXu9c0pCrm3UpoW29AY5Y
R3o3W3oCaeFwyMaj6YNQsuPj7GymtOwX65rUE61FfN5NuI6TrtsN+FHre1L11r3yyeQGhsTsnc3nrCNNh5ZJ
QuWotzeAWrvXmRkbPy6IEqMkGjkpq9v1O4Ugyn+KLGpaonBlM9AnSzu20h7hJqiBKG4XQHeRYLpWhgOk7xdm
gr8hGzacdjqdEYbL2FGxlRMhzPsswL5bqhIgz/KmyZv39V7xtHOMEQRyed4lQrsH+KD+8daXm2JQnayH0TaME
aggKz4eMEIEArX4a8LxRNk0xTWkinsJ5xfUZzyUZ8BPXygksQkP9uxpoFsFJwIDAQABo2YwZDA0BgNVHREELT
Arhil1dWlkOmZlOGE3Mzg0LTY4ZmUtNDBmZC04OTk2LWZmNDllMjRkN2U5ZDALBgNVHQ8EBAMCBPAwHwYDVR0
lBBgwFgYIKwYBBQUHAwEGCisGAQQBgjcKBQwwDQYJKoZIhvcNAQEFBQADggEBAI07K/9Pjxp4CLP8qitnlcE3
MbX6c4BH8oVwRWlazM7tOL7GKqgDAKOiAIA0y/MSkGB0IMOaVHLVosOjxA1sCX6EdVoTeL2abvBww/nrxXSDA
7KrVsmT3VP39vnD67YYacLEfLJtCGDN1HWWTLEOXy3pG+Dn/0ueVezwEv466TQaQgxq4J3oAjVaxxZ/8xpUEL
bJoGiJJs2+QHjsZHZatV1kTUtnlRjXz77P3/NdKVHsPXW1FmzDTl9Ao1udhyL9q57/iHB3doy1goA3xwkqb2Q
ArbWZ5rFlNHCemmfow8iboPdazRhju7b6n7/hCt3S9XOrLhXDV/ghD2XohfCYWXo=
        </HostCertificate>
        <IterationsRequired xmlns:dt="urn:schemas-upnp-org:service-1-0" dt:dt="ui1">
          4
        </IterationsRequired>
        <HostConfirmAuthenticator xmlns:dt="urn:schemas-upnp-org:service-1-0" dt:dt="string">
          rjVF9BZrc+pGmkffVDRk4fIpjFc=
        </HostConfirmAuthenticator>
      </m:Exchange>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4.1.2 Exchange Response Message

If the Exchange message is verified without error, the **device** sends an ExchangeResponse **SOAP** response message to the **control point**. The following example demonstrates an ExchangeResponse message where the <DeviceCertificate> and <DeviceConfirmAuthenticator> elements are encoded in a MIME **base64** scheme:

```
<s:Envelope xmlns:s=http://schemas.xmlsoap.org/soap/envelope/
  s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```


Copyright © 2017 Microsoft Corporation

```
  <s:Body>
    <u:ExchangeResponse xmlns:u="urn:schemas-microsoft-com:service:mstrustagreement:1">
      <DeviceID>
        uuid:20000000-0000-0000-0200-00125A846322
      </DeviceID>
      <DeviceCertificate>

AAABAAPYMIID1DCCA0GgAwIBAgIHElqEYyIAATAJBgUrDgMCHQUAMB0xGzAZBgNVBAMTEk1pY3Jvc29mdCBYQk9YIDM2
MDAeFw0wNTExMTYxODQ0NDFaFw0yNTExMTYxNTAwMDNaMB0xGzAZBgNVBAMTEk1pY3Jvc29mdCBYQk9YIDM2MDCBnzAN
BgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAtjmbxABdzTP45zZ05zryhSQpWCPqq6qZ9BKHopG9u1/xBudClbtrfjQNsyE0
/JriFGWiOnD3kJHML9ONJylBtxOLXWy+ryyxrXUaDaABch1LVcHsPQjR65JHUj9KDFA/ZD7DBv0iFF1A7aU2P7PUsvEi
LRUv23Tr3BG5oH8xCvUCAwEAAaOCAiQwggIgMDQGA1UdEQQtMCuGKXV1aWQ6MjAwMDAwMDAtMDAwMC0wMDAwLTAyMDAt
MDAxMjVBODQ2MzIyMAsGA1UdDwQEAwIE8DAfBgNVHSUEGDAWBggrBgEFBQcDAgYKKwYBBAGCNwoFDDCCAbgGCisGAQQB
gjc3AQEEggGoAagDGEYyJFg4MDM5NTUtMDAyAAAAAAAAAAAAATAzLTA0LTA2AAEAAdwRuaB/MQr18SItFS/bdOtA7aU2
P7PUsmQ+wwb9IhRdkkdSP0oMUD9LVcHsPQjR6611Gg2gAXIdE4tdbL6vLLHML9ONJylBtxRlojpw95CRNA2zITT8muLx
BudClbtrfvQSh6KRvbtfJClYI+qrqpn45zZ05zryhbY5m8QAXc0zw5S8ShVnyroKIPfJwCOlZeTkZK9Qgl7R26VkO3n8
Ol4qg3AJsHDll9Cw/HkE/L+N9QVUOe0iwLH7LG7P27rhu15ytY1XJUVVkmzhxTUPPsicKanij/5YvhYBFYS/8OGWMgrg
Wj0APXFt2a2OlLK7f5Gvx50l+DLdDUapAd+fIe+YJDjForQutlaH79bkecdZEdPTMabjYWRaZkuGBTfaBPnClgHToOiZ
0eRx3HKAO9Uz0AEMSPVwM/uSqw+ZpM6I2ioy8FmmcT7JkbIKh5+nLpL0mz+E29tJ/guDLrBuQD8JwuZwQjjCka/f73hA
vx4Qvwqs72vfhvzFzNNhAVb1JDAJBgUrDgMCHQUAA4GBADe1jvbLfV5zW78fER/T7OP05TL3pXDgwJXBCpKZOL1F1jMS
SSlWhdYSperuFbnmmExrMva/KXP2x7LXVXAL627bAUBReAcn5/qHcCy9/LMHl5WsfYfndpcCr9J1uRM409Qs3iza6a4b
8+QDRIqjkdOB8U2FFWRkOg8puQQ5xl6q
      </DeviceCertificate>
      <DeviceConfirmAuthenticator>
        4W3o9KV4SGjcRhdOh07W8HWxtM8=
      </DeviceConfirmAuthenticator>
    </u:ExchangeResponse>
  </s:Body>
</s:Envelope>
```

4.1.3 Commit Action Message

If the ExchangeResponse message is verified without error, the **control point** sends a Commit **SOAP** request message to the **device**. The following example demonstrates a Commit message where the <HostValidateAuthenticator> element is encoded in a MIME **base64** scheme:

```
<?xml version="1.0"?>
  <SOAP-ENV:Envelope xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
      <m:Commit xmlns:m="urn:schemas-microsoft-com:service:mstrustagreement:1">
        <HostID xmlns:dt="urn:schemas-upnp-org:service-1-0" dt:dt="string">
          uuid:fe8a7384-68fe-40fd-8996-ff49e24d7e9d
        </HostID>
        <Iteration xmlns:dt="urn:schemas-upnp-org:service-1-0" dt:dt="ui1">
          1
        </Iteration>
        <HostValidateAuthenticator xmlns:dt="urn:schemas-upnp-org:service-1-0"
dt:dt="string">
          XI2NfwU5RdKuwgnkrF8MK7jAPPw=
        </HostValidateAuthenticator>
      </m:Commit>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

4.1.4 Commit Response Message

If the Commit message is verified without error, the **device** returns a CommitResponse **SOAP** response message to the **control point**. The following example demonstrates a CommitResponse message where the <DeviceValidateAuthenticator> element is encoded in a MIME **base64** format:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
```


Copyright © 2017 Microsoft Corporation

```
    s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:CommitResponse xmlns:u="urn:schemas-microsoft-com:service:mstrustagreement:1">
      <DeviceValidateAuthenticator>
        9x7dIZOLWXOqLlmrlSAOVn1NNZ8=
      </DeviceValidateAuthenticator>
    </u:CommitResponse>
  </s:Body>
</s:Envelope>
```

4.1.5 Validate Action Message

If the CommitResponse message is verified without error, the **control point** sends a Validate **SOAP** request message to the **device**. The following example demonstrates a Validate message where the <HostValidateNonce> element is encoded in a MIME **base64** scheme:

```
<?xml version="1.0"?>
  <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
      <m:Validate xmlns:m="urn:schemas-microsoft-com:service:mstrustagreement:1">
        <HostID xmlns:dt="urn:schemas-upnp-org:service-1-0" dt:dt="string">
          uuid:fe8a7384-68fe-40fd-8996-ff49e24d7e9d
        </HostID>
        <Iteration xmlns:dt="urn:schemas-upnp-org:service-1-0" dt:dt="ui1">
          1
        </Iteration>
        <HostValidateNonce xmlns:dt="urn:schemas-upnp-org:service-1-0" dt:dt="string">
          NOq7xF1ppNMO7+mPVkyLGKfZTIo=
        </HostValidateNonce>
      </m:Validate>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

4.1.6 Validate Response Message

If the Validate message is verified without error, the **device** returns a ValidateResponse **SOAP** response message to the **control point**. The following example demonstrates a CommitResponse message where the <DeviceValidateNonce> is encoded in a MIME **base64** format:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:ValidateResponse xmlns:u="urn:schemas-microsoft-com:service:mstrustagreement:1">
      <DeviceValidateNonce>
        i1p7FF8Ji3spxKOKR1Td9tTBRrk=
      </DeviceValidateNonce>
    </u:ValidateResponse>
  </s:Body>
</s:Envelope>
```

Because the requested iterations are four at the previous **Exchange** **action**, as described in section 4.1.1, the **Commit** and **Validate** actions will be repeated four times.

4.1.7 Confirm Action Message

If the ValidateResponse message is verified without error, the **control point** sends a Confirm **SOAP** request message to the **device**. The following example demonstrates a Confirm message where the <HostConfirmNonce> element is encoded in a MIME **base64** scheme:

Copyright © 2017 Microsoft Corporation

```
<?xml version="1.0"?>
  <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <S:Body>
      <m:Confirm xmlns:m="urn:schemas-microsoft-com:service:mstrustagreement:1">
        <HostID xmlns:dt="urn:schemas-upnp-org:service-1-0" dt:dt="string">
          uuid:fe8a7384-68fe-40fd-8996-ff49e24d7e9d
        </HostID>
        <IterationsRequired xmlns:dt="urn:schemas-upnp-org:service-1-0" dt:dt="ui1">
          4
        </IterationsRequired>
        <HostConfirmNonce xmlns:dt="urn:schemas-upnp-org:service-1-0" dt:dt="string">
          5GDSOp5h92XrL9CMfvdEUfcWkAE=
        </HostConfirmNonce>
      </m:Confirm>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

4.1.8 Confirm Response Message

If the Confirm message is verified without error, the **device** returns a ConfirmResponse **SOAP** response message to the **control point**. The following example demonstrates a ConfirmResponse message where the <DeviceConfirmNonce> element is encoded in a MIME **base64** scheme:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:ConfirmResponse xmlns:u="urn:schemas-microsoft-com:service:mstrustagreement:1">
      <DeviceConfirmNonce>
        rD5m4Fgi+ifV9GS+611aO3T998Q=
      </DeviceConfirmNonce>
    </u:ConfirmResponse>
  </s:Body>
</s:Envelope>
```

After the control point verifies the response message, and if there is no error, DTAG is completed and a trust relationship is established between the control point and the device.

4.2 Error Message

If an error occurs while the **device** processes any request message, the device returns an error message instead of the response message to the **control point**. The following example demonstrates an error message on the **Validate** **action**, which indicates error code 803 (invalid **nonce**):

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <s:Fault>
      <faultcode>s:Client</faultcode>
      <faultstring>UPnPError</faultstring>
      <detail>
        <UPnPError xmlns="urn:schemas-upnp-org:control-1-0">
          <errorCode>803</errorCode>
          <errorDescription>Invalid Nonce</errorDescription>
        </UPnPError>
      </detail>
    </s:Fault>
  </s:Body>
</s:Envelope>
```


Copyright © 2017 Microsoft Corporation

5 Security

5.1 Security Considerations for Implementers

In general, DTAG provides protection at the strength of the **one-time password (OTP)**, where the OTP is required to be:

- Cryptographically random and difficult to guess.
- Transported to the **endpoints** in an out-of-band manner, such as through user interaction, the details of which are not described in this specification. For this purpose, the OTP can be relatively short enough for the user to remember.
- Generated anew each time DTAG is started or restarted.
- The number of OTP characters is required to be equal to or greater than the number of iterations.

The number of validate rounds (**N**) is required to be at least 2, with a minimum of 4 recommended.

5.2 Index of Security Parameters

Security parameter	Section
One-time Password (OTP)	3.1.4.1
N	3.1.1

Copyright © 2017 Microsoft Corporation

6 Appendix A: Full WSDL

This protocol does not contain a **WSDL**. For **UPnP**, the equivalent to WSDL are the UPnP device and **service descriptions**. Please see the UPnP device description in section 7 and the full UPnP service description in section 8.

Copyright © 2017 Microsoft Corporation

7 Appendix B: UPnP Device Description

The following is a sample **service** information of DTAG, which the **device** description has to include as part of the service list for the device.

The default namespace, "urn:schemas-upnp-org:device-1-0", is specified in [UPNPARCH1] sections 2.1 and 2.6.

```
<?xml version='1.0'?>
<root xmlns="urn:schemas-upnp-org:device-1-0"
         xmlns:pnpx="http://schemas.microsoft.com/windows/pnpx/2005/11" >
 <specVersion>
      <major>1</major>
      <minor>0</minor>
    </specVersion>
 <device>
<pnpx:X deviceCategory>MediaDevices</pnpx:X deviceCategory>
  <deviceType>urn:schemas-microsoft-com:device:MediaCenterExtenderMFD:1</deviceType>
  <friendlyName>Xbox 360 Media Center Extender</friendlyName>
  <manufacturer>Microsoft Corporation</manufacturer>
  <manufacturerURL>http://www.xbox.com/</manufacturerURL>
  <modelDescription>Xbox 360 Media Center Extender</modelDescription>
  <modelName>Xbox 360</modelName>
  <modelNumber></modelNumber>
  <modelURL>http://go.microsoft.com/fwlink/?LinkID=53081</modelURL>
  <serialNumber></serialNumber>
  <UDN>uuid:10000000-0000-0000-0200-00125A702E78</UDN>
  <UPC></UPC>
  <iconList>
        <icon>
          <mimetype>image/jpeg</mimetype>
          <width>48</width>
          <height>48</height>
          <depth>24</depth>
          <url>/IconSM.jpg</url>
        </icon>
        <icon>
          <mimetype>image/jpeg</mimetype>
          <width>120</width>
          <height>120</height>
          <depth>24</depth>
          <url>/IconLRG.jpg</url>
        </icon>
        <icon>
          <mimetype>image/png</mimetype>
          <width>48</width>
          <height>48</height>
          <depth>24</depth>
          <url>/IconSM.png</url>
        </icon>
        <icon>
          <mimetype>image/png</mimetype>
          <width>120</width>
          <height>120</height>
          <depth>24</depth>
          <url>/IconLRG.png</url>
        </icon>
        <icon>
          <mimetype>image/png</mimetype>
          <width>152</width>
          <height>152</height>
          <depth>24</depth>
          <url>/IconMCE.png</url>
        </icon>
      </iconList>
  <serviceList>
        <service>
```


Copyright © 2017 Microsoft Corporation

```
            <serviceType>urn:schemas-microsoft-com:service:NULL:1</serviceType>
            <serviceId>urn:microsoft-com:serviceId:NULL</serviceId>
            <SCPDURL>/XD/NULL.xml</SCPDURL>
            <controlURL>/UD/?0</controlURL>
            <eventSubURL/>
          </service>
        </serviceList>
    <deviceList>
        <device xmlns:mcx="http://schemas.microsoft.com/windows/mcx/2007/06"
xmlns:nss="urn:schemas-microsoft-com:WMPNSS-1-0">
      <pnpx:X compatibleId>MICROSOFT MCX 0001</pnpx:X compatibleId>
      <pnpx:X_deviceCategory>MediaDevices</pnpx:X_deviceCategory>
      <mcx:pakVersion>dv2.0.0</mcx:pakVersion>
      <mcx:supportedHostVersions>pc2.0.0</mcx:supportedHostVersions>
      <nss:X_magicPacketSendSupported>1</nss:X_magicPacketSendSupported>
      <deviceType>urn:schemas-microsoft-com:device:MediaCenterExtender:1</deviceType>
      <friendlyName>Xbox 360 Media Center Extender</friendlyName>
      <manufacturer>Microsoft Corporation</manufacturer>
      <manufacturerURL>http://www.microsoft.com/</manufacturerURL>
      <modelDescription>Xbox 360 Media Center Extender</modelDescription>
      <modelName>Xbox 360</modelName>
      <modelNumber></modelNumber>
      <modelURL>http://go.microsoft.com/fwlink/?LinkID=53081</modelURL>
      <serialNumber></serialNumber>
      <UDN>uuid:20000000-0000-0000-0200-00125A702E78</UDN>
      <UPC></UPC>
      <iconList>
          <icon>
            <mimetype>image/jpeg</mimetype>
            <width>48</width>
            <height>48</height>
            <depth>24</depth>
            <url>/IconSM.jpg</url>
          </icon>
          <icon>
            <mimetype>image/jpeg</mimetype>
            <width>120</width>
            <height>120</height>
            <depth>24</depth>
            <url>/IconLRG.jpg</url>
          </icon>
          <icon>
            <mimetype>image/png</mimetype>
            <width>48</width>
            <height>48</height>
            <depth>24</depth>
            <url>/IconSM.png</url>
          </icon>
          <icon>
            <mimetype>image/png</mimetype>
            <width>120</width>
            <height>120</height>
            <depth>24</depth>
            <url>/IconLRG.png</url>
          </icon>
          <icon>
            <mimetype>image/png</mimetype>
            <width>152</width>
            <height>152</height>
            <depth>24</depth>
            <url>/IconMCE.png</url>
          </icon>
        </iconList>
      <serviceList>
          <service>
           <serviceType>urn:schemas-microsoft-com:service:mstrustagreement:1</serviceType>
            <serviceId>urn:microsoft-com:serviceId:MSTA</serviceId>
            <SCPDURL>/XD/mstrustagreement.xml</SCPDURL>
            <controlURL>/UD/?1</controlURL>
```


Copyright © 2017 Microsoft Corporation

```
            <eventSubURL />
          </service>
        </serviceList>
      </device>
    </deviceList>
  </device>
</root>
```


Copyright © 2017 Microsoft Corporation

8 Appendix C: Full UPnP Service Description

The following is a sample **service description** of DTAG, which the **device** has to publish as a prerequisite before DTAG can take any **action**, as described in section 1.5.

The default namespace, "urn:schemas-upnp-org:service-1-0", is specified in [UPNPARCH1] sections 2.3 and 2.7.

```
<?xml version="1.0" ?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
<specVersion>
      <major>1</major>
  <minor>0</minor>
  </specVersion>
<actionList>
<action>
      <name>Exchange</name>
<argumentList>
    <argument>
  <name>HostID</name>
  <direction>in</direction>
  <relatedStateVariable>A ARG TYPE EndpointID</relatedStateVariable>
    </argument>
    <argument>
        <name>HostCertificate</name>
        <direction>in</direction>
        <relatedStateVariable>A_ARG_TYPE_Certificate</relatedStateVariable>
    </argument>
    <argument>
        <name>IterationsRequired</name>
        <direction>in</direction>
        <relatedStateVariable>A_ARG_TYPE_Rounds</relatedStateVariable>
    </argument>
        <argument>
        <name>HostConfirmAuthenticator</name>
        <direction>in</direction>
        <relatedStateVariable>A ARG TYPE Authenticator</relatedStateVariable>
      </argument>
        <argument>
        <name>DeviceID</name>
        <direction>out</direction>
        <relatedStateVariable>A_ARG_TYPE_EndpointID</relatedStateVariable>
      </argument>
        <argument>
        <name>DeviceCertificate</name>
        <direction>out</direction>
        <relatedStateVariable>A ARG TYPE Certificate</relatedStateVariable>
        </argument>
        <argument>
        <name>DeviceConfirmAuthenticator</name>
        <direction>out</direction>
        <relatedStateVariable>A_ARG_TYPE_Authenticator</relatedStateVariable>
        </argument>
  </argumentList>
  </action>
    <action>
  <name>Commit</name>
    <argumentList>
        <argument>
        <name>HostID</name>
        <direction>in</direction>
        <relatedStateVariable>A_ARG_TYPE_EndpointID</relatedStateVariable>
        </argument>
        <argument>
        <name>Iteration</name>
        <direction>in</direction>
        <relatedStateVariable>A_ARG_TYPE_Iteration</relatedStateVariable>
```


Copyright © 2017 Microsoft Corporation

```
        </argument>
          <argument>
          <name>HostValidateAuthenticator</name>
          <direction>in</direction>
          <relatedStateVariable>A_ARG_TYPE_Authenticator</relatedStateVariable>
        </argument>
          <argument>
          <name>DeviceValidateAuthenticator</name>
          <direction>out</direction>
          <relatedStateVariable>A_ARG_TYPE_Authenticator</relatedStateVariable>
          </argument>
      </argumentList>
      </action>
        <action>
        <name>Validate</name>
        <argumentList>
           <argument>
             <name>HostID</name>
           <direction>in</direction>
           <relatedStateVariable>A_ARG_TYPE_EndpointID</relatedStateVariable>
       </argument>
          <argument>
           <name>Iteration</name>
           <direction>in</direction>
           <relatedStateVariable>A_ARG_TYPE_Iteration</relatedStateVariable>
          </argument>
          <argument>
           <name>HostValidateNonce</name>
           <direction>in</direction>
           <relatedStateVariable>A_ARG_TYPE_Nonce</relatedStateVariable>
          </argument>
          <argument>
           <name>DeviceValidateNonce</name>
           <direction>out</direction>
           <relatedStateVariable>A_ARG_TYPE_Nonce</relatedStateVariable>
          </argument>
      </argumentList>
      </action>
        <action>
        <name>Confirm</name>
        <argumentList>
           <argument>
            <name>HostID</name>
            <direction>in</direction>
            <relatedStateVariable>A_ARG_TYPE_EndpointID</relatedStateVariable>
       </argument>
          <argument>
           <name>IterationsRequired</name>
           <direction>in</direction>
           <relatedStateVariable>A_ARG_TYPE_Rounds</relatedStateVariable>
          </argument>
          <argument>
           <name>HostConfirmNonce</name>
           <direction>in</direction>
           <relatedStateVariable>A_ARG_TYPE_Nonce</relatedStateVariable>
       </argument>
          <argument>
           <name>DeviceConfirmNonce</name>
           <direction>out</direction>
           <relatedStateVariable>A_ARG_TYPE_Nonce</relatedStateVariable>
       </argument>
       </argumentList>
      </action>
    </actionList>
      <serviceStateTable>
        <stateVariable sendEvents="no">
           <name>TrustState</name>
             <dataType>ui1</dataType>
          <allowedValueRange>
```


Copyright © 2017 Microsoft Corporation

```
          <minimum>0</minimum>
            <maximum>4</maximum>
       </allowedValueRange>
    </stateVariable>
      <stateVariable sendEvents="no">
           <name>A ARG TYPE Rounds</name>
         <dataType>ui1</dataType>
         <allowedValueRange>
            <minimum>2</minimum>
            <maximum>20</maximum>
       </allowedValueRange>
    </stateVariable>
      <stateVariable sendEvents="no">
         <name>A_ARG_TYPE_Iteration</name>
            <dataType>ui1</dataType>
         <allowedValueRange>
            <minimum>1</minimum>
            <maximum>20</maximum>
       </allowedValueRange>
       </stateVariable>
      <stateVariable sendEvents="no">
         <name>A_ARG_TYPE_EndpointID</name>
           <dataType>string</dataType>
    </stateVariable>
      <stateVariable sendEvents="no">
           <name>A_ARG_TYPE_Authenticator</name>
           <dataType>string</dataType>
    </stateVariable>
      <stateVariable sendEvents="no">
           <name>A_ARG_TYPE_Nonce</name>
           <dataType>string</dataType>
    </stateVariable>
      <stateVariable sendEvents="no">
           <name>A ARG TYPE Certificate</name>
           <dataType>string</dataType>
    </stateVariable>
  </serviceStateTable>
  </scpd>
```


Copyright © 2017 Microsoft Corporation

9 Appendix D: Product Behavior

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

- Windows Vista operating system
- Windows 7 operating system
- Windows 8 operating system
- Windows 8.1 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms "SHOULD" or "SHOULD NOT" implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term "MAY" implies that the product does not follow the prescription.

Copyright © 2017 Microsoft Corporation

10 Change Tracking

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

Copyright © 2017 Microsoft Corporation

11 Index

Copyright © 2017 Microsoft Corporation

I

L

M

N

O

P

R

S

T

Copyright © 2017 Microsoft Corporation

U

V

W

Copyright © 2017 Microsoft Corporation