



ISO/IEC 29341-13-10

Edition 1.0 2008-11

INTERNATIONAL STANDARD

**Information technology – UPnP Device Architecture –
Part 13-10: Device Security Device Control Protocol – Device Security Service**



THIS PUBLICATION IS COPYRIGHT PROTECTED
Copyright © 2008 ISO/IEC, Geneva, Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester.

If you have any questions about ISO/IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

IEC Central Office
3, rue de Varembe
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub

The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,...). It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub

Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.

- Electropedia: www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.

- Customer Service Centre: www.iec.ch/webstore/custserv

If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:

Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00



ISO/IEC 29341-13-10

Edition 1.0 2008-11

INTERNATIONAL STANDARD

**Information technology – UPnP Device Architecture –
Part 13-10: Device Security Device Control Protocol – Device Security Service**

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

PRICE CODE



ICS 35.200

ISBN 2-8318-1012-8

CONTENTS

FOREWORD	5
ORIGINAL UPNP DOCUMENTS (informative)	7
1. Overview and Scope	9
1.1. Acknowledgements	11
2. Service Modeling Definitions	12
2.1. Service Type	12
2.2. Namespaces	12
2.3. Referenced Specifications	12
2.4. MustUnderstand	12
2.5. State Variables	13
2.5.1. NumberOfOwners	13
2.5.2. LifetimeSequenceBase	13
2.5.3. TimeHint	14
2.5.4. TotalACLSize	14
2.5.5. FreeACLSize	14
2.5.6. TotalOwnerListSize	14
2.5.7. FreeOwnerListSize	14
2.5.8. TotalCertCacheSize	14
2.5.9. FreeCertCacheSize	14
2.5.10. A_ARG_TYPE_string	14
2.5.11. A_ARG_TYPE_base64	14
2.5.12. A_ARG_TYPE_int	14
2.5.13. A_ARG_TYPE_boolean	15
2.6. Eventing and Moderation	15
2.7. Actions	16
2.8. Cryptographic Notation for Selected Actions	17
2.9. Actions Invoked by Both CP and SC	17
2.9.1. GetPublicKeys	17
2.9.2. GetAlgorithmsAndProtocols	18
2.9.3. GetACLSizes	19
2.9.4. CacheCertificate	20
2.9.5. SetTimeHint	22
2.9.6. GetLifetimeSequenceBase	23
2.9.7. SetSessionKeys	24
2.9.8. ExpireSessionKeys	26
2.9.9. DecryptAndExecute	27
2.10. Actions Invoked by SC only	28
2.10.1. TakeOwnership	28
2.10.2. GetDefinedPermissions	30
2.10.3. GetDefinedProfiles	31
2.10.4. ReadACL	33
2.10.5. WriteACL	34
2.10.6. AddACLEntry	35
2.10.7. DeleteACLEntry	36
2.10.8. ReplaceACLEntry	37
2.10.9. FactorySecurityReset	38
2.10.10. GrantOwnership	39
2.10.11. RevokeOwnership	40
2.10.12. ListOwners	41
2.11. Relationships among Actions	43

2.11.1.	Relationships among Actions invoked by Security Console.....	43
2.11.2.	Relationships among Actions invoked by normal Control Point.....	43
2.11.3.	ACLVersion	44
2.12.	Common Error Codes	45
3.	Supporting Information.....	46
3.1.	Glossary	46
3.2.	XML Strings as UPnP Arguments	46
3.3.	BASE32 Encoding.....	47
3.4.	Namespaces	47
4.	Data Structures	48
4.1.	Namespaces	48
4.2.	Access Control List (ACL) Structure	48
4.2.1.	Note on date and time format: ISO 8601	49
4.3.	Owner List	49
4.4.	Certificates	50
4.4.1.	Authorization Certificate	50
4.4.2.	Name Definition Certificate	51
4.5.	Permission Language	52
4.5.1.	<all>	52
4.5.2.	<set>	52
4.5.3.	<elt>	52
4.5.4.	<prefix>	52
4.5.5.	<range>	52
4.6.	RSA Encryption Padding.....	53
4.6.1.	SetSessionKeys	54
4.6.2.	TakeOwnership.....	54
4.6.3.	Counteracting attacks on PKCS#1 V 1.5 padding	54
4.6.4.	Historical note about padding and padding attacks	55
4.7.	Public Keys and their hashes	55
4.8.	Symmetric cipher mode and padding.....	56
4.9.	Canonical BASE64 Encoding.....	56
5.	Theory of Operation	58
5.1.	Access Control Lists and Certificates.....	58
5.1.1.	ACL and Certificate Processing Model	59
5.2.	Signature block format	59
5.2.1.	Sequence Numbering	61
5.2.2.	Hashing and Canonicalization.....	62
5.2.3.	UPnP Certificate Transport.....	62
5.2.4.	IDs for XML-Signature	63
5.2.5.	Signature Processing Model	63
6.	XML Service Description.....	64
Annex A (normative)	Device Security Schema	71
Annex B (informative)	Security Ceremonies	79
B.1	Background	79
B.2	Security Model.....	79
B.2.1	Security Policy Data	81
B.3	Secure Component Discovery	81
B.3.1	Discovery of Secured Devices	82
B.3.2	Discovery of a Secured CP or SC.....	84

B.4	Ownership	84
B.4.1	TakeOwnership	85
B.4.2	ListOwners	86
B.4.3	GrantOwnership	87
B.4.4	RevokeOwnership	87
B.4.5	FactorySecurityReset	88
B.5	Session Keys	88
B.6	ACL Editing	89
B.7	Certificate caching	90
B.8	References	92

LIST OF TABLES

Table 1: State variable	13
Table 2: Event Moderation	15
Table 3: Actions invoked by both Control Point and Security Console	16
Table 4: Actions invoked by a Security Console only	16

LIST OF FIGURES

Figure B.1: Message Security Flowchart	80
Figure B.2: Device discovery ceremony and TakeOwnership	82
Figure B.3: Discovery of CP and SC nodes	84
Figure B.4: Taking ownership via private cable	85
Figure B.5: ListOwners	86
Figure B.6: GrantOwnership	87
Figure B.7: RevokeOwnership	87
Figure B.8: FactorySecurityReset	88
Figure B.9: Setting Session Keys	89
Figure B.10: ACL Editing	90
Figure B.11: Certificate caching on the device	91
Figure B.12: Delivering certificates to the CP	92

INFORMATION TECHNOLOGY – UPNP DEVICE ARCHITECTURE –

Part 13-10: Device Security Device Control Protocol – Device Security Service

FOREWORD

- 1) ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards. Their preparation is entrusted to technical committees; any ISO and IEC member body interested in the subject dealt with may participate in this preparatory work. International governmental and non-governmental organizations liaising with ISO and IEC also participate in this preparation.
- 2) In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.
- 3) The formal decisions or agreements of IEC and ISO on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC and ISO member bodies.
- 4) IEC, ISO and ISO/IEC publications have the form of recommendations for international use and are accepted by IEC and ISO member bodies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC, ISO and ISO/IEC publications is accurate, IEC or ISO cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
- 5) In order to promote international uniformity, IEC and ISO member bodies undertake to apply IEC, ISO and ISO/IEC publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any ISO/IEC publication and the corresponding national or regional publication should be clearly indicated in the latter.
- 6) ISO and IEC provide no marking procedure to indicate their approval and cannot be rendered responsible for any equipment declared to be in conformity with an ISO/IEC publication.
- 7) All users should ensure that they have the latest edition of this publication.
- 8) No liability shall attach to IEC or ISO or its directors, employees, servants or agents including individual experts and members of their technical committees and IEC or ISO member bodies for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication of, use of, or reliance upon, this ISO/IEC publication or any other IEC, ISO or ISO/IEC publications.
- 9) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

IEC and ISO draw attention to the fact that it is claimed that compliance with this document may involve the use of patents as indicated below.

ISO and IEC take no position concerning the evidence, validity and scope of the putative patent rights. The holders of the putative patent rights have assured IEC and ISO that they are willing to negotiate free licences or licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of the putative patent rights are registered with IEC and ISO.

Intel Corporation has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Intel Corporation
Standards Licensing Department
5200 NE Elam Young Parkway
MS: JFS-98
USA – Hillsboro, Oregon 97124

Microsoft Corporation has informed IEC and ISO that it has patent applications or granted patents as listed below:

6101499 / US; 6687755 / US; 6910068 / US; 7130895 / US; 6725281 / US; 7089307 / US; 7069312 / US;
10/783 524 / US

Information may be obtained from:

Microsoft Corporation
One Microsoft Way
USA – Redmond WA 98052

Philips International B.V. has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Philips International B.V. – IP&S
High Tech campus, building 44 3A21
NL – 5656 Eindhoven

NXP B.V. (NL) has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

NXP B.V. (NL)
High Tech campus 60
NL – 5656 AG Eindhoven

Matsushita Electric Industrial Co. Ltd. has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Matsushita Electric Industrial Co. Ltd.
1-3-7 Shiromi, Chuoh-ku
JP – Osaka 540-6139

Hewlett Packard Company has informed IEC and ISO that it has patent applications or granted patents as listed below:

5 956 487 / US; 6 170 007 / US; 6 139 177 / US; 6 529 936 / US; 6 470 339 / US; 6 571 388 / US; 6 205 466 / US

Information may be obtained from:

Hewlett Packard Company
1501 Page Mill Road
USA – Palo Alto, CA 94304

Samsung Electronics Co. Ltd. has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Digital Media Business, Samsung Electronics Co. Ltd.
416 Maetan-3 Dong, Yeongtang-Gu,
KR – Suwon City 443-742

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. IEC and ISO shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 29341-13-10 was prepared by UPnP Implementers Corporation and adopted, under the PAS procedure, by joint technical committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

The list of all currently available parts of the ISO/IEC 29341 series, under the general title *Universal plug and play (UPnP) architecture*, can be found on the IEC web site.

This International Standard has been approved by vote of the member bodies, and the voting results may be obtained from the address given on the second title page.

ORIGINAL UPnP DOCUMENTS (informative)

Reference may be made in this document to original UPnP documents. These references are retained in order to maintain consistency between the specifications as published by ISO/IEC and by UPnP Implementers Corporation. The following table indicates the original UPnP document titles and the corresponding part of ISO/IEC 29341:

UPnP Document Title	ISO/IEC 29341 Part
UPnP Device Architecture 1.0	ISO/IEC 29341-1
UPnP Basic:1 Device	ISO/IEC 29341-2
UPnP AV Architecture:1	ISO/IEC 29341-3-1
UPnP MediaRenderer:1 Device	ISO/IEC 29341-3-2
UPnP MediaServer:1 Device	ISO/IEC 29341-3-3
UPnP AVTransport:1 Service	ISO/IEC 29341-3-10
UPnP ConnectionManager:1 Service	ISO/IEC 29341-3-11
UPnP ContentDirectory:1 Service	ISO/IEC 29341-3-12
UPnP RenderingControl:1 Service	ISO/IEC 29341-3-13
UPnP MediaRenderer:2 Device	ISO/IEC 29341-4-2
UPnP MediaServer:2 Device	ISO/IEC 29341-4-3
UPnP AV Datastructure Template:1	ISO/IEC 29341-4-4
UPnP AVTransport:2 Service	ISO/IEC 29341-4-10
UPnP ConnectionManager:2 Service	ISO/IEC 29341-4-11
UPnP ContentDirectory:2 Service	ISO/IEC 29341-4-12
UPnP RenderingControl:2 Service	ISO/IEC 29341-4-13
UPnP ScheduledRecording:1	ISO/IEC 29341-4-14
UPnP DigitalSecurityCamera:1 Device	ISO/IEC 29341-5-1
UPnP DigitalSecurityCameraMotionImage:1 Service	ISO/IEC 29341-5-10
UPnP DigitalSecurityCameraSettings:1 Service	ISO/IEC 29341-5-11
UPnP DigitalSecurityCameraStillImage:1 Service	ISO/IEC 29341-5-12
UPnP HVAC_System:1 Device	ISO/IEC 29341-6-1
UPnP HVAC_ZoneThermostat:1 Device	ISO/IEC 29341-6-2
UPnP ControlValve:1 Service	ISO/IEC 29341-6-10
UPnP HVAC_FanOperatingMode:1 Service	ISO/IEC 29341-6-11
UPnP FanSpeed:1 Service	ISO/IEC 29341-6-12
UPnP HouseStatus:1 Service	ISO/IEC 29341-6-13
UPnP HVAC_SetpointSchedule:1 Service	ISO/IEC 29341-6-14
UPnP TemperatureSensor:1 Service	ISO/IEC 29341-6-15
UPnP TemperatureSetpoint:1 Service	ISO/IEC 29341-6-16
UPnP HVAC_UserOperatingMode:1 Service	ISO/IEC 29341-6-17
UPnP BinaryLight:1 Device	ISO/IEC 29341-7-1
UPnP DimmableLight:1 Device	ISO/IEC 29341-7-2
UPnP Dimming:1 Service	ISO/IEC 29341-7-10
UPnP SwitchPower:1 Service	ISO/IEC 29341-7-11
UPnP InternetGatewayDevice:1 Device	ISO/IEC 29341-8-1
UPnP LANDevice:1 Device	ISO/IEC 29341-8-2
UPnP WANDevice:1 Device	ISO/IEC 29341-8-3
UPnP WANConnectionDevice:1 Device	ISO/IEC 29341-8-4
UPnP WLANAccessPointDevice:1 Device	ISO/IEC 29341-8-5
UPnP LANHostConfigManagement:1 Service	ISO/IEC 29341-8-10
UPnP Layer3Forwarding:1 Service	ISO/IEC 29341-8-11
UPnP LinkAuthentication:1 Service	ISO/IEC 29341-8-12
UPnP RadiusClient:1 Service	ISO/IEC 29341-8-13
UPnP WANCableLinkConfig:1 Service	ISO/IEC 29341-8-14
UPnP WANCommonInterfaceConfig:1 Service	ISO/IEC 29341-8-15
UPnP WANDSLLinkConfig:1 Service	ISO/IEC 29341-8-16
UPnP WANEthernetLinkConfig:1 Service	ISO/IEC 29341-8-17
UPnP WANIPConnection:1 Service	ISO/IEC 29341-8-18
UPnP WANPOTSLinkConfig:1 Service	ISO/IEC 29341-8-19
UPnP WANPPPConnection:1 Service	ISO/IEC 29341-8-20
UPnP WLANConfiguration:1 Service	ISO/IEC 29341-8-21
UPnP Printer:1 Device	ISO/IEC 29341-9-1
UPnP Scanner:1.0 Device	ISO/IEC 29341-9-2
UPnP ExternalActivity:1 Service	ISO/IEC 29341-9-10
UPnP Feeder:1.0 Service	ISO/IEC 29341-9-11
UPnP PrintBasic:1 Service	ISO/IEC 29341-9-12
UPnP Scan:1 Service	ISO/IEC 29341-9-13
UPnP QoS Architecture:1.0	ISO/IEC 29341-10-1
UPnP QoSDevice:1 Service	ISO/IEC 29341-10-10
UPnP QoSManager:1 Service	ISO/IEC 29341-10-11
UPnP QoSPolicyHolder:1 Service	ISO/IEC 29341-10-12
UPnP QoS Architecture:2	ISO/IEC 29341-11-1
UPnP QOS v2 Schema Files	ISO/IEC 29341-11-2



UPnP Document Title	ISO/IEC 29341 Part
UPnP QosDevice:2 Service	ISO/IEC 29341-11-10
UPnP QosManager:2 Service	ISO/IEC 29341-11-11
UPnP QosPolicyHolder:2 Service	ISO/IEC 29341-11-12
UPnP RemoteUIClientDevice:1 Device	ISO/IEC 29341-12-1
UPnP RemoteUIServerDevice:1 Device	ISO/IEC 29341-12-2
UPnP RemoteUIClient:1 Service	ISO/IEC 29341-12-10
UPnP RemoteUIServer:1 Service	ISO/IEC 29341-12-11
UPnP DeviceSecurity:1 Service	ISO/IEC 29341-13-10
UPnP SecurityConsole:1 Service	ISO/IEC 29341-13-11

1. Overview and Scope

The Device Security service provides the services necessary for strong authentication, authorization, replay prevention and privacy of UPnP SOAP actions. Under this architecture, a Device enforces its own access control but its access control policy is established and maintained by an administrative application, the Security Console (see SecurityConsole:1), which uses some of the actions provided as part of this service. Nothing prevents a device with the proper user interface capabilities from providing its own administration interface, although presumably it is a valuable ability to administer access from one location for an entire household network. In what follows, the term “Security Console” refers to any Control Point that chooses to exercise the administrative functions defined here in DeviceSecurity.

DeviceSecurity implements access control for itself and for other Services in the same Device (or embedded Device). There are two classes of access control grant defined here: ownership and normal permission. Each security-aware Device has an ownership list capable of holding at least one entry. Any Security Console listed as an owner has full rights to the Device, specifically to all actions including the DeviceSecurity actions that specify other access control. In addition to the owner list, the Device usually has an access control list (ACL) maintained by DeviceSecurity. Entries in the ACL grant a Security Console or other Control Point symbolic permissions that, in turn, grant access to sets of actions. Those permissions typically grant less than the full access that ownership grants. A Security Console, at the option of the Device vendor, might also be granted the permission to delegate rights to others without having to be a full owner of the device or to define named groups of Control Points to be granted access as a group in a single operation. These last two capabilities depend on the implementation of certificate processing by the Device and that is an optional feature of DeviceSecurity.

A device implementing DeviceSecurity will need to support the basic cryptographic algorithms used in this service:

1. AES 128-bit, for symmetric bulk encryption, labeled “AES-128-CBC”, with blocks padded as described in section 4.8 below.
2. SHA1 HMAC, for symmetric signatures and for TakeOwnership, labeled “SHA1-HMAC”
3. RSA 1024-bit, for identification and for establishing secure sessions, and possibly for other operations, labeled “RSA”. In the current version of this DeviceSecurity specification, there is no reason for a device to have a public-key algorithm signature key. Therefore a device will offer a public confidentiality key but does not need to offer a signature key at this time. [In particular, actions that are authorized by public-key algorithm signatures do not have digitally signed responses.] The algorithm identifier “RSA” when used for encryption in this service implies (RSA, PKCS#1) and when used for signing implies (RSA, SHA-1, PKCS#1).

Other algorithms may be added to this list, if these develop flaws and need to be supplanted. For example, if another hash algorithm were to be added, one would need to identify it and also identify RSA using it for signing. So, for example, if one adds the hash algorithm FOOBLA, one would need to add RSA-FOOBLA and FOOBLA-HMAC to the algorithm list.

The logic of operation, from the point of view of the security-aware device, is as follows:

1. If a device has a display or printing capability and also has a source of randomness, then it is preferable for the device to generate a new password and new public key pair on any power-up when it is in factory reset state. It would then display or print the generated password and the hash of the new public key. For devices without those abilities, the password and public key pair will have been created by the manufacturer and will remain constant over the lifetime of the device. In that case, the manufacturer will have printed the password and the Security ID (hash of the device’s public key) on a card or a label attached to the device case, or both. The password does not need to be longer than about 6 upper case alphanumeric characters, provided it was generated randomly for that device and is used by TakeOwnership shortly after the device is plugged into the network. [Note: passwords or keys are not to be used in common across a set of devices. Such usage would be a security flaw.]
2. The device announces itself via SSDP, perhaps giving real details or perhaps describing itself only as “Security Aware Device”, if the device wants to prevent inventory by an unauthorized control point.
3. A Security Console (SC), presumably that of the device’s owner, calls the GetPublicKeys, GetLifetimeSequenceBase and TakeOwnership actions, supplying the device’s password to prove

authorization to take control of the device. As a result of a successful TakeOwnership action, that Security Console is listed as the device's Owner. An Owner is a control point that is empowered to edit the device's Access Control List (ACL). Subsequent TakeOwnership attempts MUST be ignored. If the device generated its password dynamically, then the password used in this TakeOwnership action should not be valid again.

4. The owning SC establishes a set of session keys by calling GetLifetimeSequenceBase and SetSessionKeys. These session keys will be used for digitally signing future action messages using XML-Signature with symmetric key signature.
5. The owning SC will then be free to use the GetACLSizes action to discover whether access permissions can be granted by ACL entries directly or need to be encoded as certificates, because the device has too little room to store an ACL.
6. The owning SC will probably also invoke GetDefinedPermissions in order to learn what permissions it might grant to chosen Control Points.
7. If the SC grants access by certificate, that operation happens without invoking any actions on the device. Otherwise, the SC grants access to desired Control Points (CPs) by way of AddACLEntry. It also reads the current ACL contents, for display to its human operator, by using ReadACL. The actions WriteACL, ReplaceACLEntry and DeleteACLEntry are also available for ACL editing. These actions will be omitted from DeviceSecurity in those implementations that do not provide any memory for an ACL.
8. A CP that wants to conduct a control session with a security aware device may call GetAlgorithmsAndProtocols, in order to confirm interoperability, and will then call SetSessionKeys, to establish a secure session with the device. These sessions do not hold network connections open and could be very long-lived, depending on the storage capacity of the Device and the CP.
9. Once a CP has established a session with the device, it invokes actions by sending normal action messages, digitally signed using XML-Signature with a symmetric signature key (e.g., HMAC) established during SetSessionKeys and with a sequence number initialized by that action. The sequence numbers during a session must be monotonically increasing but need not be sequential.
10. If a CP needs an action message to be confidential, it uses the DecryptAndExecute action, one argument of which is the ciphertext of an encrypted action. The reply from that action is then encrypted and returned in the reply to the DecryptAndExecute action.
11. A session may be ended intentionally, by ExpireSessionKeys, or may time out at the device's discretion.
12. In the event that the device's owner wants to share ownership, either with another person or (for fault-tolerance) with another SC operated by him- or her-self, a current owning SC can invoke GrantOwnership. This assumes that GetACLSizes shows that there is room in the device to record the additional owner. Ownership can be revoked via RevokeOwnership. Current owners can be listed via the action ListOwners.
13. Should a device be sold to someone else, it can be reinitialized via the FactorySecurityReset action. It is strongly recommended that a device also include some physical means for achieving the same end, although that means should not necessarily be convenient (e.g., might require opening the case). The physical Reset mechanism is to cover the case when a Security Console takes ownership of the device, has not granted any access or any co-ownership, and then dies irretrievably, leaving the device owned by an SC that can never again be used.
14. It is conceivable that a device manufacturer might want to define some UPnP maintenance actions to which it alone retains authority. To do this under UPnP DeviceSecurity, the manufacturer needs to create a separate device with its own instance of DeviceSecurity and take ownership of that sub-device at time of manufacture. Simply initializing the normal ACL with access permissions granted to the manufacturer's key(s) does not allow the manufacturer to retain control, since a device owner can delete any entries in the normal ACL. This separate sub-device would not have its ACL affected by FactorySecurityReset.

Enhancements

15. A device may, if it chooses, define named sets of permissions, called Profiles, and a Security Console may read those definitions via `GetDefinedProfiles`. These Profiles could be role names, for example, like Parent, Child, or Administrator. The Profile names can be used in the user interface provided by the Security Console.
16. If a CP has certificates to present to the device, in order to gain access, and if the device shows via `GetACLSizes` that it has certificate cache memory available, the CP may send those certificates to the device via the `CacheCertificate` action. Such certificates would then be available on the device over multiple subsequent actions. Implementation of certificate caches is up to the device, but it would make sense to validate certificates at the time they are cached. It might also make sense to derive implied ACL entries from validated certificates and current ACL entries, and store those derived entries at the time a certificate is cached.

1.1. Acknowledgements

The authors and the chair of the UPnP Security Working Committee would like to acknowledge significant contributions made by other UPnP members to help complete this work. Markus Wischy of Siemens was one of the initiators of this effort, and he made many important contributions along the way, including work on securing device discovery and leading one of the sample implementation development teams. Andrew Fiddian-Green of Siemens provided detailed technical feedback on the specification and single-handedly developed one of the sample implementations. Sony Electronics, LG Electronics, GlobespanVirata, and Atinav contributed sample implementations and provided feedback on the specification. Microsoft contributed to the specification and also enhanced the certification test tool to enable testing of secure devices.

2. Service Modeling Definitions

2.1. Service Type

The following service type identifies a service that is compliant with this template:

`urn:schemas-upnp-org:service:DeviceSecurity:1`

The shorthand DeviceSecurity:1 is used herein to refer to this service type.

2.2. Namespaces

The XML in this document should be read as if the following namespace definitions were in effect.

`xmlns="urn:schemas-upnp-org:service:DeviceSecurity:1"`

`xmlns:us="urn:schemas-upnp-org:service:DeviceSecurity:1"`

`xmlns:ds="http://www.w3.org/2000/09/xmldsig#"`

2.3. Referenced Specifications

Unless explicitly stated otherwise herein, implementation of the mandatory provisions of any standard referenced by this specification shall be mandatory for compliance with this specification.

This specification references the UPnP Device Architecture version 1.0 and XML-Signature.

- http://www.upnp.org/download/UPnPDA10_20000613.htm
- <http://www.w3.org/Signature/>

2.4. MustUnderstand

If `mustUnderstand="1"` is present on the `<us:SecurityInfo>` header, the header must be processed in accordance with this specification or an error must be generated. This attribute is not expected to be used, except possibly in some future implementations in which `<us:SecurityInfo>` is directed at an intermediate agent.

2.5. State Variables

DeviceSecurity:1 defines state variables given in the table below. In addition to the state represented by these variables, there is other persistent state (such as ACL, ACLVersion and owner list). Note that all of the security-relevant data for a device (ACL, ACLVersion, LifetimeSequenceBase, owner list and maybe the certificate cache) are expected to be in non-volatile memory so that they would survive power failures. Success of modifications to those data should not be reported until after successful update of that non-volatile memory. ACLVersion can be computed from the ACL itself (e.g., as the hash of the ACL) at the discretion of the manufacturer and in those cases need not be stored in non-volatile memory.

Note: The QueryStateVariable feature is not intended to be supported in the DeviceSecurity service.

Table 1: State variable

Variable Name	Req. or Opt. ¹	data type	Allowed Value	Default Value
NumberOfOwners	R	i4	≥ 0	0
LifetimeSequenceBase	R	string		
TimeHint	O	string		
TotalACLSize	R	i4	≥ 0	
FreeACLSize	R	i4	≥ 0	
TotalOwnerListSize	R	i4	> 0	
FreeOwnerListSize	R	i4	≥ 0	
TotalCertCacheSize	R	i4	≥ 0	
FreeCertCacheSize	R	i4	≥ 0	
A_ARG_TYPE_string	R	string		
A_ARG_TYPE_base64	R	bin.base64		
A_ARG_TYPE_int	R	i4		
A_ARG_TYPE_boolean	R	boolean		

¹ R = Required, O = Optional, X = Non-standard.

2.5.1. NumberOfOwners

A device must maintain a variable which is the count of the number of currently registered owners.

2.5.2. LifetimeSequenceBase

This variable is a non-repeating value that is used to prevent replay for TakeOwnership and SetSessionKeys. Note that if the manufacturer desires to use a sequence number for the LifetimeSequenceBase, this number would have to be held in non-volatile memory where write operations can be expensive and can also be limited in total number performed over the lifetime of the part. Since the value need only be non-repeating, one can for example increment the non-volatile value by some quantity, N, and then keep the old LifetimeSequenceBase value in volatile memory, to be incremented by 1 until it reaches the non-volatile value.

There is a minor security advantage to having LifetimeSequenceBase be unpredictable. A manufacturer is free to make it so, for example by hashing some sequence counter with a secret value (e.g., the device's private key) or, if there is enough CPU power, by digitally signing a sequence counter and using that signature (or its hash) as the LifetimeSequenceBase.

2.5.3. TimeHint

This variable is for devices that have no internal source of time but want to be able to handle certificates and ACL entries that have expiration times. A trusted Security Console or Control Point is able to reset this variable on occasion, to give the device a rough sense of the date and time.

The format of a time hint is as given in section 4.2.1, e.g., 2002-01-26T15:32:41Z.

2.5.4. TotalACLSize

This variable should be the sum of the current FreeACLSize and the number of entries already in the ACL. For devices that have chosen not to store an ACL locally, this can be 0. If TotalACLSize is 0, then the device should not offer the ACL editing actions.

2.5.5. FreeACLSize

This variable should be the number of ACL entries of average size (for that device, as a function of the number and size of permissions) that the device has remaining memory to store.

2.5.6. TotalOwnerListSize

This variable is the number of owner slots in the device's owner list. It must be at least 1. Every security aware device must be able to accept one owner. Multiple owners are optional but the space to record 3 or more owners is recommended, for convenience and fault tolerance.

2.5.7. FreeOwnerListSize

This variable is the number of owner slots in the device's owner list that are still available to be filled.

2.5.8. TotalCertCacheSize

This variable should be the number of certificates currently cached in the device plus the value of FreeCertCacheSize.

2.5.9. FreeCertCacheSize

This variable should be the estimated number of certificates of average size that would fit in the memory remaining for certificate cache on the device. A device that has no memory for certificate cache would set this and TotalCertCacheSize to 0.

2.5.10. A_ARG_TYPE_string

This is an arbitrary string parameter, not related to any real state variable.

2.5.11. A_ARG_TYPE_base64

This is an arbitrary string parameter, not related to any real state variable, carrying the BASE64 representation of a binary byte string or, in some cases, an XML document (so that it doesn't need to be escaped on a character by character basis). Refer to RFC1521 for a definition of BASE64. As defined in that spec (primarily for e-mail transmissions) BASE64 encodings are assumed to be limited in length, with a newline inserted as needed to make sure no line is over 76 characters in length. However, UPnP is not attempting to transmit data by e-mail and need not break lines to that length. In particular, one can generate BASE64 with no white space. One can use BASE64 with no line breaks in normal UPnP messages and must use BASE64 without added white space when building canonical public key encodings (see sections 4.7 and 4.9).

2.5.12. A_ARG_TYPE_int

This is an arbitrary 32-bit integer parameter, not related to any real state variable.

2.5.13. A_ARG_TYPE_boolean

This is an arbitrary Boolean parameter, not related to any real state variable.

2.6. Eventing and Moderation

Table 2: Event Moderation

Variable Name	Evented	Moderated Event	Max Event Rate ¹	Logical Combination	Min Delta per Event ²
NumberOfOwners	<i>Yes</i>	<i>No</i>			<i>1</i>
LifetimeSequenceBase	<i>Yes</i>	<i>No</i>			
TimeHint	<i>No</i>	<i>N/A</i>			
TotalACLSize	<i>No</i>	<i>N/A</i>			
FreeACLSize	<i>Yes</i>	<i>No</i>			
TotalOwnerListSize	<i>No</i>	<i>N/A</i>			
FreeOwnerListSize	<i>Yes</i>	<i>No</i>			<i>1</i>
TotalCertCacheSize	<i>No</i>	<i>N/A</i>			
FreeCertCacheSize	<i>Yes</i>	<i>No</i>			
<i>Non-standard state variables implemented by an UPnP vendor go here.</i>	<i>TBD</i>	<i>TBD</i>	<i>TBD</i>	<i>TBD</i>	<i>TBD</i>

¹ Determined by N, where Rate = (Event)/(N secs).

² (N) * (allowedValueRange Step).

2.7. Actions

The DeviceSecurity service is to be added to a UPnP device when its actions need security. The actions of DeviceSecurity fall into two categories: those used only to maintain data defined and used by DeviceSecurity itself and those that can be used to interact with the device to which DeviceSecurity has been added. Similarly, Control Points that access a secured Device are characterized in two categories: **Security Consoles (SC)**, for those whose only purpose is to manipulate the data that is private to the DeviceSecurity service and **Control Points (CP)**, for those whose purpose is to interact with the device being secured. In the tables below, actions are grouped accordingly.

Table 3: Actions invoked by both Control Point and Security Console

Name	Req. or Opt. ¹	Authoriz.
GetPublicKeys	R	N
GetAlgorithmsAndProtocols	R	N
GetACLSizes	R	O
CacheCertificate	O	O
SetTimeHint	O	R
GetLifetimeSequenceBase	R	N
SetSessionKeys	R	O (1)
ExpireSessionKeys	R	R
DecryptAndExecute	R	O (2)
<i>Non-standard actions implemented by an UPnP vendor go here.</i>	X	O

¹ R = Required, O = Optional, X = Non-standard, N = Not to be used.

Table 4: Actions invoked by a Security Console only

Name	Req. or Opt. ¹	Authoriz.
TakeOwnership	R	(1)
GetDefinedPermissions	R	O
GetDefinedProfiles	O	O
ReadACL	O ²	O
WriteACL	O ²	R
AddACLEntry	O ²	R
DeleteACLEntry	O ²	R
ReplaceACLEntry	O ²	R
FactorySecurityReset	R	R
GrantOwnership	O	R
RevokeOwnership	O	R
ListOwners	O	O
<i>Non-standard actions implemented by an UPnP vendor go here.</i>	X	O

¹ R = Required, O = Optional, X = Non-standard.

² The ACL editing Actions may be omitted from a DeviceSecurity instance if that device declares that it has no ACL (see section 2.9.3). If the device does have an ACL, then the ACL editing actions are required.

The Authoriz column indicates whether a given action may be, must be or must not be authorized.

Note (1): TakeOwnership and SetSessionKeys are special cases in that they must be signed but do not require authorization via ACL entry. TakeOwnership cannot be authorized because it works only when a device is unowned and an unowned device cannot have any ACL entries granting authorization. SetSessionKeys might be access controlled, at the vendor's discretion, if one is worried about the creation of too many session keys (perhaps as part of a denial-of-service attack), but if it is not itself authorized, no attacker can use that to get access to a secured action.

Note (2): DecryptAndExecute does not require its own authorization and must not be signed with the same session key used inside the encrypted SOAP body because that would introduce a conflict in freshness values. The encrypted SOAP body it carries should be signed and authorized, if the action in that body requires it. DecryptAndExecute used for tunneling might need to be signed, but in that case the signature keys of the outer and inner message would be different.

2.8. Cryptographic Notation for Selected Actions

To help explain the cryptographic aspects of some actions, a compact notation is used. There are typically three parties involved, a device, a control point, and a security console. The letters D, C and S are used to indicate these, respectively. The public key of a party is indicated by P, the private key by K. Thus P_D is the public key of the device, while K_D is its private key. Session encryption keys are represented by K subscripted with both parties; thus K_{DC} would be a session key used from the device to the control point, and K_{CD} the session key used from the control point to the device. Similarly, session signing keys are represented by S with the appropriate subscripts.

Encryption is represented by $[\]$, while signing is represented by $\{ \}$. These operations are prefixed by the key used. So, for example, $P_D[K_C\{m\}]$ represents a message m, signed with the private key of control point C, and encrypted with the public key of device D. Concatenation is represented by $|$. In some cases, encryption requires an Initialization Vector (IV) in which case the key used will be subscripted with the IV as well, to indicate encryption under both the key and IV.

Finally, the transmission of a message m from A to B is written as $A \rightarrow B:m$

2.9. Actions Invoked by Both CP and SC

2.9.1. GetPublicKeys

GetPublicKeys retrieves the set of public keys corresponding to private keys held by the device. This spec has defined a use only for a confidentiality key for the device, so one should expect to receive only one key in response.

2.9.1.1. Arguments

Argument(s)	Direction	relatedStateVariable
KeyArg	OUT ^R	A_ARG_TYPE_string

^R = RetVal

The KeyArg argument is an escaped XML string (see section 3.2) containing key definition elements. Within this string, the Modulus and Exponent values are BASE64 encoded.

```
<Keys>
  <Confidentiality>
    <RSAKeyValue>
      <Modulus>xA7SEU+e0y...</Modulus>
      <Exponent>AQAB</Exponent>
    </RSAKeyValue>
  </Confidentiality>
  <Signing>
    <RSAKeyValue>
      <Modulus>xA7SEU+e0y...</Modulus>
      <Exponent>AQAB</Exponent>
    </RSAKeyValue>
  </Signing>
</Keys>
```

In this version of the Security specification, devices have only confidentiality keys. The structure of a <Keys> block is designed to permit devices to have signing keys as well, should that need arise in future versions of this service. If there is no signing key, then the <Signing> element is omitted.

The device's confidentiality key is used for the TakeOwnership and SetSessionKeys actions.

Note: If a device is also a control point, that control point has a signing key, but the signing key is announced to a Security Console via a different mechanism (PresentKey) and would not be presented in this <Keys> element. (See SecurityConsole:1 for details.)

2.9.1.2. Effect on State

None.

2.9.1.3. Errors

errorCode	errorDescription	Description
501	Action Failed	See UPnP Device Architecture section on Control.
800-899	TBD	(Specified by UPnP vendor.)

2.9.2. GetAlgorithmsAndProtocols

GetAlgorithmsAndProtocols retrieves a description of the algorithms and protocols supported by the device.

2.9.2.1. Arguments

Argument(s)	Direction	relatedStateVariable
Supported	OUT ^R	A_ARG_TYPE_string

^R = RetVal

The Supported parameter is an escaped XML string (see section 3.2) giving algorithms and protocols supported by this device, in the format below. The list of all currently defined protocol and algorithm IDs is given below.

```

<Supported>
  <Protocols>
    <p>UPnP</p>
    <p>TLS</p>
    <p>IPSEC</p>
  </Protocols>
  <HashAlgorithms>
    <p>SHA1</p>
  </HashAlgorithms>
  <EncryptionAlgorithms>
    <p>NULL</p>
    <p>RSA</p>
    <p>AES-128-CBC1</p>
  </EncryptionAlgorithms>
  <SigningAlgorithms>
    <p>NULL</p>
    <p>RSA</p>
    <p>SHA1-HMAC</p>
  </SigningAlgorithms>
</Supported>

```

Note 1: The “RSA” ID refers to (RSA, PKCS#1V1.5) for encryption and (RSA, PKCS#1, SHA1) for signing. Should another algorithm tuple be defined for encryption or signing, that tuple would need a different algorithm ID.

Note 2: A NULL entry indicates that although the encryption or signing is *supported*, it is not *required*. If there is no NULL entry, then that feature is required. If there is no algorithm listed, then the feature is not supported.

2.9.2.2. Effect on State

None.

2.9.2.3. Errors

errorCode	errorDescription	Description
501	Action Failed	See UPnP Device Architecture section on Control.
800-899	TBD	(Specified by UPnP vendor.)

2.9.3. GetACLSizes

This action returns the current and total allocation sizes for ACLs and related storage.

A device can have three different blocks of memory holding ACL and related data:

1. the device ACL
2. the owner list (effectively an ACL for both the ACL and the owner list itself)
3. a certificate cache (for holding certificates that extend the base ACL via delegation)

This action returns the values of those memory allocations, both total and current, in units of number of entries. The FreeACLSize and FreeCertCacheSize values are estimates only, since ACL entries and certificates can be of variable length, although in many devices they will be of fixed length. An owner list entry is fixed length and should have definite list sizes.

¹ CBC: Cipher Block Chaining – see the Glossary.

2.9.3.1. Arguments

Argument(s)	Direction	relatedStateVariable
ArgTotalACLSize	OUT	TotalACLSize
ArgFreeACLSize	OUT	FreeACLSize
ArgTotalOwnerListSize	OUT	TotalOwnerListSize
ArgFreeOwnerListSize	OUT	FreeOwnerListSize
ArgTotalCertCacheSize	OUT	TotalCertCacheSize
ArgFreeCertCacheSize	OUT	FreeCertCacheSize

A device developer is free to define memory structures within the device. There is no reason to reveal those definitions to a caller. Therefore, sizes returned in this call are in units of number of entries. Some methods of storing ACL entries and certs allow for arbitrary permission fields, with arbitrary parameter lists, perhaps stored as linked lists. Such a memory structure is not fixed length and any device choosing such a structure should estimate the average size of an ACL entry or cert (for that device) and report an estimate of the memory remaining for such structures. A CP or SC wishing to deliver an excessively large ACL entry or cert must therefore expect the possibility of an error for lack of memory even if the corresponding free size was non-0.

In the minimum persistent memory configuration, a device will remember one owner and that one owner will issue all permissions by certificate (since there is no ACL). Owners have `<all/>` permission by default and therefore do not need ACL entries for themselves. (see, section 4.5.1).

If a device has the memory for it, it may maintain a certificate cache. A CP or SC that wishes to can then submit one or more certificates to the device via the CacheCertificate action. Such certificates would remain on the device in the cache until that storage was required for other purposes or the certificates expire. An advanced device could pre-compute an effective ACL from its ACL and the cached certificates, and use that effective ACL in subsequent action authorization computations, but that is a performance choice and of no security import.

2.9.3.2. Effect on State

None.

2.9.3.3. Errors

errorCode	errorDescription	Description
501	Action Failed	See UPnP Device Architecture section on Control.
701	Not authorized	The caller was not authorized to perform this action.
711	Signature Failure	A signature failed to verify
712	Signature Missing	This command needs to be signed and authorized
714	Invalid Sequence	The sequence base or number is incorrect
715	Invalid Control URL	The control URL within the signature block does not match the one used to deliver the action.
800-899	TBD	(Specified by UPnP vendor.)

2.9.4. CacheCertificate

An authorization certificate (see section 4.4.1) is a signed ACL entry, by which one SC delegates some or all of the rights it has been granted to another SC or CP. It can be generated by the device owner when the device has too little ACL space to hold all the entries that ACL would need. It can also be generated by a non-owner who has been given some rights and who wants to allow someone else to have some of those rights.

A name certificate is generated by a SC and is used to define a named group of control points (or SCs). Each name certificate declares one key (or other named group) to be a member of the group being defined.

This action is optional. A device may, at the manufacturer's discretion, offer memory for a cache of certificates and report that decision via GetACLSizes. If the device does not offer memory for a certificate cache, this action should not be offered.

Support for certificates is optional. When they are supported, they can be communicated from the CP to the Device within the XML-Signature signature block, or they can be sent ahead, via CacheCertificate. If they are cached, then individual messages need not carry them and therefore can be smaller. Such certificates would remain on the device in the cache until that storage was required for other purposes or the certificates expire.

If CacheCertificate is offered, it is up to the manufacturer's discretion whether it should be access-controlled. One might want to employ access control in order to ward off Denial of Service attacks by using up a device's certificate cache memory. One might want not to employ access control in order to keep from complicating the design of the device's permissions.

2.9.4.1. Arguments

Argument(s)	Direction	relatedStateVariable
Certificates	IN	A_ARG_TYPE_string

The Certificates argument is an XML structure, communicated as a properly escaped string (see, section 3.2) in this parameter. The argument is a sequence of one or more certificates (e.g., a chain of certificates that grants a subject some permission(s)). [A "subject" is the grantee of some rights and can be an individual key or a named group of keys.] Those certificates are defined in section 4.4. The structure is of the form:

```
<Sequence>
  <cert> . . . </cert> <ds:Signature> . . . </ds:Signature>
  <cert> . . . </cert> <ds:Signature> . . . </ds:Signature>
  .
  .
  .
</Sequence>
```

It is expected that a device will verify the signature on the offered certificates and cache only the certificate bodies of verified certificates, rejecting certificates that do not verify. It is up to the device whether to accept any valid certificates if they were in a <Sequence> with invalid certificates but it is recommended that valid certificates be accepted even if other certificates cached at the same time are invalid. If a certificate is invalid because its time has expired, it is up to the Control Point offering the certificate to get that certificate replaced or renewed. That is not the responsibility of the device.

2.9.4.2. Effect on State

If successful, the provided certificates are added to the device's certificate cache. Advanced devices can also pre-compute an effective ACL given the new certificate cache and the current ACL, and then use that new effective ACL for future authorization decisions when the caller provides no additional certificates with the call.

Caching of certificates via this action is independent of any decision on the part of the device to cache certificates that are presented within an XML-Signature element. Automatic caching of certificates offered in XML-Signature signature elements will not save on message size, because the CP will not know that the device has done this caching, but the device can cache certificate bodies whose signatures have been verified and thereby save signature verification time in future uses of those cached certificates.

2.9.4.3. Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.
602	Not Implemented	Optional action, not implemented
701	Not authorized	The caller was not authorized to perform this action.
711	Signature Failure	A signature failed to verify
712	Signature Missing	This command needs to be signed and authorized
714	Invalid Sequence	The sequence base or number is incorrect
715	Invalid Control URL	The control URL within the signature block does not match the one used to deliver the action.
731	Wrong device	The certificate offered is not for this device.
751	No memory	There is insufficient memory to accept the offered certificate.
800-899	TBD	(Specified by UPnP vendor.)

2.9.5. SetTimeHint

This optional action should be provided by those devices that have no local source of time but will need time for processing ACL entries or certificates that contain validity dates and times. A Security Console (SC) or other control point on discovering such a device can provide the date and time periodically, assuming it has been authorized to do so. This date and time is to be used in validity tests of certificates or ACL entries. The time interval between updates of the TimeHint can be a configuration parameter of a Security Console and not something for the device to be concerned about. The device merely accepts times as offered and assumes that to be the time until the next SetTimeHint. No attempt is expected to be made to actually count time, treating SetTimeHint as a clock setting function.

This action should be permitted only to those SCs and CPs the owner trusts, since otherwise an attacker might submit false times in order to recover expired access.

See section 4.2 for an explanation of time limits in ACL entries.

2.9.5.1. Arguments

Argument(s)	Direction	relatedStateVariable
ArgTimeHint	IN	TimeHint

The offered TimeHint replaces the stored TimeHint, assuming the caller was authorized to make this action call. Times are to be in the format given in section 4.2.1 e.g.: “2001-12-24T18:13:10Z”.

Note that a device containing a hardware calendar clock need not offer SetTimeHint and its associated state variable, TimeHint. If a device does include a calendar clock, how that clock is set is up to the device manufacturer who would probably not use SetTimeHint for that purpose.

2.9.5.2. Effect on State

The TimeHint variable is updated by the value provided. There is no requirement that changes to TimeHint be monotonic increasing.

2.9.5.3. Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.
602	Not Implemented	Optional action, not implemented
701	Not Authorized	Caller is not authorized to perform this action
711	Signature Failure	The signature failed to verify
712	Signature Missing	This command needs to be signed and authorized.
714	Invalid Sequence	The sequence base or number is incorrect
715	Invalid Control URL	The control URL within the signature block does not match the one used to deliver the action.
781	No Such Session	The signature key ID does not correspond to a valid session.
800-899	TBD	(Specified by UPnP vendor.)

2.9.6. GetLifetimeSequenceBase

This action retrieves the current value of the LifetimeSequenceBase state variable.

2.9.6.1. Arguments

Argument(s)	Direction	relatedStateVariable
ArgLifetimeSequenceBase	OUT ^R	LifetimeSequenceBase

^R = RetVal

The LifetimeSequenceBase variable holds a persistent, non-repeating string value that is changed (e.g., incremented) with each use. It is used in a public-key <Freshness> block and in computing the main parameter of TakeOwnership, in order to prevent replay.

This action must not be access controlled. It is used by a Security Console prior to a TakeOwnership action. Prior to TakeOwnership, a device can not have an ACL entry granting the permission to access this action.

The LifetimeSequenceBase must also not be reset by FactorySecurityReset. It is to be unique over the full lifetime of the device. However, if the device is capable of generating a new public-key pair, then the LifetimeSequenceBase can be reset on generation of a new key pair, since a device is known (under UPnP Security) only by its public key. With a new public-key, one has in effect a new device.

See sections 2.5.2 and 2.11 for further discussion of LifetimeSequenceBase.

2.9.6.2. Effect on State

There is no effect on state.

2.9.6.3. Errors

errorCode	errorDescription	Description
501	Action Failed	See UPnP Device Architecture section on Control.
800-899	TBD	(Specified by UPnP vendor.)

2.9.7. SetSessionKeys

SetSessionKeys allows a Control Point to establish symmetric keys for subsequent signed or encrypted operations. These will typically be:

1. signed SOAP action calls,
2. encrypted SOAP envelopes made via DecryptAndExecute,
3. and optionally other protocols (e.g., IPsec, TLS or whatever else the device manufacturer wants to provision)

A session will remain valid until either the CP or the Device discards it. The CP discards it via ExpireSessionKeys. The Device discards it silently, but the fact of that discard is apparent to the CP the next time it tries to use that session and gets an error return that there is no such session. Typically, a device will discard sessions only when they are inactive for a long time or when the Device is out of session description memory.

2.9.7.1. Arguments

Argument(s)	Direction	relatedStateVariable
EncipheredBulkKey	IN	A_ARG_TYPE_base64
BulkAlgorithm	IN	A_ARG_TYPE_string
Ciphertext	IN	A_ARG_TYPE_base64
CPKeyID	IN	A_ARG_TYPE_int
DeviceKeyID	OUT ^R	A_ARG_TYPE_int
SequenceBase	OUT	A_ARG_TYPE_string

^R = RetVal

Using the cryptographic notation, one can write this as:

$$C \rightarrow D: K_C\{\text{SetSessionKeys}(P_D[K_{Bulk}, IV_{Bulk}], \text{Algorithm}_{Bulk}, K_{Bulk, IVbulk}[Keys], CPKeyID)\}$$

$$D \rightarrow C: S_{DC}\{\text{SetSessionKeysResponse}(DeviceKeyID, SequenceBase)\}$$

where *Keys* is the XML key structure described below.

The EncipheredBulkKey ($P_D[K_{Bulk}, IV_{Bulk}]$) is described in section 4.6, below. It carries a symmetric session key and IV for the Ciphertext.

The BulkAlgorithm (Algorithm_{Bulk}) argument is the name of the symmetric algorithm used to encipher the Ciphertext. The name specifies algorithm, key length and encryption mode. For example, AES-128-CBC is a recommended algorithm, key length and mode.

The Ciphertext ($K_{Bulk, IVbulk}[Keys]$) is bulk encrypted with the symmetric algorithm, symmetric key and IV given in the first two parameters. (See 4.8 and 4.6.1) It is encoded as a BASE64 string. Its plaintext (*SessionKeys*) is an XML structure as follows (for example):

```
<SessionKeys>
  <Confidentiality>
    <Algorithm>AES-128-CBC</Algorithm>
    <KeyToDevice>XXXXXXXXXX</KeyToDevice>
    <KeyFromDevice>YYYYYYYYYY</KeyFromDevice>
  </Confidentiality>
  <Signing>
    <Algorithm>SHA1-HMAC</Algorithm>
    <KeyToDevice>XXXXXXXXXX</KeyToDevice>
    <KeyFromDevice>YYYYYYYYYY</KeyFromDevice>
  </Signing>
</SessionKeys>
```

Key fields in the plaintext block above are in network standard byte order (most significant byte first) and BASE64 encoded.

CPKeyID is an integer type input parameter that contains the index into or search key of a key table on the control point. It is made available to the device for use in event or discovery messages, should those be encrypted by session key, so that the CP can be informed about the key that was used. For the reply to a message, the CP can assume that the same session was used in reply as in the call message.

DeviceKeyID is an integer type output parameter that contains a unique session key identifier for the newly opened session. The Control Point must refer to this key identifier when it terminates the session by making a call to ExpireSessionKeys and when it uses one of these session keys, in XML-Signature blocks or in DecryptAndExecute actions.

SequenceBase is a string type output parameter (possibly an integer encoded as a string) that contains a string unique to that device for this session. That string might be a large random byte string or might be monotonic index counting up from the birth of the device. Its only requirement is that it be unique (at least statistically) over the lifetime of the device and refer to this session. Its purpose is to help prevent replay attacks within sessions. See section 5.2 for a full description of the use of the SequenceBase.

The action call must be signed by the private signing key of the Control Point (i.e. verified by its public signing key). Most actions will be signed and verified by symmetric keys, established with a SetSessionKeys call, but this one must use public key verification.

The LifetimeSequenceBase is a non-repeating value, maintained by the device. It is used in this action within the <Freshness> block of the public key signature to prevent replay attacks.

A successful reply to this action, if signed, should be signed by the session key established in this call (<Signing><KeyFromDevice>). An error reply to this action can be sent unsigned.

It is up to the device manufacturer whether this action requires authorization. A session key is not granted any power *per se*. It is associated with the public key that verified the SetSessionKeys action call. If that public key is authorized, then the associated session key is authorized. However, the session keys (and SequenceBase) must be kept in memory and that memory might be a scarce resource. In that case, the device may choose to exercise access control over the SetSessionKeys action to prevent a resource exhaustion attack.

A manufacturer is free to restrict the allocation of sessions and to define error codes to reflect those restrictions. For example, it is possible that a manufacturer could permit only one active session per control point public key. In that case, it is recommended that an otherwise successful request for a new session override any previous session (in case the control point had reinitialized and forgotten both the existence and the values of the previous session). It is also possible for a device to discard a session at any time, forcing establishment of a new session, although excessively rapid dropping of sessions would create a great deal of needless overhead.

2.9.7.2. Effect on State

SetSessionKeys defines new session keys and key IDs maintained by the device until ExpireSessionKeys is called.

For this session, the SequenceBase (generated during the call) is remembered and the sequence numbers for messages in this session are set to 0. Subsequent messages using this session must have the same SequenceBase and monotonically increasing (but not necessarily contiguous) sequence numbers in each direction in order to be accepted as fresh. That is, the CP should keep two sequence number counters, one that it uses and one that the device uses. A device may keep one or two sequence counters, since each message has precisely one reply and the device could use the incoming sequence counter value as the reply counter value.

After a successful SetSessionKeys, the LifetimeSequenceBase is incremented (see 2.5.2) so that every new session is created by a fresh command.

2.9.7.3. Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.
701	Action Not Authorized	Caller is not authorized to perform this action. This probably means that the caller has no access rights at all on this device.
711	Signature Failure	The signature failed to verify
712	Signature Missing	This action must be signed by an asymmetric key.
714	Invalid Sequence	LifetimeSequenceBase is incorrect, probable replay attack
715	Invalid Control URL	The control URL in the signature block did not match the Control URL used to deliver this action.
721	Algorithm Not Supported	A requested algorithm is not supported.
722	No IPSEC	This device is not capable of IPSEC.
800-899	TBD	(Specified by UPnP vendor.)

2.9.8. ExpireSessionKeys

ExpireSessionKeys permits a control point to inform a device explicitly that it no longer intends to use a session. The device should then destroy the keying information and free any resources associated with it.

2.9.8.1. Arguments

Argument(s)	Direction	relatedStateVariable
DeviceKeyID	IN	A_ARG_TYPE_int

DeviceKeyID is the device's session key identifier returned by SetSessionKeys. After this call, the device will have forgotten all of the indicated session keys and those keys are therefore unusable.

This call can be honored only from the owner of that session, in order to prevent denial of service annoyances.

2.9.8.2. Effect on State

If the call was successfully authorized, ExpireSessionKeys removes the corresponding session from the table of session key information. Note: if the session being expired was used in this call, the reply keys for that session should be deleted only after encrypting and/or signing the reply.

With the removal of that session, if there are other things (encrypted events, IPSEC security associations, etc.) that rely on keys that had been defined in the session just removed, then those other things will probably need to be removed or stopped as well.

2.9.8.3. Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.
701	Action Not Authorized	Caller is not authorized to perform this action.
711	Signature Failure	The signature failed to verify
712	Signature Missing	This action must be signed
714	Invalid Sequence	SessionSequence or SequenceNumber is incorrect, probable replay attack
715	Invalid Control URL	The control URL in the signature block did not match the Control URL used to deliver this action.
781	No Such Session	The signature key ID does not correspond to a valid session.
800-899	TBD	(Specified by UPnP vendor.)

2.9.9. DecryptAndExecute

DecryptAndExecute provides secrecy of a SOAP message by tunneling it inside an action argument. Specifically, that message is held in the Request argument and any reply from that message is returned in the Reply argument. The Request and Reply arguments are encrypted. The payload message is probably signed, but the signing of that message is independent of the operation of DecryptAndExecute itself. The DecryptAndExecute message may also be signed, in special circumstances (e.g., when a message needs to be signed to be authorized to get through a SOAP intermediary or a gateway device). In this latter case, the key that signs the DecryptAndExecute message will doubtless be a different key from the one that signs the payload, since they are being processed by different devices.

There are three input arguments to DecryptAndExecute: the key ID of the key being used, an IV value for the symmetric encryption and a Ciphertext block containing a SOAP message to be executed. The output arguments are an output IV value and a Ciphertext block holding the reply from that SOAP message. It is assumed that the caller has done SetSessionKeys before doing DecryptAndExecute, in order to get the DeviceKeyID for the session keys being used.

2.9.9.1. Arguments

Argument(s)	Direction	relatedStateVariable
DeviceKeyID	IN	A_ARG_TYPE_int
Request	IN	A_ARG_TYPE_base64
InIV	IN	A_ARG_TYPE_base64
Reply	OUT ^R	A_ARG_TYPE_base64
OutIV	OUT	A_ARG_TYPE_base64

^R = RetVal

Using the cryptographic notation, this action and response can be written:

C→D: DecryptAndExecute(DeviceKeyID, $K_{CD,InIV}[S_{CD}\{HTTP-SOAP-Request\}], InIV)$

D→C: DecryptAndExecuteResponse($K_{DC,OutIV}[S_{DC}\{HTTP-SOAP-Response\}], OutIV)$

The Request and Reply arguments contain Ciphertext. The former is encrypted using the indicated “to device” session key, and the latter using the corresponding “from device” key. Both ciphertexts are encoded as BASE64 strings. Plaintexts are to be padded as described in section 4.8.

The plaintext within Request or Reply is a full SOAP HTTP message. The plaintext is to be NULL-terminated, for the convenience of HTTP processors written in C. It is to be passed back into the HTTP processor, XML parser and SOAP engine, recursively.