RATAConfig:1 Service

For UPnP™ Version 1.0
Status: Standardized DCP
Date: September 30, 2009
Document Version: 1.0
Service Template Version: 2.00

Authors	Company
Bich Nguyen (Co-chair)	Cisco
Sridhar Ramaswamy	Cisco
Ayodele Damola	Ericsson
Bryan Roe	Intel
Gunner Danneels	Intel
Alexander Kokhanyuk	Motorola
Vlad Stirbu	Nokia
Cathy Chan	Nokia
Jan Brands	NXP
Jeffrey Kang	Philips
Wouter van der Beek	Philips
Shrikant Kanaparti	Samsung

Authors	Company
Se-Hee Han	Samsung
Mahfuzur Rahman (Co-chair)	Samsung
Sanjeev Verma	Samsung

Contents

List of Tables

List of Figures

1 Overview and Scope

This service definition is compliant with the UPnP Device Architecture version 1.0. It defines a service type referred to herein as *RATAConfig* service.

1.1 Introduction

The *RATAConfig* service is a UPnP service that allows control points to provision and configure the parameters that are required for enabling a Remote Access Server to accept and a Remote Access Client to initiate remote access connections. This service provides control points with the following functionality:

- Determine the Remote Access Transport Agents that can be configured by the service.
- Determine the delivery mechanisms for credentials supported by the service.
- Configure Remote Access Transport Agent profiles
- Management of Remote Access Transport Agent profiles

This service does not address:

- The trust model that will enable secure remote access connections.
- The delivery of credentials.

1.2 Notation

- In this document, features are described as Required, Recommended, or Optional as follows:

  The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” in this specification are to be interpreted as described in [RFC 2119].

  In addition, the following keywords are used in this specification:

  PROHIBITED – The definition or behavior is an absolute prohibition of this specification. Opposite of REQUIRED.

  CONDITIONALLY REQUIRED – The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is REQUIRED, otherwise it is PROHIBITED.

  CONDITIONALLY OPTIONAL – The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is OPTIONAL, otherwise it is PROHIBITED.

  These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

- Strings that are to be taken literally are enclosed in “double quotes”.
- Placeholder values that need to be replaced are enclosed in the curly brackets “{” and “}”.
- Words that are emphasized are printed in *italic*.
- Keywords that are defined by the UPnP Working Committee are printed using the *forum* character style.
- Keywords that are defined by the UPnP Device Architecture are printed using the **arch** character style.

- A double colon delimiter, "::", signifies a hierarchical parent-child (parent::child) relationship between the two objects separated by the double colon. This delimiter is used in multiple contexts, for example: Service::Action(), Action()::Argument, parentProperty::childProperty.

1.3 Vendor-defined Extensions

Whenever vendors create additional vendor-defined state variables, actions or properties, their assigned names and XML representation MUST follow the naming conventions and XML rules as specified in [DEVICE], Section 2.5, "Description: Non-standard vendor extensions".

1.4 References

1.4.1 Normative References

This section lists the normative references used in this specification and includes the tag inside square brackets that is used for each such reference:

[DEVICE] – UPnP Device Architecture, version 1.0.
Available at: http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0-20080424.pdf.
Latest version available at: http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0.pdf.

[DEVICE-IPv6] – UPnP Device Architecture, version 1.0., Annex A – IP Version 6 Support.
Available at: http://www.upnp.org/resources/documents/AnnexA-IPv6_000.pdf

[RAClient] – RAClient:1, UPnP Forum,
Available at: http://www.upnp.org/specs/ra/UPnP-ra-RAClient-v1-Device-20090930.pdf.
Latest version available at: http://www.upnp.org/specs/ra/UPnP-ra-RAClient-v1-Device.pdf.

[RAServer] – RAServer:1, UPnP Forum,
Available at: http://www.upnp.org/specs/ra/UPnP-ra-RAServer-v1-Device-20090930.pdf.
Latest version available at: http://upnp.org/specs/ra/UPnP-ra-RAServer-v1-Device.pdf.

[RADASync] – RADASync:1, UPnP Forum,
Available at: http://www.upnp.org/specs/ra/UPnP-ra-RADASync-v1-Service-20090930.pdf.
Latest version available at: http://www.upnp.org/specs/ra/UPnP-ra-RADASync-v1-Service.pdf.

[RFC 2119] – IETF RFC 2119, Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, March 1997.
Available at: http://www.ietf.org/rfcs/rfc2119.txt.

[DADS-XSD] – XML Schema for UPnP RA Discovery Agent XML Data Structures
Available at: http://www.upnp.org/schemas/ra/dads-v1-20090930.xsd.
Latest version available at: http://www.upnp.org/schemas/ra/dads-v1.xsd.

[TADS-XSD] – XML Schema for UPnP RA Transport Agent XML Data Structures
Available at: http://www.upnp.org/schemas/ra/tads-v1-20090930.xsd.
Latest version available at: http://www.upnp.org/schemas/ra/tads-v1.xsd.

[IPSEC-XSD] – XML Schema for IPSec Transport Agent Options and Configuration XML Data Structures
Available at: http://www.upnp.org/schemas/ra/tacfg-ipsec-v1-20090930.xsd.
Latest version available at: http://www.upnp.org/schemas/ra/tacfg-ipsec-v1.xsd.

[OPENVPN-XSD] – XML Schema for OpenVPN Transport Agent Options and Configuration XML Data Structures
Available at: http://www.upnp.org/schemas/ra/tacfg-openvpn-v1-20090930.xsd.
Latest version available at: http://www.upnp.org/schemas/ra/tacfg-openvpn-v1.xsd.

[XML] – "Extensible Markup Language (XML) 1.0 (Third Edition)", François Yergeau, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, eds., W3C Recommendation, February 4, 2004.
Available at: http://www.w3.org/TR/2004/REC-xml-20040204/.

1.4.2 Informative References

This section lists the informative references that are provided as information in helping understand this specification:

[IGD] – InternetGatewayDevice:1, UPnP Forum, November, 2001
Available at: http://www.upnp.org/standardizeddcps/documents/UPnP_IGD_1.0.zip

[RAARCH] – RAArchitecture:1, UPnP Forum,
Available at: http://www.upnp.org/specs/ra/UPnP-ra-RAArchitecture-v1-20090930.pdf.
Latest version available at: http://www.upnp.org/specs/ra/UPnP-ra-RAArchitecture-v1.pdf.

[RADAConfig] – RADAConfig:1, UPnP Forum,
Available at: http://www.upnp.org/specs/ra/UPnP-ra-RADAConfig-v1-Service-20090930.pdf.
Latest version available at: http://www.upnp.org/specs/ra/UPnP-ra-RADAConfig-v1-Service.pdf.

[RFC 2406] – IETF RFC 2406, IP Encapsulating Security Payload (ESP), S. Kent, R. Atkinson, November 1998
Available at: http://www.ietf.org/rfc/rfc2406.txt

[RFC 3706] – IETF RFC 3706, A Traffic-Based Method of Detecting Dead Internet Key Exchange (IKE) Peers, G. Huang, et. Al., February 2004
Available at: http://www.ietf.org/rfc/rfc3706.txt

[RFC 3947] – IETF RFC 3947, Negotiation of NAT-Traversal in the IKE, T. Kivinen, B. Swander, A. Huttunen, V. Volpe, January 2005.
Available at: http://www.ietf.org/rfc/rfc3947.txt.

[RFC 4306] – IETF RFC 4306, Internet Key Exchange (IKEv2) Protocol, C. Kaufman, Ed., December 2005
Available at: http://www.ietf.org/rfc/rfc4306.txt

2 Service Modeling Definitions

2.1 Service Type

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

urn:schemas-upnp-org:service:*RATAConfig:1*

RATAConfig service is used herein to refer to this service type.

2.2 Terms and Abbreviations

2.2.1 Abbreviations

Table 2-1: Abbreviations

Definition	Description
DPD	Dead Peer Detection
ESP	Encapsulating Security Payload
IKE	Internet Key Exchange
IPsec	IP security
RAC	Remote Access Client
RADA	Remote Access Discovery Agent
RAS	Remote Access Server
RAT	Remote Access Transport
RATA	Remote Access Transport Agent

2.2.2 Terms

2.2.2.1 Credentials

The term credentials refer to certificates, shared secrets or other means of authentication used in the RATA context.

2.2.2.2 Local Device

A local device is a UPnP device that is attached to the physical network where the RADA is located.

2.2.2.3 Management Console

The collection of Control Points used to configure and monitor Remote Access related services.

2.2.2.4 Remote Access Client

The Remote Access Client (RAC) is the peer physical device that is not part of the physical home network. The RAC is exposing only the UPnP devices and services that are embedded in the physical device.

2.2.2.5 Remote Access Network Interface

The RA network interface is the network interface that is created by the Remote Access Transport Agent. The settings for this interface are contained in a RAT profile.

2.2.2.6 Remote Access Server

The Remote Access Server (RAS) is the peer physical device located in the home network. RAS is exposing to the RAC the UPnP devices and services available in the physical home network as well as any embedded in the physical RAS device.

2.2.2.7 Remote Access Transport Agent Profile

A RATA profile is a configured RATA connection ready to be used by either accepting connections on the RAS side or to initiate connections on the RAC side.

2.2.2.8 Remote Device

A remote device is a UPnP device that is not attached to the physical network where the RADA is located.

2.3 *RATAConfig* Service Architecture

This service is responsible with providing a configuration interface for a secure communication channel that enables a remote UPnP device to interact with the UPnP devices located in the home network.

2.4 State Variables

Reader Note: For a first-time reader, it may be more helpful to read the action definitions before reading the state variable definitions.

2.4.1 State Variable Overview

Table 2-2: State Variables

Variable Name	R/O[1]	Data Type	Allowed Values	Eng. Units
SystemInfo	*R*	**string**	See Section 2.4.2	
TransportAgentCapabilities	*R*	**string**	See Section 2.4.3	
CredentialDelivery	*R*	**string**	See Section 2.4.4	
CredentialsList	*R*	**string**	See Section 2.4.5	
ProfileList	*R*	**string**	See Section 2.4.6	
A_ARG_TYPE_ProfileConfigInfo	*R*	**string**	See Section 2.4.7	
A_ARG_TYPE_ProfileID	*R*	**ui4**	See Section 2.4.8	

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

2.4.2 *SystemInfo*

This state variable contains the snapshot of all networks the RATA has a relationship with, the status of the connection and the identity associated with the remote network.

The structure of the *SystemInfo* argument is a DADS XML Document:

- <systemInfo> is the root element.
- See the DADS schema [DADS-XSD] for more details on the structure. The available properties and their names are described in Appendix A.1 of [RADASync].

Note that since the value of *SystemInfo* is XML, it needs to be escaped (using the normal XML rules: [XML] Section 2.4 Character Data and Markup) before embedding in a SOAP response message.

Note: The *SystemInfo* maintained by the RATAConfig service is also shared by the RADASync and RADAConfig services. This state variable MUST be updated by the device and propagated internally to those other services when new remote networks become available or existing remote networks become unavailable, for triggering the RADA synchronization process. Furthermore, invocations of the *AddProfile()*, *DeleteProfile()* and *EditProfile()* actions also result in a modification of this state variable. Each modification in *SystemInfo* MUST be signalled by the device through the *SystemInfoUpdateID* evented state variable of the RADAConfig service (see Section 2.4.3 of [RADAConfig]).

2.4.3 *TransportAgentCapabilities*

This state variable contains the list of remote access transport agent protocols and their capabilities supported by the RATAConfig.

The structure of the *TransportAgentCapabilities* argument is a TADS XML Document.

- `<transportAgentCapability>` is the root element.
- See the TADS schema [TADS-XSD] for more details on the structure. The available properties and their names are described in Appendix A.3. Examples are provided in Appendix C.2.1.1, C.2.2.1 and C.2.3.1.

Note that since the value of *TransportAgentCapabilities* is XML, it needs to be escaped (using the normal XML rules: [XML] Section 2.4 Character Data and Markup) before embedding in a SOAP response message.

2.4.4 *CredentialDelivery*

This state variable contains the list of credential delivery mechanisms supported by the RATAConfig.

The structure of the *CredentialDelivery* argument is a TADS XML Document.

- `<credentialDelivery>` is the root element.
- See the TADS schema [TADS-XSD] for more details on the structure. The available properties and their names are described in Appendix A.4.

Note that since the value of *CredentialDelivery* is XML, it needs to be escaped (using the normal XML rules: [XML] Section 2.4 Character Data and Markup) before embedding in a SOAP response message.

2.4.5 *CredentialsList*

This state variable contains the list of credentials that are present on the RATA.

The structure of the *CredentialsList* argument is a TADS XML Document:

- `<credentialsList>` is the root element.
- See the TADS schema [TADS-XSD] for more details on the structure. The available properties and their names are described in Appendix A.5.

Note that since the value of *CredentialsList* is XML, it needs to be escaped (using the normal XML rules: [XML] Section 2.4 Character Data and Markup) before embedding in a SOAP response message.

2.4.6 *ProfileList*

This state variable contains the list of configured profiles on the RATA.

The structure of the *ProfileList* argument is a TADS XML Document:

- `<profileList>` is the root element.
- See the TADS schema [TADS-XSD] for more details on the structure. The available properties and their names are described in Appendix A.1.

Note that since the value of *ProfileList* is XML, it needs to be escaped (using the normal XML rules: [XML] Section 2.4 Character Data and Markup) before embedding in a SOAP response message.

2.4.7 *A_ARG_TYPE_ProfileConfigInfo*

This state variable contains the profile configuration information for particular remote access transport protocol supported by RATA.

The structure of the A_ARG_TYPE_ProfileConfigInfo is a TADS XML Document:

- `<profileConfig>` is the root element.
- See the TADS schema [TADS-XSD] for more details on the structure. The available properties and their names are described in Appendix A.2. Examples are provided in Appendix C.2.1.2, C.2.1.3, C.2.2.2, C.2.2.3, C.2.3.2, C.2.3.3, D.2.1 and D.2.2.

Note that since the value of *A_ARG_TYPE_ProfileConfigInfo* is XML, it needs to be escaped (using the normal XML rules: [XML] Section 2.4 Character Data and Markup) before embedding in a SOAP response message.

2.4.8 *A_ARG_TYPE_ProfileID*

This state variable contains the unique id for a profile.

2.5 Eventing and Moderation

Table 2-3: Eventing and Moderation

Variable Name	Evented	Moderated Event	Max Event Rate[1]	Logical Combination	Min Delta per Event[2]
SystemInfo	*NO*	*NO*			
TransportAgentCapabilities	*NO*	*NO*			
CredentialDelivery	*NO*	*NO*			
CredentialsList	*YES*	*NO*			
ProfileList	*NO*	*NO*			
A_ARG_TYPE_ProfileConfigInfo	*NO*	*NO*			
A_ARG_TYPE_ProfileID	*NO*	*NO*			

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

[2] (N) * (allowedValueRange Step).

2.5.1 Relationships Between State Variables

None.

2.6 Actions

Table 2-4: Actions

Name	R/O[1]
GetTransportAgentCapabilities()	*R*
GetSupportedCredentialDelivery()	*R*
GetCredentialsList()	*R*
GetProfileList()	*R*
AddProfile()	*R*
EditProfile()	*R*
DeleteProfile()	*R*
GetProfileConfigInfo()	*R*

[1] *R* = REQUIRED, *O* = OPTIONAL, *X* = Non-standard.

2.6.1 *GetTransportAgentCapabilities()*

This action specifies a mechanism to determine the remote access transport agent protocols and their capabilities supported by the RATA.

2.6.1.1 Arguments

Table 2-5: Arguments for *GetTransportAgentCapabilities()*

Argument	Direction	relatedStateVariable
TransportAgentCapabilities	*OUT*	*TransportAgentCapabilities*

2.6.1.1.1 *TransportAgentCapabilities*

This argument exposes the capabilities of the transport agent.

2.6.1.2 Dependency on State

None.

2.6.1.3 Effect on State

None.

2.6.1.4 Control Point Requirements

None.

2.6.1.5 Errors

Table 2-6: Error Codes for *GetTransportAgentCapabilities()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.

2.6.2 *GetSupportedCredentialDelivery()*

This action specifies a mechanism to determine what are the mechanisms for delivering credentials that are supported by the RATA.

2.6.2.1 Arguments

Table 2-7: Arguments for *GetSupportedCredentialDelivery()*

Argument	Direction	relatedStateVariable
SupportedCredentialDelivery	*OUT*	*CredentialDelivery*

2.6.2.1.1 *SupportedCredentialDelivery*

This argument exposes which credential delivery mechanisms are supported by the device hosting the service.

2.6.2.2 Dependency on State

None.

2.6.2.3 Effect on State

None.

2.6.2.4 Control Point Requirements

None.

2.6.2.5 Errors

Table 2-8: Error Codes for *GetSupportedCredentialDelivery()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.

2.6.3 *GetCredentialsList()*

This action specifies a mechanism to determine the credentials currently available on the RATA.

2.6.3.1 Arguments

Table 2-9: Arguments for *GetCredentialsList()*

Argument	Direction	relatedStateVariable
CurrentCredentialsList	*OUT*	*CredentialsList*

2.6.3.1.1 *CurrentCredentialsList*

This argument contains the list of credentials currently available on the RATA. Each entry in the list contains also a pointer to the respective credential.

2.6.3.2 Dependency on State

None.

2.6.3.3 Effect on State

None.

2.6.3.4 Control Point Requirements

Control points MUST select and remember the CredentialID from the list in order to provide this pointer in the *AddProfile()* action.

2.6.3.5 Errors

Table 2-10: Error Codes for *GetCredentialsList()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.

2.6.4 *GetProfileList()*

This action specifies a mechanism to determine the profiles currently configured on the RATA.

2.6.4.1 Arguments

Table 2-11: Arguments for *GetProfileList()*

Argument	Direction	relatedStateVariable
ProfileList	*OUT*	*ProfileList*

2.6.4.1.1 *ProfileList*

This argument contains a list of configured profiles.

2.6.4.2 Dependency on State

None.

2.6.4.3 Effect on State

None.

2.6.4.4 Control Point Requirements

None.

2.6.4.5 Errors

Table 2-12: Error Codes for *GetProfileList()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.

2.6.5 *AddProfile()*

This action defines a mechanism to configure profile for RATA.

2.6.5.1 Arguments

Table 2-13: Arguments for *AddProfile()*

Argument	Direction	relatedStateVariable
NewProfileConfigInfo	*IN*	*A_ARG_TYPE_ProfileConfigInfo*

2.6.5.1.1 *NewProfileConfigInfo*

This argument contains the protocol config options and associated credentials for the new RATA profile.

2.6.5.2 Dependency on State

None.

2.6.5.3 Effect on State

The effect of this action is that the device must generate a unique ID for the newly created profile and update the *ProfileList* state variable. Furthermore, the device MUST update the *SystemInfo* state variable with the information on this newly created profile. Note: Since the *SystemInfo* state variable is shared with the RADAConfig and RADASync services, the implementation MUST propagate the modification of its value internally to those services, if present on the same device.

2.6.5.4 Control Point Requirements

None.

2.6.5.5 Errors

Table 2-14: Error Codes for *AddProfile()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
701	Invalid Profile Data	The profile data provided is not valid

2.6.6 *EditProfile()*

This action defines a mechanism for updating the options and parameters of an already configured profile.

2.6.6.1 Arguments

Table 2-15: Arguments for *EditProfile()*

Argument	Direction	relatedStateVariable
ProfileID	*IN*	*A_ARG_TYPE_ProfileID*
UpdatedProfileConfigInfo	*IN*	*A_ARG_TYPE_ProfileConfigInfo*

2.6.6.1.1 *ProfileID*

This argument indicates the ID of the profile that is edited.

2.6.6.1.2 *UpdatedProfileConfigInfo*

This argument contains the updated protocol config options and associated credentials for a RAT profile. However, it MUST NOT be used to change type of transport. The transport type can only be changed by creating a new profile.

This argument only contains only the parameter values that need to be changed.

The deletion of specific parameters can be achieved by deleting and recreating an entire profile.

2.6.6.2 Dependency on State

The profile indicated by the *ProfileID* must exist.

2.6.6.3 Effect on State

Updating the profile may potentially result in modifications to the *SystemInfo* state variable (e.g. if the credentialID is modified). Since the *SystemInfo* state variable is shared with the RADAConfig and RADASync services, the implementation MUST propagate the modification of its value internally to those services, if present on the same device.

2.6.6.4 Control Point Requirements

None.

2.6.6.5 Errors

Table 2-16: Error Codes for *EditProfile()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
701	Invalid Profile Data	The profile data provided is not valid
702	Invalid Profile ID	The profile identified by profileID does not exist.

2.6.7 *DeleteProfile()*

This action defines a mechanism to delete profiles from a RATA.

2.6.7.1 Arguments

Table 2-17: Arguments for *DeleteProfile()*

Argument	Direction	relatedStateVariable
ProfileID	*IN*	*A_ARG_TYPE_ProfileID*

2.6.7.1.1 *ProfileID*

This argument indicates the ID of the profile that is deleted.

2.6.7.2 Dependency on State

The profile indicated by the *ProfileID* must exist.

2.6.7.3 Effect on State

The effect is that the *ProfileList* must be updated. Furthermore, the device MUST update the *SystemInfo* state variable. Note: Since the *SystemInfo* state variable is shared with the RADAConfig and RADASync services, the implementation MUST propagate the modification of its value internally to those services, if present on the same device.

2.6.7.4 Control Point Requirements

None.

2.6.7.5 Errors

Table 2-18: Error Codes for *DeleteProfile()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.

ErrorCode	errorDescription	Description
702	Invalid Profile ID	The profile identified by profileID does not exist.

2.6.8 *GetProfileConfigInfo()*

This action a mechanism the determine the options and parameters of an already configured profile.

2.6.8.1 Arguments

Table 2-19: Arguments for *GetProfileConfigInfo()*

Argument	Direction	relatedStateVariable
ProfileID	*IN*	*A_ARG_TYPE_ProfileID*
ProfileConfigInfo	*OUT*	*A_ARG_TYPE_ProfileConfigInfo*

2.6.8.1.1 *ProfileID*

This argument indicates the ID of the profile for which the profile date is wanted.

2.6.8.1.2 *ProfileConfigInfo*

This argument contains the protocol config options and associated credentials for a RAT profile associated with the *ProfileID*.

2.6.8.2 Dependency on State

The profile indicated by the *ProfileID* must exist.

2.6.8.3 Effect on State

None.

2.6.8.4 Control Point Requirements

None.

2.6.8.5 Errors

Table 2-20: Error Codes for *GetProfileConfigInfo()*

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
702	Invalid Profile ID	The profile identified by profileID does not exist.

2.6.9 Error Code Summary

The following table lists error codes common to actions for this service type. If an action results in multiple errors, the most specific error should be returned.

Table 2-21: Error Code Summary

ErrorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
700		Reserved for future extensions.
701	Invalid Profile Data	The profile data provided is not valid
702	Invalid Profile ID	The profile identified by profileID does not exist.

Note: 800-899 Error Codes are not permitted for standard actions. See UPnP Device Architecture section on Control for more details.

2.7 Theory of Operation

2.7.1 The Interaction Model

Remote Access Transport connections can be established only if the *RAServer* [RAServer] has a profile configured for accepting connections and the *RAClient* [RAClient] has matching profile configured to initiate connection to the particular server. A server profile may have several corresponding client profiles.

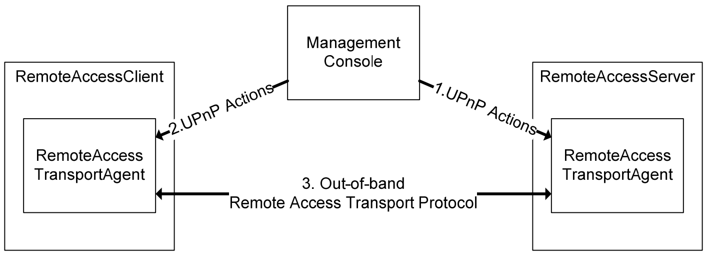
Management Console
RemoteAccessClient
RemoteAccess TransportAgent
2.UPnP Actions
1.UPnP Actions
RemoteAccessServer
RemoteAccess TransportAgent
3. Out-of-band Remote Access Transport Protocol

Figure 2-1: The Interaction Model.

The Management Console may configure both client and server at the same time but it may also do the configuration in two steps: first the server and then the client. This flexibility allows the Management Console to configure a client even if it is not present in the same network/location as the server. In such cases, the Management Console MUST cache the server profile information.

2.7.2 Detecting the RATA Role

The *RATAConfig* service can be embedded in either *RAServer* Device [RAServer] or *RAClient* Device [RAClient]. Based on this information, the Control Point can detect if the RATA operates in server or client mode, so that it can deliver the appropriate configuration options while setting up the RAT profiles.

2.7.3 Configuring Remote Access Transport Profile (Server)

Before starting the configuration of Remote Access Transport profiles, the Control Point MUST determine the operational role played by the RATA, e.g. client or server. The exact procedure is described in Section 2.7.2.

The Management Console will start the configuration procedure by first querying the *RATAConfig* service to detect which are the transport protocols supported (e.g. *GetTransportAgentCapabilities()*) and what are the mechanisms for delivering the credentials (e.g. *GetSupportedCredentialDelivery()*). If following this request, the Management Console and the *RATAConfig* are sharing a common credential delivery, the user is triggering the delivery using the specific procedures of the out-of-band mechanism.

When the delivery process is successfully completed, the Management console will query again the *RATAConfig* service to find out which credentials are available on the device (e.g. *GetCredentialsList()*). From the retrieved list, the Management Console selects the delivered credential and will remember the pointer to it.

Next, the Management Console will select the transport protocol options desired for this particular connection, will include the pointer to the selected credential and will deliver the profile settings to the *RATAConfig* (e.g. *AddProfile()*). At this point the server is ready to accept incoming connections.

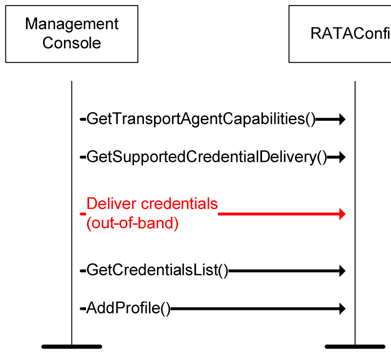
Management Console
RATAConfig
GetTransportAgentCapabilities()
GetSupportedCredentialDelivery()
Deliver credentials (out-of-band)
GetCredentialsList()
AddProfile()

Figure 2-2: Configuring Remote Access Transport Profiles.

2.7.4 Configuring Remote Access Transport Profile (Client)

The configuration procedure of the Remote Access Transport profile on the client follows the same message exchange patterns like for the server (see Section 2.7.3).

An additional step is performed internally by the Management Console, which has to check that the supported protocols reported by the client are matching those of the server. In case of multiple matches the Management Console will select one according to preconfigured policies or will ask the user to select one.

2.7.5 Editing a Profile

Before starting the editing of Remote Access Transport profiles, the Control Point MUST determine the operational role played by the RATA, e.g. client or server. The exact procedure is described in Section 2.7.2.

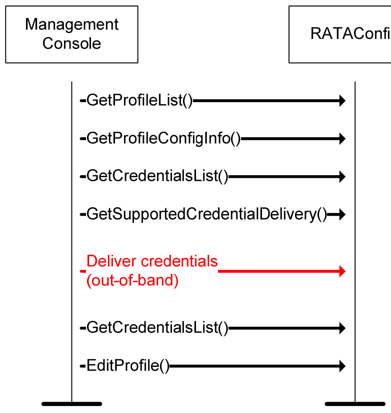

Figure 2-3: Editing Remote Access Transport Profiles.

The Management Console will start the editing process by first querying the *RATAConfig* service to get the list of profiles (e.g. *GetProfileList()*) that are configured on RATA. Once the profile is identified, the Management Console will get the configuration details (e.g. *GetProfileConfigInfo()*).

In case the credentials need to be modified, the Management Console will query the RATAConfig to see which credentials are available on the device (e.g. *GetCredentilasList()*). Optionally the Management Console will deliver additional credentials (e.g. *GetSupportedCredentialDelivery()*, out-of-band credential delivery and *GetCredentialsList()*).

When all changes are decided, the Management Console can deliver the new settings to the device (e.g. *EditProfile()*).

2.7.6 Deleting a Profile

The Management Console will start the deleting process by first querying the *RATAConfig* service to get the list of profiles (e.g. *GetProfileList()*) that are configured on RATA. Once the profile is identified, the Management Console will delete the profile (e.g. *DeleteProfile()*).

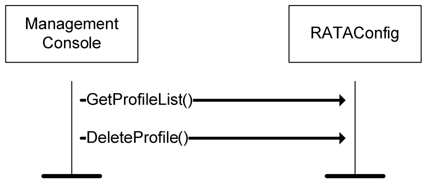

Figure 2-4: Deleting Remote Access Transport Profiles.

3 XML Service Description

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">

   <specVersion>
      <major>1</major>
      <minor>0</minor>
   </specVersion>

   <actionList>

      <action>
         <name>GetTransportAgentCapabilities</name>
         <argumentList>
            <argument>
               <name>TransportAgentCapabilities</name>
               <direction>out</direction>
               <relatedStateVariable>
                  TransportAgentCapabilities
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>GetSupportedCredentialDelivery</name>
         <argumentList>
            <argument>
               <name>SupportedCredentialDelivery</name>
               <direction>out</direction>
               <relatedStateVariable>
                  CredentialDelivery
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>GetCredentialsList</name>
         <argumentList>
            <argument>
               <name>CurrentCredentialsList</name>
               <direction>out</direction>
               <retval/>
               <relatedStateVariable>
                  CredentialsList
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>GetProfileList</name>
         <argumentList>
            <argument>
               <name>ProfileList</name>
               <direction>out</direction>
               <retval/>
               <relatedStateVariable>
                  ProfileList
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>AddProfile</name>
         <argumentList>
            <argument>
               <name>newProfileConfigInfo</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_ProfileConfigInfo
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>EditProfile</name>
         <argumentList>
            <argument>
               <name>ProfileID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_ProfileID
               </relatedStateVariable>
            </argument>
            <argument>
               <name>UpdatedProfileConfigInfo</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_ProfileConfigInfo
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>DeleteProfile</name>
         <argumentList>
            <argument>
               <name>ProfileID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_ProfileID
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
```

```
         <name>GetProfileConfigInfo</name>
         <argumentList>
            <argument>
               <name>ProfileID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_ProfileID
               </relatedStateVariable>
            </argument>
            <argument>
               <name>ProfileConfigInfo</name>
               <direction>out</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_ProfileConfigInfo
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <!-- Declarations for other actions defined by UPnP vendor
           (if any)go here. -->

   </actionList>

   <serviceStateTable>

      <stateVariable sendEvents="no">
         <name>SystemInfo</name>
         <dataType>string</dataType>
      </stateVariable>

      <stateVariable sendEvents="no">
         <name>TransportAgentCapabilities</name>
         <dataType>string</dataType>
      </stateVariable>

      <stateVariable sendEvents="no">
         <name>CredentialDelivery</name>
         <dataType>string</dataType>
      </stateVariable>

      <stateVariable sendEvents="yes">
         <name>CredentialsList</name>
         <dataType>string</dataType>
      </stateVariable>

      <stateVariable sendEvents="no">
         <name>ProfileList</name>
         <dataType>string</dataType>
      </stateVariable>

      <stateVariable sendEvents="no">
         <name>A_ARG_TYPE_ProfileConfigInfo</name>
         <dataType>string</dataType>
      </stateVariable>

      <stateVariable sendEvents="no">
```

```
         <name>A_ARG_TYPE_ProfileID</name>
         <dataType>ui4</dataType>
      </stateVariable>

      <!-- Declarations for other state variables defined by UPnP vendor
           (if any)go here. -->

   </serviceStateTable>
</scpd>
```

4 Test

No semantic tests have been specified for this service.

Appendix A. RATransportAgent Data Structures (Normative)

A.1 ProfileList Template

The following shows the generalized layout of a ProfileList Template. More elements and/or attributes MAY be added in future versions of ProfileList templates.

The *forum* character style is used to indicate names defined by the RAWC. Implementations need to fill out the parts that are printed in *vendor* character style.

```
<?xml version="1.0" encoding="UTF-8"?>
<tads xmlns="urn:schemas-upnp-org:ra:tads"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:ra:tads
   http://www.upnp.org/schemas/ra/tads-v1.xsd">
   <profileList dataStructureType="client">
      <profileInfo
         id="profile unique id"
         transportAgentName="transport agent name">
         friendly description
      </profileInfo>
      <!-- Other profiles (if any) go here. -->
   </profileList>
</tads>
```

xml
REQUIRED for all XML documents. Case sensitive.

tads
REQUIRED. Must have "urn:schemas-upnp-org:ra:tads" as the value for the xmlns attribute; this references the UPnP Remote Access Working Committee RATA Datastructure Template Schema. As long as the same xmlns is used, the data structure template MUST be backward compatible, i.e. usable by legacy implementations.

profileList
REQUIRED. Enumerates a set of configured profiles. There MUST be a **profile** element for each configured profile on the RATA.

@dataStructureType
REQUIRED. xs:token. Identifies the data structure type. The token value MUST be client or server.

profileInfo
OPTIONAL. xs:string. Contains a friendly name of the profile. MUST contain the following attributes:

@id
REQUIRED. xs:integer. Unique profile ID.

@transportAgentName
REQUIRED. xs:string. Contains the identification name of the Remote Access Transport mechanism. Possible values are:

"OpenVPN"
"IPSec"

Vendors may define other values.

A.2 ProfileConfig Template

The following shows the generalized layout of a ProfileConfig Template. More elements and/or attributes MAY be added in future versions of ProfileConfig templates.

The *forum* character style is used to indicate names defined by the RAWC. Implementations need to fill out the parts that are printed in *vendor* character style.

```
<?xml version="1.0" encoding="UTF-8"?>
<tads
   xmlns="urn:schemas-upnp-org:ra:tads"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:ra:tads
   http://www.upnp.org/schemas/ra/tads-v1.xsd">
   <profileConfig dataStructureType="client">
      <profileInfo
         id=" profile unique id"
         transportAgentName="transport agent name">
         friendly description
      </profileInfo>
      <profileData>
         <!-- Placeholder for defining data specific for each transport
             mechanism. Data structures defined in another namespace -->
      </profileData>
   </profileConfig>
</tads>
```

xml
REQUIRED for all XML documents. Case sensitive.

tads
REQUIRED. Must have "urn:schemas-upnp-org:ra:tads" as the value for the xmlns attribute; this references the UPnP Remote Access Working Committee RATA Datastructure Template Schema. As long as the same xmlns is used, the data structure template MUST be backward compatible, i.e. usable by legacy implementations.

profileConfig
REQUIRED. Enumerates a set of configured profiles. There MUST be a **profile** element for each configured profile on the RATA. MUST contain the following sub elements:

@dataStructureType
REQUIRED. xs:token. Identifies the data structure type. The token value MUST be "client" or "server".

profileInfo
REQUIRED. xs:string. Contains a friendly name of the profile. MUST contain the following attributes:

@id
REQUIRED. xs:integer. Unique profile ID.

@transportAgentName
REQUIRED. xs:string. Contains the identification name of the Remote Access Transport mechanism. Possible values are:
"OpenVPN"
"IPSec"
Vendors may define other values.

profileData
REQUIRED. xs:any. Contains the required configuration options and parameters for the respective Remote Access profile. The content is specific for each transport agent type and is defined in a schema specific to the transport agent type in use. See C.1.2 for an example schema defined for IPSec.

A.3 TransportAgentCapabilities Template

The following shows the generalized layout of a TransportAgentCapabilities Template. More elements and/or attributes MAY be added in future versions of TransportAgentCapabilities templates.

The *forum* character style is used to indicate names defined by the RAWC. Implementations need to fill out the parts that are printed in *vendor* character style.

```
<?xml version="1.0" encoding="UTF-8"?>
<tads
   xmlns="urn:schemas-upnp-org:ra:tads"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:ra:tads
   http://www.upnp.org/schemas/ra/tads-v1.xsd">
   <transportAgentCapability
      transportAgentName="IPsec">
      <transportAgentOptions>
         <!-- Placeholder for defining data specific for each transport
             mechanism. Data structures defined in another namespace -->
      </transportAgentOptions>
      <!-- Other transport agent options (if any) go here. -->
   </transportAgentCapability>
   <!-- Other transport agent capabilities (if any) go here. -->
</tads>
```

xml
REQUIRED for all XML documents. Case sensitive.

tads
REQUIRED. Must have “urn:schemas-upnp-org:ra:tads” as the value for the xmlns attribute; this references the UPnP Remote Access Working Committee RATA Datastructure Template Schema. As long as the same xmlns is used, the data structure template MUST be backward compatible, i.e. usable by legacy implementations.

transportAgentCapability
REQUIRED. Contains the available options for a particular transport agent. There MUST be a **transportAgentCapability** element for each transport agent supported by the RATA. MUST contain the following sub element:

@transportAgentName
REQUIRED. xs:string. Identifies the transport agent. Possible values are:
“OpenVPN”
“IPSec”
Vendors may define other values.

transportAgentOptions
REQUIRED. xs:any. Contains the options that are supported by the transport agent identified by the **@transportAgentName**. Typically, this data structure can be seen as a template for configuring profiles. There MAY be several **transportAgentOptions** elements for a particular transport agent, each defining a different set of options. The content is specific for each transport agent type and is defined in a schema specific for the transport agent type in use. See C.1.1 for an example schema defined for IPSec.

A.4 CredentialDelivery Template

The following shows the generalized layout of a CredentialDelivery Template. More elements and/or attributes MAY be added in future versions of CredentialDelivery templates.

The *forum* character style is used to indicate names defined by the RAWC. Implementations need to fill out the parts that are printed in *vendor* character style.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<tads xmlns="urn:schemas-upnp-org:ra:tads"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:ra:tads
   http://www.upnp.org/schemas/ra/tads-v1.xsd">
   <credentialDelivery credentialDeliveryMechanism="mechanism name">
      <credentialType credentialEncoding="RSA Raw Key">
         RSA
      </credentialType>
      <!-- Other credential types (if any) go here. -->
   </credentialDelivery>
   <!-- Other credential delivery (if any) go here. -->
</tads>
```

xml
REQUIRED for all XML documents. Case sensitive.

tads
REQUIRED. Must have "urn:schemas-upnp-org:ra:tads" as the value for the xmlns attribute; this references the UPnP Remote Access Working Committee RATA Datastructure Template Schema. As long as the same xmlns is used, the data structure template MUST be backward compatible, i.e. usable by legacy implementations.

credentialDelivery
REQUIRED. Contains the available options for a credential delivery mechanism. There MUST be a **credentialDelivery** element for each credential delivery mechanism supported by the RATA. MUST contain the following sub elements:

@credentialDeliveryMechanism
REQUIRED. xs:string. Identifies the credential delivery mechanism. Possible values are:

"NFC"	Near-field communication
"FTP"	File Transfer Protocol
"HTTP"	*Hyper-text Transfer Protocol*

Vendors may define other values.

credentialType
REQUIRED. xs:string. Identifies the credential type that can be delivered by the credential delivery mechanism. There MUST be a **credentialType** element for each credential type that can be delivered. MUST contain the following attribute:

@credentialEncoding
REQUIRED. xs:string. Identifies the encoding of the particular certificate type.

A.5 CredentialsList Template

The following shows the generalized layout of a CredentialsList Template. More elements and/or attributes MAY be added in future versions of CredentialsList templates.

The *forum* character style is used to indicate names defined by the RAWC. Implementations need to fill out the parts that are printed in *vendor* character style.

```
<?xml version="1.0" encoding="UTF-8"?>
<tads xmlns="urn:schemas-upnp-org:ra:tads"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:ra:tads
   http://www.upnp.org/schemas/ra/tads-v1.xsd">
   <credentialsList>
      <credential scope="local">
         <credentialID>ID</credentialID>
         <credentialFriendlyName>friendly name</credentialFriendlyName>
         <credentialType credentialEncoding="RSA Raw
Key">RSA</credentialType>
      </credential>
      <!-- Other credential (if any) go here. -->
   </credentialsList>
</tads>
```

xml
REQUIRED for all XML documents. Case sensitive.

tads
REQUIRED. Must have “urn:schemas-upnp-org:ra:tads” as the value for the xmlns attribute; this references the UPnP Remote Access Working Committee RATA Datastructure Template Schema. As long as the same xmlns is used, the data structure template MUST be backward compatible, i.e. usable by legacy implementations.

credentialsList
REQUIRED. Contains the available credentials available on the RATA. MUST contain the following sub elements:

credential
REQUIRED. Contains the credential information. There MUST be a **credential** element for each credential available on RATA. MUST contain the following sub elements:

@scope
REQUIRED. xs:token. Indicates if the credential is associated with the local or a remote RADA.

credentialID
REQUIRED. xs:string. Uniquely identifies the credential on the RATA.

credentialFriendlyName
REQUIRED. xs:string. Friendly name of the credential. Used to identify the credential to the user.

credentialType
REQUIRED. xs:string. Determines the credential type. MUST contain the following attribute:

@credentialEncoding
REQUIRED. xs:string. Identifies the encoding of the particular certificate type.

A.6 TransportAgent Datastructure Schema

```
xsi:schemaLocation="
   urn:schemas-upnp-org:ra:tads
      http://www.upnp.org/schemas/ra/tads-v1.xsd"
```

where the number 1 after the “v” is the version number. Each TADS schema version update must be backward compatible with the previous version. Specifically, XML elements and/or attributes may be added to more recent TADS schema versions, but must not ever be removed. As a result, when examining the schema version value, implementations will likely want to perform a greater-than-or-equal-to comparison rather than just a plain equality check.

Appendix B. Addressing Considerations (Informative)

B.1 IPv4 Considerations

B.1.1 IPv4 Address Allocation

In order to be able to interact with remote devices, home UPnP devices must be correctly configured to access the internet first; in practice this means that home UPnP devices must acquire an IP address from the home DHCP server.

B.1.2 Address Space Collisions

It is expected that remote UPnP device could use other home networks as access networks, e.g. when visiting a friend’s home. In such environments there is a highly probability that the access network’s residential gateway is configured to allocate IP addresses in the same address space as the residential gateway in the home network. The address space collision problem is facilitated by ISP’s practice to configure all the residential gateways with the same settings for the LAN interface. Also, in the cases when the residential gateway is not provided by the ISP and is purchased from a retail chain, the consumers will use them directly with the manufacturer settings which typically are the same for all devices from a particular manufacturer. This will make almost impossible the remote access when both the home network and the access network are connected to the Internet via the same ISP or when both networks have a residential gateway from the same manufacturer used with the default settings.

The address space collision leads to basic routing problems that will prevent packets originating in the remote device to reach the devices in its home network unless the remote device is a multi-home aware device.

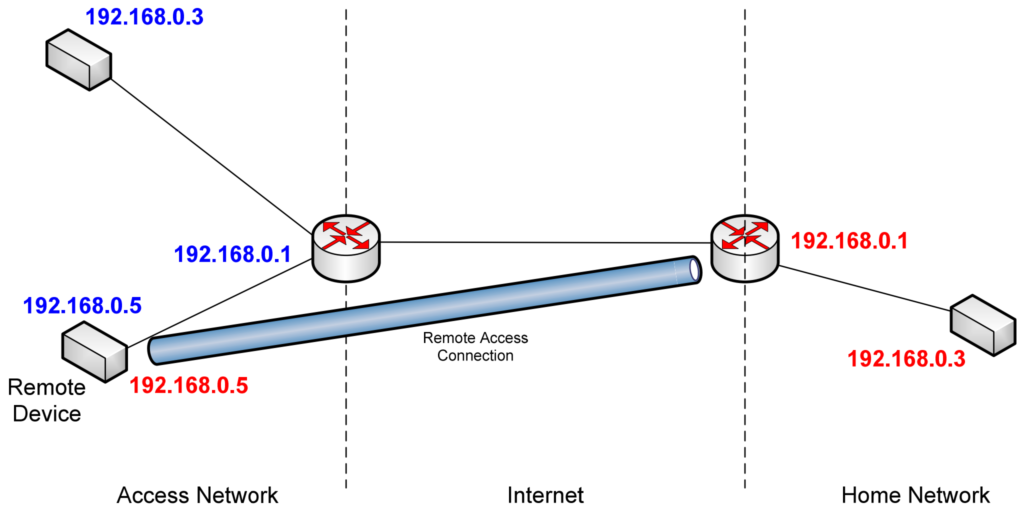

Figure B-1: Address Space Collision Problem.

In order to reduce the probability of address space collision, the home owner can reconfigure the home network, during a setup procedure performed once, to use a random address space. To automate the procedure, it is recommended to use the LANHostConfigManagement service available on the IGDv1 [IGD] compatible residential gateways.

It must be noted that this procedure does not eliminate the possibility of address space collisions but will lead to a situation where, in practice, it will be highly unlikely that the access network and home network will be sharing the same address space. The transition to IPv6 will eliminate the problem of address space collision.

Appendix C. Using IPsec as Remote Access Transport (Normative)

The purpose of this section is to describe how to use a subset of IPsec protocol suite as the Remote Access transport mechanism in UPnP Remote Access.

This section may serve as a model for defining additional Remote Access Transport mechanism.

C.1 IPsec Templates

C.1.1 IPsec Options Template

The current IPsec options template is intended to be used as a template for configuring Remote Access IPsec profiles. Each IPsec option contains a set of cryptographic algorithms and protocols associated with a single authentication method. If the IPsec supports multiple authentication methods, an IPsec option MUST be defined for each supported authentication method, e.g. RSA digital signatures, shared secret, EAP.

```
<?xml version="1.0" encoding="UTF-8"?>
<ipsecOPT xmlns="urn:schemas-upnp-org:ra:tacfg:ipsec"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

   xsi:schemaLocation="urn:schemas-upnp-org:ra:tacfg:ipsec
   http://www.upnp.org/schemas/ra/tacfg-ipsec-v1.xsd"
   authenticationMethod="RSA Digital Signature"
   credentialEncoding="PKCS #7 wrapped X.509 certificate"
   keyExchangeProtocol="IKEv2">
   <encryptionAlgorithm>AES_CBC</encryptionAlgorithm>
   <authenticationAlgorithm>HMAC_SHA1_96</authenticationAlgorithm>
   <integrityAlgorithm>AES_XCBC_96</integrityAlgorithm>
   <pseudoRandomFunction>AES128_XCBC</pseudoRandomFunction>
</ipsecOPT>
```

xml
REQUIRED for all XML documents. Case sensitive.

ipsecOPT
REQUIRED. Must have "urn:schemas-upnp-org:ra:tacfg:ipsec" as the value for the xmlns attribute; this references the UPnP Remote Access Working Committee RATA IPsec Options Template Schema. As long as the same xmlns is used, the data structure template MUST be backward compatible, i.e. usable by legacy implementations. Contains the following attributes and sub elements:

@authenticationMethod
REQUIRED. xs:token. Determines the authentication method used.

@credentialEncoding
REQUIRED. xs:token. Determines the encoding used for the credential specific to method described in **@authenticationMethod**.

@keyExchangeProtocol
REQUIRED. xs:token. Determines the key exchange protocol for this ipsec option. Possible values are "IKEv1" and "IKEv2".

encryptionAlgorithm
REQUIRED. xs:token. Determines the encryption algorithm to be used with this IPsec option. If multiple encryption algorithms are supported, they MUST be listed here in the order of preference.

authenticationAlgorithm
REQUIRED. xs:token. Determines the authentication algorithm to be used with this IPsec option. If multiple authentication algorithms are supported, they MUST be listed here in the order of preference.

integrityAlgorithm

REQUIRED. xs:token. Determines the integrity algorithm to be used with this IPsec option. If multiple integrity algorithms are supported, they MUST be listed here in the order of preference.

pseudoRandomFunction

REQUIRED. xs:token. Determines the pseudo random function to be used with this IPsec option. If multiple pseudo random function are supported, they MUST be listed here in the order of preference.

C.1.2 IPsec Configuration Template

The current IPsec configuration template is primarily designed to be used with IKEv2 but it contains enough information to allow IKEv1 to be used.

```
<?xml version="1.0" encoding="UTF-8"?>
<ipsecCFG xmlns="urn:schemas-upnp-org:ra:tacfg:ipsec"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:ra:tacfg:ipsec
   http://www.upnp.org/schemas/ra/tacfg-ipsec-v1.xsd"
   configurationType="client">
   <policy>
      <perfectForwardSecrecy>true</perfectForwardSecrecy>
      <replayWindowLength>10</replayWindowLength>
      <remoteIdentity>MyIdentity</remoteIdentity>
      <proposal protocol="ESP">
         <encryptionAlgorithm keyLength="256">
            AES_CBC
         </encryptionAlgorithm>
         <integrityAlgorithm> </integrityAlgorithm>
         <lifetime>
            <seconds>28800</seconds>
            <kBytes>5000</kBytes>
         </lifetime>
      </proposal>
   </policy>
   <ike version="IKEv2">
      <remoteAddress>X.X.X.X</remoteAddress>
      <sendNotification>true</sendNotification>
      <idType>ID_DER_ASN1_DN</idType>
      <useIPsecExpire>true</useIPsecExpire>
      <useReplayDetection>true</useReplayDetection>
      <useInternalAddress>true</useInternalAddress>
      <dpdHeartbeat>600</dpdHeartbeat>
      <natKeepalive>100</natKeepalive>
      <rekeyingThreshold>90</rekeyingThreshold>
      <proposal protocol="IKE">
         <encryptionAlgorithm keyLength="256">
            AES_CBC
         </encryptionAlgorithm>
         <integrityAlgorithm>AES_XCBC_96</integrityAlgorithm>
         <pseudoRandomFunction>AES128_XCBC</pseudoRandomFunction>
         <groupDescription>MODP_1536</groupDescription>
         <groupType>Group_2</groupType>
         <lifetime>
            <seconds>28800</seconds>
            <kBytes>500</kBytes>
         </lifetime>
      </proposal>
      <authenticationMethod>RSA Digital Signature</authenticationMethod>
```

```
        <credentialID>100</credentialID>
    </ike>
</ipsecCFG>
```

xml
REQUIRED for all XML documents. Case sensitive.

ipsecCFG
REQUIRED. Must have “urn:schemas-upnp-org:ra:tacfg:ipsec” as the value for the xmlns attribute; this references the UPnP Remote Access Working Committee RATA IPsec Configuration Template Schema. As long as the same xmlns is used, the data structure template MUST be backward compatible, i.e. usable by legacy implementations. Contains the following attributes and sub elements:

@configurationType
REQUIRED. xs:token. Determines the type of configuration. Possible values are “client” or “server”.

policy
REQUIRED. Enumerates a set of parameters needed for configuring IPsec SAs. Contains the following sub elements:

perfectForwardSecrecy
REQUIRED. xs.token. Determines if IKE initiates a new Diffie-Hellman exchange to obtain new master key keying material for each new session key that IPsec SAs require.

replayWindowLength
REQUIRED. xs.positiveInteger. Determine if the antireplay service for the IPsec SA is used. The maximum supported value is 32.

remoteIdentity
REQUIRED. xs.string. Determines the identity of the remote host.

proposal
REQUIRED. Contains a set of attributes that are used when initiating an IPsec SA negotiation. Contains the following sub elements:

@protocol
REQUIRED. xs:token. Describes the proposal type. The allowed value is ESP.

encryptionAlgorithm
REQUIRED. xs:token. Describes the encryption algorithm proposed. Allowed values are defined in IKEv2 [RFC 4306]. Contains the following attribute:

@keyLength
OPTIONAL. xs:integer. Describes the key length of the encryption algorithm proposed. MAY be present in any instance if the encryption algorithm mentioned in **encryptionAlgorithm** permits variable key lengths. Allowed values are defined in IKEv2.

integrityAlgorithm
REQUIRED. xs:token. Describes the authenticathion algorithm proposed as recommended by RFC 2406. Allowed values are defined in IKEv2.

lifetime
REQUIRED. Describes the maximum lifetime of the IKE SA. Contains the following sub elements:

seconds
REQUIRED. xs:integer. Maximum duration of the IKE SA.

kBytes
REQUIRED. xs:integer. Maximum amount of data (in kBytes) that the IKE SA protects.

ike

REQUIRED. Enumerates a set of parameters needed for configuring IKE. Contains the following sub elements:

@version

REQUIRED. xs:token. Determines the IKE protocol version. Possible values are "IKEv1" or "IKEv2".

remoteAddress

OPTIONAL. xs:string. Contains the IP address or the FQDN of the RAS. MUST be present in any instance if the **configurationType** value is *client*.

sendNotification

REQUIRED. xs:boolean. Determines if IKE sends notification messages if error occur. Set the value to TRUE to make troubleshooting easier.

idType

REQUIRED. xs:token. Determines how the RAC identifies itself to the RAS. The allowed values are specified in IKEv2.

useIPsecExpire

REQUIRED. xs:boolean. Determines how IPsec SAs expire:

- TRUE when the IKE SA that was used to negotiate them expires or is deleted,
- FALSE according to their lifetime.

useReplayDetection

REQUIRED. xs:boolean. Determines whether the responder performs antireplay detection:

- TRUE replay detection is enabled,
- FALSE replay detection is disabled.

useInternalAddress

OPTIONAL. xs:boolean. Determines whether the RAC aquires an IP address from the hone network address pool making it virtually part of the home network. Default value is TRUE. MUST be present in any instance if the **configurationType** value is *client*.

useNATProbe

OPTIONAL. xs:boolean. Determines whether the RAC is using the automatic NAT detection. Functionality is defined in [RFC 3947]. Default value is TRUE. MUST be present in any instance if the **configurationType** value is *client*.

dpdHeartbeat

OPTIONAL. xs:integer. Determines how often the RAC uses the Dead Peer Detection (DPD) feature defined in [RFC 3706]. MUST be present in any instance if the **configurationType** value is *client*.

natKeepalive

OPTIONAL. xs:integer. Determines how often the RAC sends an empty UDP packet to port 4500 of the RAS. Default value is 120 seconds. Functionality is defined in [RFC 3947]. MUST be present in any instance if the **configurationType** value is *client*.

rekeyingThreshold

REQUIRED. xs:integer. Starts the IKE SA rekeying when the specified percentage of the IKE SA expiration timeout is reached. Accepted percentage values are in the range of 70 to 95.

proposal

REQUIRED. Contains a set of attributes that are used when initiating an IKE negotiation. Contains the following sub elements:

@protocol

REQUIRED. xs:token. Describes the proposal type. The allowed value is IKE.

encryptionAlgorithm

REQUIRED. xs:token. Describes the encryption algorithm proposed. Allowed values are defined in IKEv2. Contains the following attribute:

keyLength
OPTIONAL. xs:integer. Describes the key length of the encryption algorithm proposed. MAY be present in any instance if the encryption algorithm mentioned in **encryptionAlgorithm** permits variable key lengths. Allowed values are defined in IKEv2.

integrityAlgorithm
REQUIRED. xs:token. Describes the authenticathion algorithm proposed. Allowed values are defined in IKEv2.

pseudoRandomFunction
REQUIRED. xs:token. Describes the pseudo random function proposed. Allowed values are defined in IKEv2.

groupDescription
REQUIRED. xs:token. Describes the group to use during Diffie-Hellman (DH) exchange. Allowed values are defined in IKEv2.

groupType
REQUIRED. xs:token. Describes the type of DH group used (e.g. modular or elliptic). Allowed values are defined in IKEv2.

lifetime
REQUIRED. Describes the maximum lifetime of the IKE SA. Contains the following sub elements:

seconds
REQUIRED. xs:integer. Maximum duration of the IKE SA.

kBytes
REQUIRED. xs:integer. Maximum amount of data (in kBytes) that the IKE SA protects.

authenticationMethod
REQUIRED. xs:token. Contains the method of authentication used. Allowed values are defined in IKEv2.

credentialID
REQUIRED. xs:string. Contains the unique ID of a credential stored on the RATA.

C.2 Sample IPsec Files

C.2.1 Sample IPSec based on certificates

C.2.1.1 Sample IPsec TransportAgentCapabilities

This simple TransportAgentCapabilities file describes the capability of the IPsec engine.

```
<?xml version="1.0" encoding="UTF-8"?>
<tads xmlns="urn:schemas-upnp-org:ra:tads"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:cfg="urn:schemas-upnp-org:ra:tacfg:ipsec"
   xsi:schemaLocation="urn:schemas-upnp-org:ra:tads
   http://www.upnp.org/schemas/ra/tads-v1.xsd
   urn:schemas-upnp-org:ra:tacfg:ipsec
   http://www.upnp.org/schemas/ra/tacfg-ipsec-v1.xsd">
   <transportAgentCapability transportAgentName="IPsec">
      <transportAgentOptions>
         <cfg:ipsecOPT
            authenticationMethod="RSA Digital Signature"
            credentialEncoding="PKCS #7 wrapped X.509 certificate"
            keyExchangeProtocol="IKEv2">
            <cfg:encryptionAlgorithm>AES_CBC</cfg:encryptionAlgorithm>
            <cfg:encryptionAlgorithm>AES_CTR</cfg:encryptionAlgorithm>
            <cfg:authenticationAlgorithm></cfg:authenticationAlgorithm>
            <cfg:integrityAlgorithm>AES_XCBC_96</cfg:integrityAlgorithm>
            <cfg:pseudoRandomFunction>
               AES128_XCBC
            </cfg:pseudoRandomFunction>
         </cfg:ipsecOPT>
      </transportAgentOptions>
   </transportAgentCapability>
</tads>
```

C.2.1.2 Sample IPsec ConfigInfo for Server

This sample ConfigInfo file instructs the server to present to IKE proposals for any correspondent IPsec peer that try to establish connectivity and the established IKE SA will be used to negociate the use of ESP with AES_CBC cipher suit with a 256 key length. The ESP SA expires after 28800 seconds or 5000 kBytes transferred.

```
<?xml version="1.0" encoding="UTF-8"?>
<tads
   xmlns="urn:schemas-upnp-org:ra:tads"
   xmlns:cfg="urn:schemas-upnp-org:ra:tacfg:ipsec"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:ra:tads
   http://www.upnp.org/schemas/ra/tads-v1.xsd
   urn:schemas-upnp-org:ra:tacfg:ipsec
   http://www.upnp.org/schemas/ra/tacfg-ipsec-v1.xsd">
   <profileConfig dataStructureType="server">
      <profileInfo id="12" transportAgentName="IPsec">
         IPsec configuration
      </profileInfo>
      <profileData>
         <cfg:ipsecCFG configurationType="server">
            <cfg:policy>
               <cfg:perfectForwardSecrecy>
                  true
               </cfg:perfectForwardSecrecy>
               <cfg:replayWindowLength>10</cfg:replayWindowLength>
               <cfg:remoteIdentity>bob@home.com</cfg:remoteIdentity>
               <cfg:proposal protocol="ESP">
                  <cfg:encryptionAlgorithm keyLength="256">
                     AES_CBC
                  </cfg:encryptionAlgorithm>
                  <cfg:lifetime>
                     <cfg:seconds>28800</cfg:seconds>
                     <cfg:kBytes>5000</cfg:kBytes>
                  </cfg:lifetime>
               </cfg:proposal>
            </cfg:policy>
            <cfg:ike version="IKEv2">
               <cfg:sendNotification>true</cfg:sendNotification>
               <cfg:idType>ID_DER_ASN1_DN</cfg:idType>
               <cfg:useIPsecExpire>true</cfg:useIPsecExpire>
               <cfg:useReplayDetection>true</cfg:useReplayDetection>
               <cfg:rekeyingThreshold>90</cfg:rekeyingThreshold>
               <cfg:proposal protocol="IKE">
                  <cfg:encryptionAlgorithm keyLength="256">
```

```
                     AES_CBC
                  </cfg:encryptionAlgorithm>
                  <cfg:integrityAlgorithm>
                     AES_XCBC_96
                  </cfg:integrityAlgorithm>
                  <cfg:pseudoRandomFunction>
                     AES128_XCBC
                  </cfg:pseudoRandomFunction>
                  <cfg:groupDescription>MODP_1536</cfg:groupDescription>
                  <cfg:groupType>MODP</cfg:groupType>
                  <cfg:lifetime>
                     <cfg:seconds>28800</cfg:seconds>
                     <cfg:kBytes>5000</cfg:kBytes>
                  </cfg:lifetime>
               </cfg:proposal>
               <cfg:proposal protocol="IKE">
                  <cfg:encryptionAlgorithm keyLength="128">
                     AES_CBC
                  </cfg:encryptionAlgorithm>
                  <cfg:integrityAlgorithm>
                     AES_XCBC_96
                  </cfg:integrityAlgorithm>
                  <cfg:pseudoRandomFunction>
                     AES128_XCBC
                  </cfg:pseudoRandomFunction>
                  <cfg:groupDescription>MODP_1024</cfg:groupDescription>
                  <cfg:groupType>MODP</cfg:groupType>
                  <cfg:lifetime>
                     <cfg:seconds>28800</cfg:seconds>
                     <cfg:kBytes>5000</cfg:kBytes>
                  </cfg:lifetime>
               </cfg:proposal>
               <cfg:authenticationMethod>
                  RSA Digital Signature
               </cfg:authenticationMethod>
               <cfg:credentialID>100</cfg:credentialID>
            </cfg:ike>
         </cfg:ipsecCFG>
      </profileData>
   </profileConfig>
</tads>
```

C.2.1.3 Sample IPsec ConfigInfo for Client

This ConfigInfo file allows a IPsec client to establish IPsec connection with the IPsec server configured in section C.2.1.2. The IKE and ESP proposals will match the ones from the server.

```
<?xml version="1.0" encoding="UTF-8"?>
<tads
   xmlns="urn:schemas-upnp-org:ra:tads"
   xmlns:cfg="urn:schemas-upnp-org:ra:tacfg:ipsec"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:ra:tads
   http://www.upnp.org/schemas/ra/tads-v1.xsd
   urn:schemas-upnp-org:ra:tacfg:ipsec
   http://www.upnp.org/schemas/ra/tacfg-ipsec-v1.xsd">
   <profileConfig dataStructureType="client">
      <profileInfo id="12" transportAgentName="IPsec">
         IPsec configuration
      </profileInfo>
      <profileData>
         <cfg:ipsecCFG configurationType="client">
            <cfg:policy>
               <cfg:perfectForwardSecrecy>
                  true
               </cfg:perfectForwardSecrecy>
               <cfg:replayWindowLength>10</cfg:replayWindowLength>
               <cfg:remoteIdentity>alice@home.com</cfg:remoteIdentity>
               <cfg:proposal protocol="ESP">
                  <cfg:encryptionAlgorithm keyLength="256">
                     AES_CBC
                  </cfg:encryptionAlgorithm>
                  <cfg:lifetime>
                     <cfg:seconds>28800</cfg:seconds>
                     <cfg:kBytes>5000</cfg:kBytes>
                  </cfg:lifetime>
               </cfg:proposal>
            </cfg:policy>
            <cfg:ike version="IKEv2">
               <cfg:remoteAddress>129.178.89.81</cfg:remoteAddress>
               <cfg:sendNotification>true</cfg:sendNotification>
               <cfg:idType>ID_DER_ASN1_DN</cfg:idType>
               <cfg:useIPsecExpire>true</cfg:useIPsecExpire>
               <cfg:useReplayDetection>true</cfg:useReplayDetection>
               <cfg:useInternalAddress>true</cfg:useInternalAddress>
               <cfg:dpdHeartbeat>600</cfg:dpdHeartbeat>
               <cfg:natKeepalive>100</cfg:natKeepalive>
               <cfg:rekeyingThreshold>90</cfg:rekeyingThreshold>
               <cfg:proposal protocol="IKE">
                  <cfg:encryptionAlgorithm keyLength="256">
                     AES_CBC
                  </cfg:encryptionAlgorithm>
                  <cfg:integrityAlgorithm>
                     AES_XCBC_96
                  </cfg:integrityAlgorithm>
                  <cfg:pseudoRandomFunction>
                     AES128_XCBC
                  </cfg:pseudoRandomFunction>
                  <cfg:groupDescription>MODP_1536</cfg:groupDescription>
                  <cfg:groupType>MODP</cfg:groupType>
                  <cfg:lifetime>
                     <cfg:seconds>28800</cfg:seconds>
                     <cfg:kBytes>5000</cfg:kBytes>
                  </cfg:lifetime>
               </cfg:proposal>
               <cfg:authenticationMethod>
                  RSA Digital Signature
               </cfg:authenticationMethod>
               <cfg:credentialID>100</cfg:credentialID>
            </cfg:ike>
         </cfg:ipsecCFG>
      </profileData>
   </profileConfig>
</tads>
```

C.2.2 Sample IPSec based on shared key null policy

C.2.2.1 Sample IPsec TransportAgentCapabilities

This simple TransportAgentCapabilities file describes the capability of the IPsec engine.

```
<?xml version="1.0" encoding="UTF-8"?>
<tads xmlns="urn:schemas-upnp-org:ra:tads"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:cfg="urn:schemas-upnp-org:ra:tacfg:ipsec"
   xsi:schemaLocation="urn:schemas-upnp-org:ra:tads
   http://www.upnp.org/schemas/ra/tads-v1.xsd
   urn:schemas-upnp-org:ra:tacfg:ipsec
   http://www.upnp.org/schemas/ra/tacfg-ipsec-v1.xsd">
   <transportAgentCapability transportAgentName="IPsec">
      <transportAgentOptions>
         <cfg:ipsecOPT
            authenticationMethod="Shared Key Message Integrity Code"
            credentialEncoding="Pre-Shared Key"
            keyExchangeProtocol="IKEv2">
            <cfg:encryptionAlgorithm>NULL</cfg:encryptionAlgorithm>
            <cfg:authenticationAlgorithm></cfg:authenticationAlgorithm>
           <cfg:integrityAlgorithm>HMAC_SHA1_96</cfg:integrityAlgorithm>
            <cfg:pseudoRandomFunction>
               HMAC_SHA1
            </cfg:pseudoRandomFunction>
         </cfg:ipsecOPT>
      </transportAgentOptions>
   </transportAgentCapability>
</tads>
```

C.2.2.2 Sample IPsec ConfigInfo for Server

```
<?xml version="1.0" encoding="UTF-8"?>
<tads
   xmlns="urn:schemas-upnp-org:ra:tads"
   xmlns:cfg="urn:schemas-upnp-org:ra:tacfg:ipsec"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:ra:tads
   http://www.upnp.org/schemas/ra/tads-v1.xsd
   urn:schemas-upnp-org:ra:tacfg:ipsec
   http://www.upnp.org/schemas/ra/tacfg-ipsec-v1.xsd">
   <profileConfig dataStructureType="client">
      <profileInfo id="12" transportAgentName="IPsec">
         IPsec configuration
      </profileInfo>
      <profileData>
         <cfg:ipsecCFG configurationType="server">
            <cfg:policy>
               <cfg:perfectForwardSecrecy>
                  true
               </cfg:perfectForwardSecrecy>
               <cfg:replayWindowLength>10</cfg:replayWindowLength>
               <cfg:remoteIdentity>bob@home.com</cfg:remoteIdentity>
               <cfg:proposal protocol="ESP">
                  <cfg:encryptionAlgorithm>
                     NULL
                  </cfg:encryptionAlgorithm>
                  <integrityAlgorithm>
                     HMAC_SHA1_96
                  </integrityAlgorithm>
                  <cfg:lifetime>
                     <cfg:seconds>28800</cfg:seconds>
                     <cfg:kBytes>5000000</cfg:kBytes>
                  </cfg:lifetime>
               </cfg:proposal>
            </cfg:policy>
            <cfg:ike version="IKEv2">
               <cfg:sendNotification>true</cfg:sendNotification>
               <cfg:idType>ID_KEY_ID</cfg:idType>
               <cfg:useIPsecExpire>true</cfg:useIPsecExpire>
               <cfg:useReplayDetection>true</cfg:useReplayDetection>
               <cfg:rekeyingThreshold>90</cfg:rekeyingThreshold>
               <cfg:proposal protocol="IKE">
                  <cfg:encryptionAlgorithm keyLength="128">
                     AES_CBC
                  </cfg:encryptionAlgorithm>
                  <cfg:integrityAlgorithm>
                     HMAC_SHA1_96
                  </cfg:integrityAlgorithm>
                  <cfg:pseudoRandomFunction>
                     HMAC_SHA1
                  </cfg:pseudoRandomFunction>
                  <cfg:groupDescription>MODP_768</cfg:groupDescription>
                  <cfg:groupType>MODP</cfg:groupType>
                  <cfg:lifetime>
                     <cfg:seconds>28800</cfg:seconds>
                     <cfg:kBytes>5000</cfg:kBytes>
                  </cfg:lifetime>
               </cfg:proposal>
               <cfg:authenticationMethod>
                  Shared Key Message Integrity Code
               </cfg:authenticationMethod>
               <cfg:credentialID>100</cfg:credentialID>
            </cfg:ike>
         </cfg:ipsecCFG>
      </profileData>
   </profileConfig>
</tads>
```

C.2.2.3 Sample IPsec ConfigInfo for Client

This ConfigInfo file allows a IPsec client to establish IPsec connection with the IPsec server configured in section 0.

```
<?xml version="1.0" encoding="UTF-8"?>
<tads
   xmlns="urn:schemas-upnp-org:ra:tads"
   xmlns:cfg="urn:schemas-upnp-org:ra:tacfg:ipsec"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:ra:tads
   http://www.upnp.org/schemas/ra/tads-v1.xsd
   urn:schemas-upnp-org:ra:tacfg:ipsec
   http://www.upnp.org/schemas/ra/tacfg-ipsec-v1.xsd">
   <profileConfig dataStructureType="client">
      <profileInfo id="12" transportAgentName="IPsec">
         IPsec configuration
      </profileInfo>
      <profileData>
         <cfg:ipsecCFG configurationType="client">
            <cfg:policy>
               <cfg:perfectForwardSecrecy>
                  true
               </cfg:perfectForwardSecrecy>
               <cfg:replayWindowLength>10</cfg:replayWindowLength>
               <cfg:remoteIdentity>alice@home.com</cfg:remoteIdentity>
               <cfg:proposal protocol="ESP">
                  <cfg:encryptionAlgorithm>
                     NULL
                  </cfg:encryptionAlgorithm>
                  <integrityAlgorithm>
                     HMAC_SHA1_96
                  </integrityAlgorithm>
                  <cfg:lifetime>
                     <cfg:seconds>28800</cfg:seconds>
                     <cfg:kBytes>5000000</cfg:kBytes>
                  </cfg:lifetime>
               </cfg:proposal>
            </cfg:policy>
            <cfg:ike version="IKEv2">
               <cfg:remoteAddress>129.178.89.81</cfg:remoteAddress>
               <cfg:sendNotification>true</cfg:sendNotification>
               <cfg:idType>ID_KEY_ID</cfg:idType>
               <cfg:useIPsecExpire>true</cfg:useIPsecExpire>
               <cfg:useReplayDetection>true</cfg:useReplayDetection>
               <cfg:useInternalAddress>true</cfg:useInternalAddress>
               <cfg:dpdHeartbeat>600</cfg:dpdHeartbeat>
               <cfg:natKeepalive>100</cfg:natKeepalive>
               <cfg:rekeyingThreshold>90</cfg:rekeyingThreshold>
               <cfg:proposal protocol="IKE">
                  <cfg:encryptionAlgorithm keyLength="128">
                     AES_CBC
                  </cfg:encryptionAlgorithm>
                  <cfg:integrityAlgorithm>
                     HMAC_SHA1_96
                  </cfg:integrityAlgorithm>
                  <cfg:pseudoRandomFunction>
                     HMAC_SHA1
                  </cfg:pseudoRandomFunction>
                  <cfg:groupDescription>MODP_768</cfg:groupDescription>
                  <cfg:groupType>MODP</cfg:groupType>
                  <cfg:lifetime>
                     <cfg:seconds>28800</cfg:seconds>
                     <cfg:kBytes>5000</cfg:kBytes>
                  </cfg:lifetime>
               </cfg:proposal>
               <cfg:authenticationMethod>
                  Shared Key Message Integrity Code
               </cfg:authenticationMethod>
               <cfg:credentialID>100</cfg:credentialID>
            </cfg:ike>
         </cfg:ipsecCFG>
      </profileData>
   </profileConfig>
</tads>
```

C.2.3 Sample IPSec based on shared key advanced policy

C.2.3.1 Sample IPsec TransportAgentCapabilities

This simple TransportAgentCapabilities file describes the capability of the IPsec engine.

```
<?xml version="1.0" encoding="UTF-8"?>
<tads xmlns="urn:schemas-upnp-org:ra:tads"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:cfg="urn:schemas-upnp-org:ra:tacfg:ipsec"
   xsi:schemaLocation="urn:schemas-upnp-org:ra:tads
   http://www.upnp.org/schemas/ra/tads-v1.xsd
   urn:schemas-upnp-org:ra:tacfg:ipsec
   http://www.upnp.org/schemas/ra/tacfg-ipsec-v1.xsd">
   <transportAgentCapability transportAgentName="IPsec">
      <transportAgentOptions>
         <cfg:ipsecOPT
            authenticationMethod="Shared Key Message Integrity Code"
            credentialEncoding="Pre-Shared Key"
            keyExchangeProtocol="IKEv2">
            <cfg:encryptionAlgorithm>AES_CBC</cfg:encryptionAlgorithm>
            <cfg:authenticationAlgorithm></cfg:authenticationAlgorithm>
           <cfg:integrityAlgorithm>HMAC_SHA1_96</cfg:integrityAlgorithm>
         <cfg:pseudoRandomFunction>HMAC_SHA1</cfg:pseudoRandomFunction>
         </cfg:ipsecOPT>
      </transportAgentOptions>
   </transportAgentCapability>
</tads>
```

C.2.3.2 Sample IPsec ConfigInfo for Server

```
<?xml version="1.0" encoding="UTF-8"?>
<tads
   xmlns="urn:schemas-upnp-org:ra:tads"
   xmlns:cfg="urn:schemas-upnp-org:ra:tacfg:ipsec"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:ra:tads
   http://www.upnp.org/schemas/ra/tads-v1.xsd
   urn:schemas-upnp-org:ra:tacfg:ipsec
   http://www.upnp.org/schemas/ra/tacfg-ipsec-v1.xsd">
   <profileConfig dataStructureType="client">
      <profileInfo id="12" transportAgentName="IPsec">
         IPsec configuration
      </profileInfo>
      <profileData>
         <cfg:ipsecCFG configurationType="server">
            <cfg:policy>
               <cfg:perfectForwardSecrecy>
                  true
               </cfg:perfectForwardSecrecy>
               <cfg:replayWindowLength>10</cfg:replayWindowLength>
               <cfg:remoteIdentity>bob@home.com</cfg:remoteIdentity>
               <cfg:proposal protocol="ESP">
                  <cfg:encryptionAlgorithm keyLength="128">
                     AES_CBC
                  </cfg:encryptionAlgorithm>
                  <integrityAlgorithm>
                     HMAC_SHA1_96
                  </integrityAlgorithm>
                  <cfg:lifetime>
                     <cfg:seconds>28800</cfg:seconds>
                     <cfg:kBytes>5000000</cfg:kBytes>
                  </cfg:lifetime>
               </cfg:proposal>
            </cfg:policy>
            <cfg:ike version="IKEv2">
               <cfg:sendNotification>true</cfg:sendNotification>
               <cfg:idType>ID_KEY_ID</cfg:idType>
               <cfg:useIPsecExpire>true</cfg:useIPsecExpire>
               <cfg:useReplayDetection>true</cfg:useReplayDetection>
               <cfg:rekeyingThreshold>90</cfg:rekeyingThreshold>
               <cfg:proposal protocol="IKE">
                  <cfg:encryptionAlgorithm keyLength="128">
                     AES_CBC
                  </cfg:encryptionAlgorithm>
                  <cfg:integrityAlgorithm>
                     HMAC_SHA1_96
                  </cfg:integrityAlgorithm>
                  <cfg:pseudoRandomFunction>
                     HMAC_SHA1
                  </cfg:pseudoRandomFunction>
                  <cfg:groupDescription>MODP_1536</cfg:groupDescription>
                  <cfg:groupType>MODP</cfg:groupType>
                  <cfg:lifetime>
                     <cfg:seconds>28800</cfg:seconds>
                     <cfg:kBytes>5000</cfg:kBytes>
                  </cfg:lifetime>
               </cfg:proposal>
               <cfg:authenticationMethod>
                  Shared Key Message Integrity Code
               </cfg:authenticationMethod>
               <cfg:credentialID>100</cfg:credentialID>
            </cfg:ike>
         </cfg:ipsecCFG>
      </profileData>
   </profileConfig>
</tads>
```

C.2.3.3 Sample IPsec ConfigInfo for Client

This ConfigInfo file allows a IPsec client to establish IPsec connection with the IPsec server configured in section 0.

```
<?xml version="1.0" encoding="UTF-8"?>
<tads
   xmlns="urn:schemas-upnp-org:ra:tads"
   xmlns:cfg="urn:schemas-upnp-org:ra:tacfg:ipsec"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:ra:tads
   http://www.upnp.org/schemas/ra/tads-v1.xsd
   urn:schemas-upnp-org:ra:tacfg:ipsec
   http://www.upnp.org/schemas/ra/tacfg-ipsec-v1.xsd">
   <profileConfig dataStructureType="client">
      <profileInfo id="12" transportAgentName="IPsec">
```

```
         IPsec configuration
      </profileInfo>
      <profileData>
         <cfg:ipsecCFG configurationType="client">
            <cfg:policy>
               <cfg:perfectForwardSecrecy>
                  true
               </cfg:perfectForwardSecrecy>
               <cfg:replayWindowLength>10</cfg:replayWindowLength>
               <cfg:remoteIdentity>alice@home.com</cfg:remoteIdentity>
               <cfg:proposal protocol="ESP">
                  <cfg:encryptionAlgorithm keyLength="128">
                     AES_CBC
                  </cfg:encryptionAlgorithm>
                  <integrityAlgorithm>
                     HMAC_SHA1_96
                  </integrityAlgorithm>
                  <cfg:lifetime>
                     <cfg:seconds>28800</cfg:seconds>
                     <cfg:kBytes>5000000</cfg:kBytes>
                  </cfg:lifetime>
               </cfg:proposal>
            </cfg:policy>
            <cfg:ike version="IKEv2">
               <cfg:remoteAddress>129.178.89.81</cfg:remoteAddress>
               <cfg:sendNotification>true</cfg:sendNotification>
               <cfg:idType>ID_KEY_ID</cfg:idType>
               <cfg:useIPsecExpire>true</cfg:useIPsecExpire>
               <cfg:useReplayDetection>true</cfg:useReplayDetection>
               <cfg:useInternalAddress>true</cfg:useInternalAddress>
               <cfg:dpdHeartbeat>600</cfg:dpdHeartbeat>
               <cfg:natKeepalive>100</cfg:natKeepalive>
               <cfg:rekeyingThreshold>90</cfg:rekeyingThreshold>
               <cfg:proposal protocol="IKE">
                  <cfg:encryptionAlgorithm keyLength="128">
                     AES_CBC
                  </cfg:encryptionAlgorithm>
                  <cfg:integrityAlgorithm>
                     HMAC_SHA1_96
                  </cfg:integrityAlgorithm>
                  <cfg:pseudoRandomFunction>
                     HMAC_SHA1
                  </cfg:pseudoRandomFunction>
                  <cfg:groupDescription>MODP_1536</cfg:groupDescription>
                  <cfg:groupType>MODP</cfg:groupType>
                  <cfg:lifetime>
                     <cfg:seconds>28800</cfg:seconds>
                     <cfg:kBytes>5000</cfg:kBytes>
                  </cfg:lifetime>
               </cfg:proposal>
               <cfg:authenticationMethod>
                  Shared Key Message Integrity Code
               </cfg:authenticationMethod>
               <cfg:credentialID>100</cfg:credentialID>
            </cfg:ike>
         </cfg:ipsecCFG>
      </profileData>
   </profileConfig>
</tads>
```

Appendix D. Using OpenVPN as Remote Access Transport (Normative)

The purpose of this section is to describe how to use the OpenVPN protocol as the Remote Access transport mechanism in UPnP Remote Access.

D.1 OpenVPN Templates

D.1.1 OpenVPN Configuration Template

The current OpenVPN configuration template is primarily designed to be used to populate the OpenVPN configuration file that is used by OpenVPN.

```
<?xml version="1.0" encoding="UTF-8"?>
<openvpnCFG xmlns="urn:schemas-upnp-org:ra:tacfg:openvpn"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:ra:tacfg:openvpn
   http://www.upnp.org/schemas/ra/tacfg-openvpn-v1.xsd"
   configurationType="server">
   <protocol type="UDP">
      <dev>tap0</dev>
   </protocol>
   <sslTls>
      <cipher>AES-128-CBC</cipher>
      <credentialID>100</credentialID>
   </sslTls>
   <client remoteHost="vpn.mydomain.org" remotePort="1194">
      <resolveRetry seconds="-1"/>
      <devNode>MyVPNInterface</devNode>
      <httpProxy proxyIP="10.0.0.40" proxyPort="7788"/>
   </client>
   <server listeningIP="10.0.0.10" listeningPort="1194">
      <persistPool>pool.txt</persistPool>
      <bridge ip="10.1.1.10" netMask="255.255.255.0" startingIP="10.1.1.20"
endingIP="10.1.1.30"/>
      <routedIP ip="10.1.1.0" netMask="255.255.255.0"/>
      <push>
         <route ip="192.168.1.0" subnetMask="255.255.255.0"/>
         <gateway redirect="true"/>
         <dhcpOption name="DNS" data="10.66.0.4"/>
      </push>
      <clientToClient>true</clientToClient>
      <duplicateKeyPairs>0</duplicateKeyPairs>
      <maxClients>5</maxClients>
   </server>
   <options>
      <keepAlive interval="10" timeout="120"/>
      <enableCompression algorithm="lzo">true</enableCompression>
      <status update="1" filename="status.txt"/>
      <log enabled="1" append="1" filename="log.txt" verbosity="4"
silenceAfter="5"/>
   </options>
</openvpnCFG>
```

xml

REQUIRED for all XML documents. Case sensitive.

openvpnCFG

REQUIRED. Must have "urn:schemas-upnp-org:ra:tacfg:openvpn" as the value for the xmlns attribute; this references the UPnP Remote Access Working Committee RATA OpenVPN Configuration Template Schema. As long as the same xmlns is used, the data structure template MUST be backward compatible, i.e. usable by legacy implementations. Contains the following attributes and sub elements:

@configurationType
REQUIRED. xs:token. Determines the type of configuration. Possible values are "client" or "server".

protocol
REQUIRED. Determines the basic underlying transport of the OpenVPN. Contains the following sub elements and attributes:

@type
REQUIRED. xs:token. Determines if the underlying transport is TCP or UDP. Possible values are "TCP" or "UDP".

dev
REQUIRED. xs:token. Specifies the type of tunnel.
"tun" will create a routed IP tunnel,
"tap" will create an ethernet tunnel.
use "tap0" if you are ethernet bridging and have precreated a tap0 virtual interface and bridged it with your ethernet interface.

sslTls
REQUIRED. Specifies the SSL/TLS settings that are used by OpenVPN. Contains the following sub elements:

cipher
OPTIONAL. xs:token. The cryptographic cipher used. Acceptable values are:

"BF-CBC"	Blowfish (default)
"AES-128-CBC"	AES
"DES-EDE3-CBC"	Triple-DES

credentialID
REQUIRED. xs:string. Contains the unique ID of a credential stored on the RATA.

client
OPTIONAL. Specifies client specific configuration. **MUST** be specified if *@configurationType* is "client". Contains the following sub elements and attributes:

@remoteHost
REQUIRED. xs:string. Hostname of the OpenVPN server to connect to.

@remotePort
REQUIRED. xs:integer. The port number of the OpenVPN server to connect to.

resolveRetry
REQUIRED. Specifies the retry timeout for hostname resolution of the remote hostname. MUST have the following attribute:

@seconds
REQUIRED. xs:integer. The number of seconds that the OpenVPN client should continue to retry hostname resolution if it fails. A value of -1 means that the client should continue to retry for an infinite amount of time.

devNode
OPTIONAL. xs:string. Specifies the OpenVPN interface name. MUST be used on Windows.

httpProxy
OPTIONAL. Specifies the usage of an HTTP Proxy to connect to the OpenVPN server. Contains the following attributes:

@proxyIP
REQUIRED. xs:string. The IP address or the FQDN of the http proxy server.

@proxyPort
REQUIRED. xs:integer. The port number of the http proxy server.

server
OPTIONAL. Specifies server specific configuration. **MUST** be specified if *@configurationType* is "server". Contains the following sub elements and attributes:

@listeningIP
OPTIONAL. xs:string. The local IP address the server binds to.

@listeningPort
OPTIONAL. xs:integer. The port number that the server binds to.

persistPool

OPTIONAL. xs:string. Reconnecting clients can be assigned the same virtual IP address from the pool that was previously assigned. The value specified is the filename used to store this persistent information.

bridge

OPTIONAL. Configure server mode for ethernet bridging. The configuration must set aside an IP range in the specified subnet to allocate to connecting clients. **MUST** be specified if the server is configured for ethernet bridging. MUST NOT be used in conjunction with the routedIP element. Contains the following attributes:

@ip

REQUIRED. xs:string. IP Address of the bridge interface.

@netMask

REQUIRED. xs:string. Subnet mask of the bridge interface.

@startingIP

REQUIRED. xs:string. Starting IP Address to allocate to the pool.

@endingIP

REQUIRED. xs:string. Ending IP Address to allocate to the pool.

routedIP

OPTIONAL. Configure server mode for routed IP, to supply a VPN subnet for OpenVPN to draw client addresses from. The server will take the first IP address for itself, the rest will be made available to clients. Each client will be able to reach the server on that self-allocated IP address. MUST NOT be used in conjunction with the bridge element. **MUST** be specified if the server is configured for routed IP. Contains the following attributes:

@ip

REQUIRED. xs:string. Subnet IP Address of the routed VPN tunnel.

@netMask

REQUIRED. xs:string. Subnet mask of the subnet address.

push

OPTIONAL. Allows the server to push various options to the client. Contains one of more of the following sub elements.

route

OPTIONAL. Allows the server to push routes to the client to allow it to reach other private subnets behind the server. Contains the following attributes:

@ip

REQUIRED. xs:string.The IP address portion of the route to be pushed.

@subnetMask

REQUIRED. xs:string. The netmask portion of the route to be pushed.

gateway

OPTIONAL. Allows the server to specify the default gateway to the client. Contains the following attribute:

@redirect

REQUIRED. xs:boolean. Specifies if the default gateway should be redirected to the OpenVPN server.

dhcpOption

OPTIONAL. Allows the server to push DHCP options to the client. Contains the following attributes:

@name

REQUIRED. xs:string. The DHCP option name to push.

@data

REQUIRED. xs:string. The value for the above specified DHCP option.

clientToClient

OPTIONAL. xs:boolean. Specifies if the connected clients are visible to each other.

duplicateKeyPairs

OPTIONAL. xs:boolean. Specifies if clients are required to login with unique certificates. OpenVPN recommends that duplicate certificates only be allowed in debug environments.

maxClients

OPTIONAL. xs:integer. Specifies the maximum number of concurrent clients to allow.

options

OPTIONAL. Specifies additional options. Contains the following sub elements:

keepAlive
OPTIONAL. Specifies the keep alive parameters for the client/server endpoint. Contains the following attributes:
@interval
REQUIRED. xs:integer. The frequency interval specified in seconds. A Keep Alive will be sent every *x* seconds, where *x* is the specified value.
@timeout
REQUIRED. xs:integer. The number of seconds that must elapse without receiving a keep alive before the endpoint considers the session expired.

enableCompression
OPTIONAL. xs:boolean. Specifies if the OpenVPN link utilizes compression. Contains the following attribute:

@algorithm
OPTIONAL. xs:token. Currently only "lzo" is supported. MUST be present if compression is used.

status
OPTIONAL. Specifies a status file showing current connections, truncated and rewritten every minute. Contains the following attributes:

@update
REQUIRED. xs:boolean. Specifies if status file updates are enabled.
@filename
REQUIRED. xs:string. Specifies the filename of the file to write the updates to.

log
OPTIONAL. xs:string. By default, log messages will go to the syslog (or on Windows, if running as a service, they will go to the "\Program Files\OpenVPN\log" directory). This allows one to override this default behavior. Contains the following attributes:

@enabled
REQUIRED. xs:boolean. Specifies if the default behavior will be over-ridden.
@append
REQUIRED. xs:boolean. Specifies if the log file will be appended or over-written.
@filename
REQUIRED. xs:string. The filename to write the log to.
@verbosity
OPTIONAL. xs:integer. The log file verboseness, from 0-9.
0 is silent, except for fatal errors
4 is reasonable for general usage
5 and 6 can help to debug connection problems
9 is extremely verbose
@silenceAfter
OPTIONAL. xs:integer. Silence repeating messages. At most *n* sequential messages of the same message category will be output to the log, where *n* is the specified value.

D.2 Sample OpenVPN configuration

D.2.1 Sample configuration for Server

```
<?xml version="1.0" encoding="UTF-8"?>
<tads
   xmlns="urn:schemas-upnp-org:ra:tads"
   xmlns:cfg="urn:schemas-upnp-org:ra:tacfg:openvpn"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:ra:tads
   http://www.upnp.org/schemas/ra/tads-v1.xsd
   urn:schemas-upnp-org:ra:tacfg:openvpn
   http://www.upnp.org/schemas/ra/tacfg-openvpn-v1.xsd">
   <profileConfig dataStructureType="server">
      <profileInfo id="12" transportAgentName="OpenVPN">
         OpenVPN configuration
      </profileInfo>
      <profileData>
         <cfg:openvpnCFG configurationType="server">
            <cfg:protocol type="UDP">
               <cfg:dev>tap0</cfg:dev>
            </cfg:protocol>
            <cfg:sslTls>
               <cfg:cipher>AES-128-CBC</cfg:cipher>
               <cfg:credentialID>100</cfg:credentialID>
            </cfg:sslTls>
            <cfg:server listeningPort="1194">
               <cfg:persistPool>pool.txt</cfg:persistPool>
               <cfg:bridge
                  ip="10.1.1.10"
                  netMask="255.255.255.0"
                  startingIP="10.1.1.20"
                  endingIP="10.1.1.30"/>
               <cfg:push>
                  <cfg:gateway redirect="true"/>
               </cfg:push>
               <cfg:clientToClient>true</cfg:clientToClient>
            </cfg:server>
            <cfg:options>
               <cfg:keepAlive interval="10" timeout="120"/>
               <cfg:enableCompression algorithm="lzo">
                  true
               </cfg:enableCompression>
            </cfg:options>
         </cfg:openvpnCFG>
      </profileData>
   </profileConfig>
</tads>
```

D.2.2 Sample configuration for Client

```
<?xml version="1.0" encoding="UTF-8"?>
<tads
   xmlns="urn:schemas-upnp-org:ra:tads"
   xmlns:cfg="urn:schemas-upnp-org:ra:tacfg:openvpn"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:schemas-upnp-org:ra:tads
   http://www.upnp.org/schemas/ra/tads-v1.xsd
   urn:schemas-upnp-org:ra:tacfg:ipsec
   http://www.upnp.org/schemas/ra/tacfg-openvpn-v1.xsd">
   <profileConfig dataStructureType="server">
      <profileInfo id="12" transportAgentName="OpenVPN">
         OpenVPN configuration
      </profileInfo>
      <profileData>
         <cfg:openvpnCFG configurationType="client">
            <cfg:protocol type="UDP">
               <cfg:dev>tap</cfg:dev>
            </cfg:protocol>
            <cfg:sslTls>
               <cfg:cipher>AES-128-CBC</cfg:cipher>
               <cfg:credentialID>100</cfg:credentialID>
            </cfg:sslTls>
            <cfg:client remoteHost="vpn.mydomain.org" remotePort="1194">
```

```
                <cfg:resolveRetry seconds="-1"/>
                <cfg:devNode>MyVPNInterface</cfg:devNode>
             </cfg:client>
             <cfg:options>
                <cfg:keepAlive interval="10" timeout="120"/>
                <cfg:enableCompression algorithm="lzo">
                   true
                </cfg:enableCompression>
             </cfg:options>
          </cfg:openvpnCFG>
      </profileData>
   </profileConfig>
</tads>
```