InboundConnectionConfig: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
Mark Baugher	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

Authors	Company
Jeffrey Kang	Philips
Wouter van der Beek	Philips
Shrikant Kanaparti	Samsung
Se-Hee Han	Samsung
Mahfuzur Rahman (Co-chair)	Samsung
Sanjeev Verma	Samsung

UPnP Forum in no way guarantees the accuracy or completeness of this author list and in no way implies any rights for or support from those members listed. This list is not the specifications' contributor list that is kept on the UPnP Forum's website.

Contents

List of Tables

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 *InboundConnectionConfig* service.

1.1 Introduction

The *InboundConnectionConfig* service is a UPnP service that allows control points to configure the parameters that will enable the service to test if the host device is reachable from the internet. InboundConnectionConfig uses Dynamic DNS to manage at least one public address for home-network services; it uses STUN to ensure that any intermediate NAT device is navigatable, i.e. it is a full-cone NAT. Thus, this service provides control points with the following functionality:

- Configure the dynamic DNS client co-located with the service,
- Configure the STUN client co-located with the service,
- Check if the device hosting the service is reachable from the internet.

This service does not address:

- Configuration of relay services in the public network, e.g. TURN.

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, UPnP Forum, June 13, 2000.
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.

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

[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://www.upnp.org/specs/ra/UPnP-ra-RAServer-v1-Device.pdf.

[RFC 1034] – IETF RFC 1034, DOMAIN NAMES - CONCEPTS AND FACILITIES, P. Mockapetris, November 1987
Available at: http://www.ietf.org/rfc/rfc1034.txt

[RFC 1035] – IETF RFC 1035, DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION, P. Mockapetris, November 1987
Available at: http://www.ietf.org/rfc/rfc1035.txt

[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

[RFC 2136] – IETF RFC 2136, Dynamic Updates in the Domain Name System (DNS UPDATE), P. Vixie (Editor), April 1997
Available at: http://www.ietf.org/rfc/rfc2136.txt

[RFC 3489] – IETF RFC 3489, STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs), J. Rosenberg, et. Al., March 2003
Available at: http://www.ietf.org/rfc/rfc3489.txt

[RFC 3986] – IETF RFC 3986, Uniform Resource Identifier (URI): Generic Syntax, Tim Berners-Lee, et. Al., January 2005.
Available at: http://www.ietf.org/rfc/rfc3986.txt

[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 specificatio:

[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-v1-20090930pdf.
Latest version available at: http://www.upnp.org/specs/ra/UPnP-ra-RAArchitecture-v1.pdf.

[TURN] – IETF Internet Draft, Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN), J. Rosenberg, July 8, 2007
Available at: http://www.ietf.org/internet-drafts/draft-ietf-behave-turn-04.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:*InboundConnectionConfig:1*

InboundConnectionConfig 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
DNS	Domain Name System
FQDN	Fully Qualified Domain Name
NAT	Network Address Translation
RAS	Remote Access Server
STUN	Simple Traversal of UDP Through NATs
TURN	Traversal Using Relays around NAT

2.3 *InboundConnectionConfig* Service Architecture

This service provides the features that enable the end user to determine if a Remote Access Server can be deployed in the home network by checking if the Remote Access Server is reachable from the Internet.

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
DynamicDNSSupportedProtocols	*R*	**string**	CSV(**string**) See Section 2.4.2	
DynamicDNSConfigInfo	*R*	**string**	See Section 2.4.3	
STUNServerAddress	*R*	**string**	See Section 2.4.4	
NetworkTopologyInfo	*R*	**string**	See Section 2.4.5	

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

2.4.2 *DynamicDNSSupportedProtocols*

This state variable contains the list of protocols supported by the Dynamic DNS client. If an Internet service provider maintains the DNS to always reflect the dynamically-allocated or statically-assigned public IP address, then this is signaled by the name "RFC2136", which indicates Dynamic DNS configuration is not required. Otherwise, the name of the protocol indicates how to configure Dynamic DNS with a commercial provider.

2.4.3 *DynamcDNSConfigInfo*

This state variable contains the configuration information for the Dynamic DNS client. The structure of the *DynamicDNSConfigInfo* state variable is a ICCDS XML Document:

- `<dynamicDNSConfig>` is the root element.
- See the ICCS schema [ICCDS-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 *DynamicDNSConfigInfo* 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 *STUNServerAddress*

This state variable contains IP address or the Fully Qualified Domain Name of the STUN server.

2.4.5 *NetworkTopologyInfo*

This state variable contains the information about the topology of the network located between the device hosting the service and the public internet. It includes the information collected by the STUN client about the NAT devices and types or from the IGD if the service is not co-located with the residential gateway.

The structure of the *NetworkTopologyInfo* state variable is a ICCDS XML Document:

- `<networkTopologyInfo>` is the root element.
- See the ICCS schema [ICCDS-XSD] for more details on the structure. The available properties and their names are described in Appendix A.2.

Note that since the value of *NetworkTopologyInfo* 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.

2.5 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]
DynamicDNSSupportedProtocols	*NO*	*NO*			
DynamicDNSConfigInfo	*NO*	*NO*			
STUNServerAddress	*NO*	*NO*			
NetworkTopologyInfo	*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]
GetDynamicDNSSupportedProtocols()	*R*
SetDynamicDNSConfigInfo()	*R*
SetSTUNServerAddress()	*R*
GetNetworkTopologyInfo()	*R*

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

2.6.1 *GetDynamicDNSSupportedProtocols()*

This action is used to get the list of the protocols (e.g. services) supported by Dynamic DNS client.

2.6.1.1 Arguments

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

Argument	Direction	relatedStateVariable
DynamicDNSSupportedProtocols	*OUT*	*DynamicDNSSupportedProtocols*

2.6.1.1.1 *DynamicDNSSupportedProtocols*

This argument indicates the supported dynamic DNS update protocols.

2.6.1.2 Dependency on State

None.

2.6.1.3 Effect on State

None.

2.6.1.4 Control Point Requirements

Control points MUST select the preferred dynamic DNS protocol from the list and use the same value for `<dynDNSProtocol>` as part of the *NewDynamicDNSConfigInfo* *argument in the SetDynamicDNSConfigInfo() action*.

2.6.1.5 Errors

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

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 *SetDynamicDNSConfigInfo()*

This action is used to deliver the configuration parameters for the DynamicDNS client. It should be used for configuring the the commercial Dynamic DNS clients. No configuration needs to be done if the device supports the mechanism defined by RFC2136 [RFC 2136].

2.6.2.1 Arguments

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

Argument	Direction	relatedStateVariable
NewDynamicDNSConfigInfo	*IN*	*DynamicDNSConfigInfo*

2.6.2.1.1 *NewDynamicDNSConfigInfo*

This argument contains the configurations that allows the client to update the DNS records.

2.6.2.2 Dependency on State

None.

2.6.2.3 Effect on State

The effect of *SetDynamicDNSConfigInfo()* action is to update the *DynamicDNSConfigInfo* state variable with the value provided as *NewDynamicDNSConfigInfo*.

2.6.2.4 Control Point Requirements

Control points MUST use for the `<dynDNSProtocol>` the value of the preferred protocol selected from the *DynamicDNSSupportedProtocols*.

The control point may already have all the information required to create the *dynamicDNSConfig* XML structure or it can obtain it by contacting the dynamic DNS service provider. Alternatively, the service provider can provide an already made *dynamicDNSConfig* XML structure at the end of the registration process or following a configuration request.

2.6.2.5 Errors

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

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

ErrorCode	errorDescription	Description
600-699	TBD	See UPnP Device Architecture section on Control.
705	Unsupported DDNS configuration	*SetDynamicDNSConfigInfo()* failed because the supplied DDNS configuration is not supported.

2.6.3 *SetSTUNServerAddress()*

This action is used to configure the IP address or the fully qualified domain name of the STUN server that will be queried by the STUN client co-located with the service.

2.6.3.1 Arguments

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

Argument	Direction	relatedStateVariable
NewSTUNServerAddress	*IN*	*STUNServerAddress*

2.6.3.1.1 *NewSTUNServerAddress*

This argument contains the IP address or the fully qualified domain name of the STUN server.

2.6.3.2 Dependency on State

None.

2.6.3.3 Effect on State

The effect of *SetSTUNServerAddress()* action is to update the *STUNServerAddress* state variable with the value provided as *NewSTUNServerAddress*.

2.6.3.4 Control Point Requirements

None.

2.6.3.5 Errors

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

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 *GetNetworkTopologyInfo()*

This action returns the information about the network topology existent between the device and the public internet.

2.6.4.1 Arguments

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

Argument	Direction	relatedStateVariable
CurrentNetworkTopologyInfo	*OUT*	*NetworkTopologyInfo*

2.6.4.1.1 *CurrentNetworkTopologyInfo*

This argument indicates if the network topology is supported as well as giving additional information that will help the troubleshooting process.

2.6.4.2 Dependency on State

None.

2.6.4.3 Effect on State

None.

2.6.4.4 Control Point Requirements

Although the InboundConnectionConfig service reports if the network topology between the device hosting the service and the public internet is supported, control points can use the reported information to provide additional guidance to users about the nature of the problems and even suggest solutions.

Under the UPnP Remote Access 1.0 architecture, the values returned by the NetworkTopologyInfo have the meaning as listed in Table 2-12.

Table 2-12: NetworkTopologyInfo meaning in UPnP Remote Access 1.0

Supported Topology	NAT Type	Dynamic DNS Config status
True	none	configured
	fullCone	configured
False	restricted	n/a
	portRestricted	n/a
	symmetric	n/a

Before invoking the *GetNetworkTopologyInfo()* action, the *GetDynamicDNSSupportedProtocols()*, *SetDynamicDNSConfigInfo()* and *SetSTUNServerAddress()* actions MUST have been invoked first. Otherwise, this action will fail, and the corresponding error code will be returned indicating the reason of failure. The control point SHOULD then invoke the relevant actions again before re-invoking the *GetNetworkTopologyInfo()* action for a successful completion of the action.

2.6.4.5 Errors

Table 2-13: Error Codes for *GetNetworkTopologyInfo()*

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

ErrorCode	errorDescription	Description
600-699	TBD	See UPnP Device Architecture section on Control.
701	STUN server address not configured	*GetNetworkTopologyInfo()* failed because the STUN server address is not yet configured.
702	DDNS client not configured	*GetNetworkTopologyInfo()* failed because the Dynamic DNS client is not yet configured.
703	STUN server not reachable	*GetNetworkTopologyInfo()* failed because the STUN server is not reachable.
704	Invalid DDNS configuration	*GetNetworkTopologyInfo()* failed because the Dynamic DNS client was not configured correctly.

2.6.5 Relationships Between Actions

All actions in this service are inter-related, and the invocation sequence of these actions MUST be respected. See sections of the respective actions for details.

2.6.6 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 2-14: 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	STUN server address not configured	*GetNetworkTopologyInfo()* failed because the STUN server address is not yet configured.
702	DDNS client not configured	*GetNetworkTopologyInfo()* failed because the Dynamic DNS client is not yet configured.
703	STUN server not reachable	*GetNetworkTopologyInfo()* failed because the STUN server is not reachable.
704	Invalid DDNS configuration	*GetNetworkTopologyInfo()* failed because the Dynamic DNS client was not configured correctly.
705	Unsupported DDNS configuration	*SetDynamicDNSConfigInfo()* failed because the supplied DDNS configuration is not supported.

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

From the Control Point perspective, configuring the home network for accepting inbound connections is a three step process. The first step is to configure the STUN client co-located with the service via the *SetSTUNServerAddress()* action. The second step is to configure the dynamic DNS client co-located with the service via the *SetDynamicDNSConfigInfo()* action. Finally, the Control Point checks if the home network is reachable from internet by invoking the *GetNetworkTopology()* action.

If these three steps are succesfull, the Control Point might use the IGDv1 [IGD] functionality of the residential gateway in order to create the appropriate port mappings so that inbound connections are forwarded to the corresponding device in the home network.

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>GetDynamicDNSSupportedProtocols</name>
         <argumentList>
            <argument>
               <name>DynamicDNSSupportedProtocols</name>
               <direction>out</direction>
               <retval/>
               <relatedStateVariable>
                  DynamicDNSSupportedProtocols
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>SetDynamicDNSConfigInfo</name>
         <argumentList>
            <argument>
               <name>NewDynamicDNSConfigInfo</name>
               <direction>in</direction>
               <relatedStateVariable>
                  DynamicDNSConfigInfo
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>SetSTUNServerAddress</name>
         <argumentList>
            <argument>
               <name>NewSTUNServerAddress</name>
               <direction>in</direction>
               <relatedStateVariable>
                  STUNServerAddress
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>GetNetworkTopologyInfo</name>
         <argumentList>
            <argument>
```

```
               <name>CurrentNetworkTopologyInfo</name>
               <direction>in</direction>
               <relatedStateVariable>
                 NetworkTopologyInfo
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

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

   </actionList>

   <serviceStateTable>

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

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

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

      <stateVariable sendEvents="no">
         <name>NetworkTopologyInfo</name>
         <dataType>string</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. InboundConnectionConfig Data Structures

A.1 DynamicDNSConfig Template

The following shows the generalized layout of a DynamicDNSConfig Template. More elements and/or attributes MAY be added in future versions of DynamicDNSConfig 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"?>
<iccds xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="urn:schemas-upnp-org:ra:iccds"
   xsi:schemaLocation="
   urn:schemas-upnp-org:ra:iccds
   http://www.upnp.org/schemas/ra/iccds-v1.xsd">
   <dynamicDNSConfiguration>
      <dynDNSProtocol>dynamic DNS protocol name</dynDNSProtocol>
      <updateConfiguration>
         <server>ip address or FQDN of dynamic DNS server</server>
         <username>username</username>
         <password>password</password>
         <domain>domain that will be updated</domain>
      </updateConfiguration>
      <policy>
         <slaURI>URI to SLA of the dynamic DNS provider</slaURI>
         <forceUpdate>time interval between force updates</forceUpdate>
      </policy>
   </dynamicDNSConfiguration>
</iccds>
```

xml
REQUIRED for all XML documents. Case sensitive.

iccds
REQUIRED. Must have “urn:schemas-upnp-org:ra:iccds” as the value for the xmlns attribute; this references the UPnP Remote Access Working Committee ICC 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.

dynamicDNSConfiguration
REQUIRED. Contains the information required to configure a dynamicDNS client. MUST contain the following sub-elements:

dynDNSProtocol
REQUIRED. xs:string. Contains the identification name of a dynamic DNS update protocol.

updateConfiguration
REQUIRED. Contains the dynamic DNS client configuration parameters. MUST Contain the following sub-elements:

server
REQUIRED. xs:string. Contains the IP address or the FQDN of the dynamic DNS server.

username
REQUIRED. xs:string. Contains the username for the dynamic DNS service account.

password
REQUIRED. xs:string. Contains the password for the dynamic DNS service account.

domain
REQUIRED. xs:string. Contains the domain name that will be updated.

policy
REQUIRED. Contains policy information for the dynamic DNS service.

slaURI
REQUIRED. xs:anyURI. Contains the URI to the service level agreement of the dynamic DNS service.

forceUpdate
REQUIRED. xs:duration. Contains the time interval between forced updates.

A.2 NetworkTopologyInfo Template

The following shows the generalized layout of a NetworkTopologyInfo Template. More elements and/or attributes MAY be added in future versions of NetworkTopologyInfo 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"?>
<iccds xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="urn:schemas-upnp-org:ra:iccds"
   xsi:schemaLocation="
   urn:schemas-upnp-org:ra:iccds
   http://www.upnp.org/schemas/ra/iccds-v1.xsd">
   <networkTopologyInfo>
      <supportedTopology>
         true|false
      </supportedTopology>
      <hostName nameSystem="DNS" lastUpdate="" v4="true" v6="false">
         myserver.myisp.net
      </hostName>
      <natType>NAT type detected by STUN client</natType>
   </networkTopologyInfo>
</iccds>
```

xml
REQUIRED for all XML documents. Case sensitive.

iccds
REQUIRED. Must have "urn:schemas-upnp-org:ra:iccds" as the value for the xmlns attribute; this references the UPnP Remote Access Working Committee ICC 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.

networkTopologyInfo
REQUIRED. Contains the network topology info on the WAN side of the device. MUST contain the following sub elements:

supportedTopology
REQUIRED. xs:boolean. Indicates if the topology is supported by the device. The value is set based on the device's determination of the network topology (according to Table 2-12).

hostName
REQUIRED. xs:string. Contains the server name. Contains the following attributes:

@nameSystem

REQUIRED. xs:token. Indicates the name system where the hostName can be resolved. Value supported by UPnP Remote Access Architecture v1.0 is “DNS”.

@lastUpdate

OPTIONAL. xs:dateTime. Indicates the time when the last successful dynamic DNS update happened.

@v4

OPTIONAL. xs:boolean. Indicates if the DNS resolver contains A record, e.g. public IPv4 address.

@v6

OPTIONAL. xs:boolean. Indicates if the DNS resolver contains AAAA record, e.g. public IPv6 address.

natType

REQUIRED. xs:token. Contains the NAT type detected by the STUN Client. See Table 2-12 for allowed values.