INTERNATIONAL STANDARD

ISO/IEC 29341-26-12

First edition
2017-09

Information technology — UPnP Device Architecture —

Part 26-12: Telephony device control protocol — Level 2 — Messaging service

Technologies de l'information — Architecture de dispositif UPnP —

Partie 26-12: Protocole de contrôle de dispositif de téléphonie — Niveau 2 — Service de messagerie

Reference number
ISO/IEC 29341-26-12:2017(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

CONTENTS

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see http://www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of Standard, the meaning of the ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword – Supplementary information

ISO/IEC 29341-26-12 was prepared by UPnP Forum and adopted, under the PAS procedure, by joint technical committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

The list of all currently available parts of ISO/IEC 29341 series, under the general title *Information technology — UPnP Device Architecture*, can be found on the ISO web site.

Introduction

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

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

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

Information may be obtained from:

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

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

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

Information may be obtained from:

Microsoft Corporation
One Microsoft Way
USA – Redmond WA 98052

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

Information may be obtained from:

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

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

Information may be obtained from:

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

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

Information may be obtained from:

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

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

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

Information may be obtained from:

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

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

Information may be obtained from:

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

Huawei Technologies Co., Ltd. has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Huawei Technologies Co., Ltd.
Administration Building, Bantian Longgang District
Shenzhen – China 518129

Qualcomm Incorporated has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Qualcomm Incorporated
5775 Morehouse Drive
San Diego, CA – USA 92121

Telecom Italia S.p.A.has informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Telecom Italia S.p.A.
Via Reiss Romoli, 274
Turin - Italy 10148

Cisco Systems informed IEC and ISO that it has patent applications or granted patents.

Information may be obtained from:

Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA – USA 95134

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

Original UPnP Document

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

UPnP Document Title	ISO/IEC 29341 Part
UPnP Device Architecture 1.0	ISO/IEC 29341-1:2008
UPnP Device Architecture Version 1.0	ISO/IEC 29341-1:2011
UPnP Device Architecture 1.1	ISO/IEC 29341-1-1:2011
UPnP Device Architecture 2.0	ISO/IEC 29341-1-2
UPnP Basic:1 Device	ISO/IEC 29341-2
UPnP AV Architecture:1	ISO/IEC 29341-3-1:2008
UPnP AV Architecture:1	ISO/IEC 29341-3-1:2011
UPnP AVTransport:1 Service	ISO/IEC 29341-3-10
UPnP ConnectionManager:1 Service	ISO/IEC 29341-3-11
UPnP ContentDirectory:1 Service	ISO/IEC 29341-3-12
UPnP RenderingControl:1 Service	ISO/IEC 29341-3-13
UPnP MediaRenderer:1 Device	ISO/IEC 29341-3-2
UPnP MediaRenderer:2 Device	ISO/IEC 29341-3-2:2011
UPnP MediaServer:1 Device	ISO/IEC 29341-3-3
UPnP AVTransport:2 Service	ISO/IEC 29341-4-10:2008
UPnP AVTransport:2 Service	ISO/IEC 29341-4-10:2011
UPnP ConnectionManager:2 Service	ISO/IEC 29341-4-11:2008
UPnP ConnectionManager:2 Service	ISO/IEC 29341-4-11:2011
UPnP ContentDirectory:2 Service	ISO/IEC 29341-4-12
UPnP RenderingControl:2 Service	ISO/IEC 29341-4-13:2008
UPnP RenderingControl:2 Service	ISO/IEC 29341-4-13:2011
UPnP ScheduledRecording:1	ISO/IEC 29341-4-14
UPnP ScheduledRecording:2	ISO/IEC 29341-4-14:2011
UPnP MediaRenderer:2 Device	ISO/IEC 29341-4-2
UPnP MediaServer:2 Device	ISO/IEC 29341-4-3
UPnP AV Datastructure Template:1	ISO/IEC 29341-4-4:2008
UPnP AV Datastructure Template:1	ISO/IEC 29341-4-4:2011
UPnP DigitalSecurityCamera:1 Device	ISO/IEC 29341-5-1
UPnP DigitalSecurityCameraMotionImage:1 Service	ISO/IEC 29341-5-10
UPnP DigitalSecurityCameraSettings:1 Service	ISO/IEC 29341-5-11
UPnP DigitalSecurityCameraStillImage:1 Service	ISO/IEC 29341-5-12
UPnP HVAC_System:1 Device	ISO/IEC 29341-6-1
UPnP ControlValve:1 Service	ISO/IEC 29341-6-10
UPnP HVAC_FanOperatingMode:1 Service	ISO/IEC 29341-6-11
UPnP FanSpeed:1 Service	ISO/IEC 29341-6-12
UPnP HouseStatus:1 Service	ISO/IEC 29341-6-13
UPnP HVAC_SetpointSchedule:1 Service	ISO/IEC 29341-6-14
UPnP TemperatureSensor:1 Service	ISO/IEC 29341-6-15
UPnP TemperatureSetpoint:1 Service	ISO/IEC 29341-6-16

UPnP HVAC_UserOperatingMode:1 Service	ISO/IEC 29341-6-17
UPnP HVAC_ZoneThermostat:1 Device	ISO/IEC 29341-6-2
UPnP BinaryLight:1 Device	ISO/IEC 29341-7-1
UPnP Dimming:1 Service	ISO/IEC 29341-7-10
UPnP SwitchPower:1 Service	ISO/IEC 29341-7-11
UPnP DimmableLight:1 Device	ISO/IEC 29341-7-2
UPnP InternetGatewayDevice:1 Device	ISO/IEC 29341-8-1
UPnP LANHostConfigManagement:1 Service	ISO/IEC 29341-8-10
UPnP Layer3Forwarding:1 Service	ISO/IEC 29341-8-11
UPnP LinkAuthentication:1 Service	ISO/IEC 29341-8-12
UPnP RadiusClient:1 Service	ISO/IEC 29341-8-13
UPnP WANCableLinkConfig:1 Service	ISO/IEC 29341-8-14
UPnP WANCommonInterfaceConfig:1 Service	ISO/IEC 29341-8-15
UPnP WANDSLLinkConfig:1 Service	ISO/IEC 29341-8-16
UPnP WANEthernetLinkConfig:1 Service	ISO/IEC 29341-8-17
UPnP WANIPConnection:1 Service	ISO/IEC 29341-8-18
UPnP WANPOTSLinkConfig:1 Service	ISO/IEC 29341-8-19
UPnP LANDevice:1 Device	ISO/IEC 29341-8-2
UPnP WANPPPConnection:1 Service	ISO/IEC 29341-8-20
UPnP WLANConfiguration:1 Service	ISO/IEC 29341-8-21
UPnP WANDevice:1 Device	ISO/IEC 29341-8-3
UPnP WANConnectionDevice:1 Device	ISO/IEC 29341-8-4
UPnP WLANAccessPointDevice:1 Device	ISO/IEC 29341-8-5
UPnP Printer:1 Device	ISO/IEC 29341-9-1
UPnP ExternalActivity:1 Service	ISO/IEC 29341-9-10
UPnP Feeder:1.0 Service	ISO/IEC 29341-9-11
UPnP PrintBasic:1 Service	ISO/IEC 29341-9-12
UPnP Scan:1 Service	ISO/IEC 29341-9-13
UPnP Scanner:1.0 Device	ISO/IEC 29341-9-2
UPnP QoS Architecture:1.0	ISO/IEC 29341-10-1
UPnP QosDevice:1 Service	ISO/IEC 29341-10-10
UPnP QosManager:1 Service	ISO/IEC 29341-10-11
UPnP QosPolicyHolder:1 Service	ISO/IEC 29341-10-12
UPnP QoS Architecture:2	ISO/IEC 29341-11-1
UPnP QosDevice:2 Service	ISO/IEC 29341-11-10
UPnP QosManager:2 Service	ISO/IEC 29341-11-11
UPnP QosPolicyHolder:2 Service	ISO/IEC 29341-11-12
UPnP QOS v2 Schema Files	ISO/IEC 29341-11-2
UPnP RemoteUIClientDevice:1 Device	ISO/IEC 29341-12-1
UPnP RemoteUIClient:1 Service	ISO/IEC 29341-12-10
UPnP RemoteUIServer:1 Service	ISO/IEC 29341-12-11
UPnP RemoteUIServerDevice:1 Device	ISO/IEC 29341-12-2
UPnP DeviceSecurity:1 Service	ISO/IEC 29341-13-10
UPnP SecurityConsole:1 Service	ISO/IEC 29341-13-11
UPnP ContentDirectory:3 Service	ISO/IEC 29341-14-12:2011
UPnP MediaServer:3 Device	ISO/IEC 29341-14-3:2011

UPnP ContentSync:1	ISO/IEC 29341-15-10:2011
UPnP Low Power Architecture:1	ISO/IEC 29341-16-1:2011
UPnP LowPowerProxy:1 Service	ISO/IEC 29341-16-10:2011
UPnP LowPowerDevice:1 Service	ISO/IEC 29341-16-11:2011
UPnP QoS Architecture:3	ISO/IEC 29341-17-1:2011
UPnP QosDevice:3 Service	ISO/IEC 29341-17-10:2011
UPnP QosManager:3 Service	ISO/IEC 29341-17-11:2011
UPnP QosPolicyHolder:3 Service	ISO/IEC 29341-17-12:2011
UPnP QosDevice:3 Addendum	ISO/IEC 29341-17-13:2011
UPnP RemoteAccessArchitecture:1	ISO/IEC 29341-18-1:2011
UPnP InboundConnectionConfig:1 Service	ISO/IEC 29341-18-10:2011
UPnP RADAConfig:1 Service	ISO/IEC 29341-18-11:2011
UPnP RADASync:1 Service	ISO/IEC 29341-18-12:2011
UPnP RATAConfig:1 Service	ISO/IEC 29341-18-13:2011
UPnP RAClient:1 Device	ISO/IEC 29341-18-2:2011
UPnP RAServer:1 Device	ISO/IEC 29341-18-3:2011
UPnP RADiscoveryAgent:1 Device	ISO/IEC 29341-18-4:2011
UPnP SolarProtectionBlind:1 Device	ISO/IEC 29341-19-1:2011
UPnP TwoWayMotionMotor:1 Service	ISO/IEC 29341-19-10:2011
UPnP AV Architecture:2	ISO/IEC 29341-20-1
UPnP AVTransport:3 Service	ISO/IEC 29341-20-10
UPnP ConnectionManager:3 Service	ISO/IEC 29341-20-11
UPnP ContentDirectory:4 Device	ISO/IEC 29341-20-12
UPnP RenderingControl:3 Service	ISO/IEC 29341-20-13
UPnP ScheduledRecording:2 Service	ISO/IEC 29341-20-14
UPnP MediaRenderer:3 Service	ISO/IEC 29341-20-2
UPnP MediaServer:4 Device	ISO/IEC 29341-20-3
UPnP AV Datastructure Template:1	ISO/IEC 29341-20-4
UPnP InternetGatewayDevice:2 Device	ISO/IEC 29341-24-1
UPnP WANIPConnection:2 Service	ISO/IEC 29341-24-10
UPnP WANIPv6FirewallControl:1 Service	ISO/IEC 29341-24-11
UPnP WANConnectionDevice:2 Service	ISO/IEC 29341-24-2
UPnP WANDevice:2 Device	ISO/IEC 29341-24-3
UPnP Telephony Architecture:2	ISO/IEC 29341-26-1
UPnP CallManagement:2 Service	ISO/IEC 29341-26-10
UPnP MediaManagement:2 Service	ISO/IEC 29341-26-11
UPnP Messaging:2 Service	ISO/IEC 29341-26-12
UPnP PhoneManagement:2 Service	ISO/IEC 29341-26-13
UPnP AddressBook:1 Service	ISO/IEC 29341-26-14
UPnP Calendar:1 Service	ISO/IEC 29341-26-15
UPnP Presense:1 Service	ISO/IEC 29341-26-16
UPnP TelephonyClient:2 Device	ISO/IEC 29341-26-2
UPnP TelephonyServer:2 Device	ISO/IEC 29341-26-3
UPnP Friendly Info Update:1 Service	ISO/IEC 29341-27-1
UPnP MultiScreen MultiScreen Architecture:1	ISO/IEC 29341-28-1
UPnP MultiScreen Application Management:1 Service	ISO/IEC 29341-28-10

UPnP MultiScreen Screen:1 Device	ISO/IEC 29341-28-2
UPnP MultiScreen Application Management:2 Service	ISO/IEC 29341-29-10
UPnP MultiScreen Screen:2 Device	ISO/IEC 29341-29-2
UPnP IoT Management and Control Architecture Overview:1	ISO/IEC 29341-30-1
UPnP DataStore:1 Service	ISO/IEC 29341-30-10
UPnP IoT Management and Control Data Model:1 Service	ISO/IEC 29341-30-11
UPnP IoT Management and Control Transport Generic:1 Service	ISO/IEC 29341-30-12
UPnP IoT Management and Control:1 Device	ISO/IEC 29341-30-2
UPnP Energy Management:1 Service	ISO/IEC 29341-31-1

1 Scope

This service definition is compliant with [1]. It defines a service type referred to herein as *Messaging* service.

The *Messaging* service is a UPnP service that allows control points to use the messaging features provided by a Telephony Server (TS) or a Telephony Client (TC).

It provides in the UPnP network the overall set of messaging capabilities of a phone (e.g., smartphone, IP phone, VoIP gateway, etc.), as the role of a TS. Additionally, or as an alternative, a device with the role of a TC can provide the *Messaging* service via UPnP Telephony.

This service provides the control points with the following functionalities:

- Page Mode Messaging (e.g., e-Mail, SMS, MMS, etc.): notification of incoming messages, reading messages, sending a message, deleting a message.
- Session Mode Messaging (e.g., Instant Messaging, SMS, etc.): notification of incoming messages, retrieving and sending the messages within a session, creating, modifying or closing a messaging session.
- File transfer session, to transfer the files in near real time.

This service does not provide the following functionalities:

- Connection to a remote server for sending or receiving the messages (e.g., WAN side);
- Rendering of message notifications and messages to the end user;
- Configuring the *Messaging* service with account for connecting to remote servers;

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

[1] – UPnP Device Architecture, version 1.0, UPnP Forum, October 15, 2008. Available at: http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0-20081015.pdf. Latest version available at: http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0.pdf.

[2] – Data elements and interchange formats – Information interchange -- Representation of dates and times, International Standards Organization, December 21, 2000. Available at: ISO 8601:2000.

[3] – IETF RFC 2046, Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types, N. Freed, Innosoft, N. Borenstein, First Virtual, November 1996. Available at: http://www.ietf.org/rfc/rfc2046.txt.

[4] – IETF RFC 2396, Uniform Resource Identifiers (URI): Generic Syntax, T. Berners-Lee, MIT/LCS, R. Fielding, U.C. Irvine, L. Masinter, Xerox Corporation, August 1998. Available at: http://www.ietf.org/rfc/rfc2396.txt.

[5] – IETF RFC 3261SIP: Session Initiation Protocol, J. Rosenberg, dynamicsoft, H. Schulzrinne, Columbia U., G. Camarillo, Ericsson, A. Johnston, WorldCom, J. Peterson,

Neustar, R. Sparks, dynamicsoft, M. Handley, ICIR, E. Schooler, AT&T June 2002. Available at: http://www.ietf.org/rfc/rfc3261.txt.

[6] – IETF RFC 3339, Date and Time on the Internet: Timestamps, G. Klyne, Clearswift Corporation, C. Newman, Sun Microsystems, July 2002. Available at: http://www.ietf.org/rfc/rfc3339.txt.

[7] – IETF RFC 3966, The tel URI for Telephone Numbers, H. Schulzrinne, Columbia University, December 2004. Available at: http://www.ietf.org/rfc/rfc3966.txt.

[8] – 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.

[9] – XML Schema Part 2: Data Types, Second Edition, Paul V. Biron, Ashok Malhotra, W3C Recommendation, 28 October 2004. Available at: http://www.w3.org/TR/2004/REC-xmlschema-2-20041028.

3 Terms, definitions, symbols and abbreviated terms

For the purposes of this document, the terms and definitions given in [1] and the following apply.

3.1 Provisioning terms

3.1.1

conditionally allowed
CA
The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is allowed, otherwise it is not allowed.

3.1.2

conditionally required
CR
The definition or behavior depends on a condition. If the specified condition is met, then the definition or behavior is required, otherwise it is not allowed.

3.1.3

not allowed
The definition or behavior is prohibited by this specification. Opposite of required.

3.2 Symbols

3.2.1

::
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.

3.3 General terms

3.3.1

Message
an information unit (e.g., e-Mail, SMS, MMS, Instant Message, etc.) that is exchanged between two or more entities (e.g., UPnP TelCP and TS/TC).

3.3.2

Messaging Session
a logical connection established between two or more recipients before starting to exchange Messages. The Messages are then orderly managed within the Messaging Session.

The Messaging Sessions are also referred to as “Sessions” in this specification.

3.3.3

Page Mode Messaging
a mode of sending and receiving messages from any recipients without establishing a session between the recipients.

The typical Messages exchanged under the Page Mode Messaging are e-Mail, SMS, and MMS.

3.3.4

Session Mode Messaging
the mode of sending and receiving Messages within a Messaging Session.

The typical Messages exchanged under the Session Mode Messaging are Instant Messages (IM) or chat. SMS, MMS, etc. can also be managed under the concept of Session Mode Messaging and can be used to logically group the Messages into an one conversation. It is also possible to have a Messaging Session with different Message classes.

3.3.5

Peer
a recipient for a Message or a participant in a Messaging Session.

3.4 Abbreviated terms

3.4.1

GUI
Graphical User Interface

3.4.2

ID
Identifier

3.4.3

IM
Instant Messaging

3.4.4

ISDN
Integrated Services Digital Network

3.4.5

MMS
Multimedia Messaging Service

3.4.6

PLMN
Public Land Mobile Network

3.4.7

PSTN
Public Switched Telephone Network

3.4.8

SMS
Short Message Service

3.4.9

TC
Telephony Client

3.4.10

TelCP
Telephony Control Point

3.4.11

TS
Telephony Server

3.4.12

VoIP
Voice over IP

3.4.13

WAN
Wide Area Network

4 Notations and conventions

4.1 Text conventions

- Strings that are to be taken literally are enclosed in “double quotes”.
- 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 [1] are printed using the **arch** character style.

4.2 Data Types

This specification uses data type definitions from two different sources. Data types from [1] are used to define state variable and action argument data types [1]. The XML Schema namespace is used to define property data types [9].

For Boolean data types from [1], it is strongly recommended to use the value “**0**” for false, and the value “**1**” for true. The values “**true**”, “**yes**”, “**false**”, or “**no**” may also be used but are not recommended. The values “**yes**” and “**no**” are deprecated and shall not be sent out by devices but shall be accepted on input.

For XML Schema defined Boolean data types, it is strongly recommended to use the value “*0*” for false, and the value “*1*” for true. The values “*true*”, “*yes*”, “*false*”, or “*no*” may also be used

but are not recommended. The values "*yes*" and "*no*" are deprecated and shall not be sent out by devices but shall be accepted on input.

4.3 Vendor-defined Extensions

Whenever vendors create additional vendor-defined state variables, actions or properties, their assigned names and XML representation shall follow the naming conventions and XML rules as specified in [1], 2.5, "Description: Non-standard vendor extensions".

5 Service Modeling Definitions

5.1 Service Type

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

urn:schemas-upnp-org:service:*Messaging:2*

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

5.2 *Messaging* Service Architecture

The *Messaging* service provides in the UPnP network the overall set of messaging capabilities of a phone (e.g., smartphone, IP phone, VoIP gateway, etc.), as the role of a TS.

According to the Telephony architecture in [10], this *Messaging* service can be included both in the TS or TC devices. More than one *Messaging* services can coexist in the same UPnP network, it's up to a TelCP to manage multiple *Messaging* services.

The *Messaging* service provides the TelCP with the following features:

- Sending, retrieving and deleting of Messages.
- Receiving notification of Messages.
- Managing messaging and file transfer Sessions.

Both Page Mode and Session Mode Messaging are supported by the *Messaging* service.

The architecture for the *Messaging* service is shown in Figure 1.

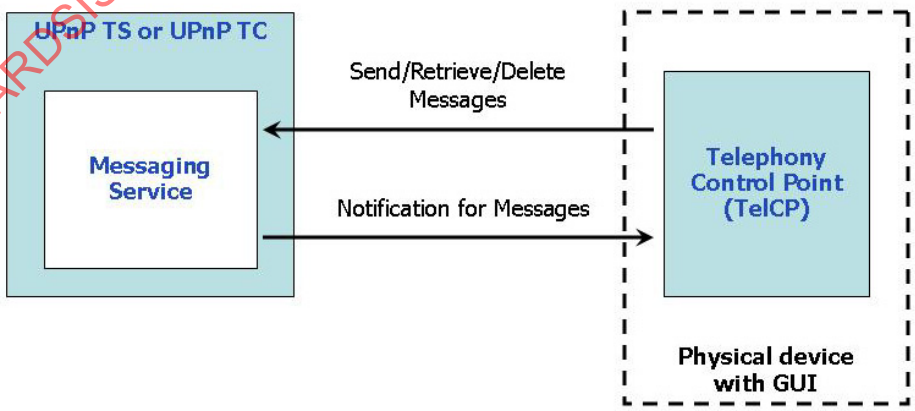

Figure 1 — Architecture of the *Messaging* Service

It is assumed that the TelCP features are implemented in a physical device that provides a Graphical User Interface (GUI), for a user to manage locally the messaging functionalities. These are however out of the scope of the *Messaging* service design.

The UPnP device that implements the *Messaging* service is capable of exposing messaging features to the peers through some telecommunication means (e.g., PSTN/ISDN, PLMN, VoIP, etc.). The implementation of these messaging features with peer networks are out of the scope.

It is possible to have a local user interface for the *Messaging* service in UPnP server. The local management of Messages by the end user should be taken into consideration when implementing the *Messaging* service.

5.3 State Variables

Note: For first-time reader, it may be more insightful to read the theory of operations first and then the action definitions before reading the state variable definitions.

5.3.1 State Variable Overview

Table 1 — State Variables

Variable Name	R/A [a]	Data Type	Reference
A_ARG_TYPE_MessagingCapabilities	*R*	string (XML fragment)	See 5.3.2
NewMessages	*R*	string (XML fragment)	See 5.3.3
SessionUpdates	*A*	string (XML fragment)	See 5.3.4
A_ARG_TYPE_TelephonyServerIdentity	*R*	string	See 5.3.5
A_ARG_TYPE_MessageID	*R*	string	See 5.3.6
A_ARG_TYPE_MessageClass	*A*	string	See 5.3.7
A_ARG_TYPE_MessageFolder	*A*	string	See 5.3.8
A_ARG_TYPE_MessageStatus	*A*	string	See 5.3.9
A_ARG_TYPE_Message	*R*	string (XML fragment)	See 5.3.10
A_ARG_TYPE_MessageList	*A*	string (XML fragment)	See 5.3.11
A_ARG_TYPE_SessionID	*A*	string	See 5.3.12
A_ARG_TYPE_SessionClass	*A*	string	See 5.3.13
A_ARG_TYPE_RecipientsList	*A*	string (XML fragment)	See 5.3.14
A_ARG_TYPE_SessionInfo	*A*	string (XML fragment)	See 5.3.15
A_ARG_TYPE_SessionsList	*A*	string (XML fragment)	See 5.3.16
A_ARG_TYPE_SupportedContentType	*A*	string	See 5.3.17
A_ARG_TYPE_Subject	*A*	string	See 5.3.18
A_ARG_TYPE_SessionStatus	*A*	string	See 5.3.19
A_ARG_TYPE_FileInfoList	*A*	string (XML fragment)	See 5.3.20

Variable Name	R/A [a]	Data Type	Reference
[a] R = required, A = allowed, CR = conditionally required, CA = conditionally allowed, X = Non-standard, add -D when deprecated (e.g., R-D, A-D).			

5.3.2 A_ARG_TYPE_MessagingCapabilities

This state variable defines an XML fragment that contains the features supported by the *Messaging* service, regarding:

- The Message classes supported for Page Mode Messaging;
- The Session classes supported for Session Mode Messaging;
- The Message folders supported.

5.3.2.1 XML Schema Definition

This is a string containing an XML fragment. The XML fragment in this argument shall validate against the XML schema for *MessagingCapabilities* in the XML namespace "urn:schemas-upnp-org:phone:messaging" which is located at "http://www.upnp.org/schemas/phone/messaging-v2.xsd".

5.3.2.2 Description of fields in the *MessagingCapabilities* structure

```
<?xml version="1.0" encoding="UTF-8"?>
<messaging:messagingCapabilities
   xsi:schemaLocation="urn:schemas-upnp-org:phone:messaging
   http://www.upnp.org/schemas/phone/messaging-v2.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:messaging="urn:schemas-upnp-org:phone:messaging">
   <supportedMessageClasses>
      <messageClass>Message class</messageClass>
      <!-- Any other messageClass(if any) go here.-->
   </supportedMessageClasses>
   <supportedSessionClasses>
      <sessionClass>Session class</sessionClass>
      <!-- Any other sessionClass(if any) go here.--></supportedSessionClasses>
   <supportedMessageFolders>
      <messageFolder>Message Folder</messageFolder>
      <!-- Any other messageFolder(if any) go here.-->
   </supportedMessageFolders>
</messaging:messagingCapabilities>
```

<xml>

Required. Case Sensitive.

<messagingCapabilities>

Required. Shall include a namespace declaration for the *Messaging* service Schema ("urn:schemas-upnp-org:phone:messaging"). This element includes the following sub elements:

<supportedMessageClasses>

Required. Indicates the list of supported Message classes.

<messageClass>

Required. Indicates the Message class as defined for the *A_ARG_TYPE_MessageClass* state variable. The empty value is not allowed. This element can appear one or more times.

<supportedSessionClasses>

Required. Indicates the list the supported Session classes.

<sessionClass>

Allowed. Indicates the Session class as defined for the *A_ARG_TYPE_SessionClass* state variable. The empty value is not allowed. This element can appear one or more times.

<supportedMessageFolders>

Required. Indicates the list of the supported folders for the Messages.

<messageFolder>

Required. Indicates the folder name as defined for the *A_ARG_TYPE_MessageFolder* state variable. The empty value is not allowed. This element can appear one or more times.

5.3.3 *NewMessages*

This state variable notifies incoming Messages (for both Page Mode and Session Mode Messaging), to the TelCP.

This state variable includes a list of *MessageID*s. The TelCP can retrieve the details for each Message using the *MessageID*. It can also contain an overview of the Message.

5.3.3.1 XML Schema Definition

This is a string containing an XML fragment. The XML fragment in this argument shall validate against the XML schema for *NewMessages* in the XML namespace "urn:schemas-upnp-org:phone:messaging" which is located at "http://www.upnp.org/schemas/phone/messaging-v2.xsd".

5.3.3.2 Description of fields in the *NewMessages* structure

```
<?xml version="1.0" encoding="UTF-8"?>
<messaging:newMessages
   xsi:schemaLocation="urn:schemas-upnp-org:phone:messaging
   http://www.upnp.org/schemas/phone/messaging-v2.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:messaging="urn:schemas-upnp-org:phone:messaging">
  <newMessage>
    <messageID>ID of the Message</messageID>
    <messageOverview>
      Preview of the Message (i.e.,SMS from +390112288046:Hallo,how…)
    </messageOverview>
  </newMessage>
  <!-- Any other newMessage(if any) go here.-->
</messaging:newMessages>
```

<xml>

Required. Case Sensitive.

<newMessages>

Required. Shall include a namespace declaration for the *Messaging* service Schema ("urn:schemas-upnp-org:phone:messaging"). This element includes the following sub elements:

<newMessage>

Allowed. Includes a MessageID and Message overview. This element can appear zero or more times.

<messageID>

Required. Includes the unique identifier of the Message.

<messageOverview>

Allowed. xsd:string, Includes the overview of the Message.

5.3.4 *SessionUpdates*

This state variable notifies any changes (e.g., session requested, session accepted, session closed, session modified, and user typing etc.) to the Messaging Sessions.

The notification message includes the Session ID, the Session status and any changes to the Session.

These information are contained in an XML data structure.

5.3.4.1 XML Schema Definition

This is a string containing an XML fragment. The XML fragment in this argument shall validate against the XML schema for *SessionUpdates* in the XML namespace "urn:schemas-upnp-org:phone:messaging" which is located at "http://www.upnp.org/schemas/phone/messaging-v2.xsd".

5.3.4.2 Description of fields in the *SessionUpdates* structure

```
<?xml version="1.0" encoding="UTF-8"?>
<messaging:sessionUpdates
   xsi:schemaLocation="urn:schemas-upnp-org:phone:messaging
   http://www.upnp.org/schemas/phone/messaging-v2.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:messaging="urn:schemas-upnp-org:phone:messaging">
   <sessionUpdate>
      <sessionID>ID of the Session</sessionID>
      <sessionEvent>Session event description</sessionEvent>
      <!-- Any other sessionEvent (if any) go here.-->
      <sessionStatus>Status of the session</sessionStatus>
   </sessionUpdate>
   <!-- Any other sessionUpdate (if any) go here.-->
</messaging:sessionUpdates>
```

<xml>

Required. Case Sensitive.

<sessionUpdates>

Required. Shall include a namespace declaration for the *Messaging* service Schema ("urn:schemas-upnp-org:phone:messaging"). This element includes the following sub elements:

<sessionUpdate>

Allowed. Includes sessionID, Session events to capture the Session changes and Session status. This element can appear zero or more times.

<sessionID>

Required. Includes the unique identifier of the Session.

<sessionEvent>

Required. xsd:string, Includes the descriptions of the events occurred. This element can appear one or more times. Table 2 shows the possible values of sessionEvent element and the resulting Session status:

Table 2 — Allowed values for *sessionEvent* and the corresponding values of the *sessionStatus*

sessionEvent	sessionStatus	Description
"session requested by: "<peer:name>	"*Pending*"	An incoming Session has been requested by a peer

sessionEvent	sessionStatus	Description
“session accepted by: ”<peer:name>	“*Running*”	A Session has been accepted by a peer
“session closed”	“*Closed*”	The Session has been closed or rejected
“session modified”	“*Running*”	The Session has been modified
<peer:name>“ is typing”	“*Running*”	One of the Session participants is typing a Message
“one TelCP has left”	“*Running*”/ “*Parked*”	One of the TelCP has left the Session. The Session status shall be “*Parked*” if the last TelCP has left.
“one TelCP has joined”	“*Running*”	A “*Parked*” Session has been joined by a TelCP
<peer:name> “ has left”	“*Running*”	One of the peers has left the Session.
<peer:name> “ has joined”	“*Running*”	One of the peers has joined the Session.
“file transfer started”	“*Running*”	A file transfer has started within the Session
“file transfer completed”	“*Running*”	A file transfer has been completed within a Session
“file transfer cancelled”	“*Running*”	A file transfer has been cancelled
“file transfer progress: ”<*nBytes*>“ Bytes”	“*Running*”	A file transfer is in progress within the Session and *nBytes* Bytes have been transferred since the transfer has started.
“file download failed”	“*Running*”	The file can not be retrieved by the TS/TC.
TBD	*TBD*	*(Specified by UPnP vendors.)*

<sessionStatus>

Required. Includes the current status of the Session.

5.3.5 *A_ARG_TYPE_TelephonyServerIdentity*

This state variable contains the unique identifier of a Telephony Server. The identity of a Telephony Server is expressed using the standard URI (Unified Resource Identifier) scheme as specified in [4]. In case of SIP, the identity of the Telephony Server identity contains the SIP URI. In case of a generic resource identified by a telephone number, the Telephony Server identity contains the TEL URI [7].

5.3.6 *A_ARG_TYPE_MessageID*

This state variable contains an identifier that uniquely identifies a Message.

The format of this state variable is as follows:

- *MessageID*
  - to be used for Page Mode Messages;
- *SessionID*“.”*MessageID*
  - to be used for Session Mode Messages.

where:

- The *MessageID* is an unique alphanumerical identifier that shall contain only characters from the sets: “0..9”, “A..Z”, “a..z”, generated by the UPnP device hosting the *Messaging* service.
- The *SessionID* is an unique alphanumerical identifier of the Messaging Session, as defined in 5.3.12. It’s applicable for Session Mode Messaging.

Example:

Some examples of valid values for this variable are:

- Valid examples for Page Mode Messaging are: 123456, qwerty, M000123, XXY123.
- Valid examples for Session Mode Messaging are: abcdef.789012, ses001.123455.

5.3.7 *A_ARG_TYPE_MessageClass*

This state variable contains the value for the Message class. The *Messaging* service shall support at least one of the Message classes. The possible values of this state variable are shown in the table below.

Table 3 — allowedValueList for the *A_ARG_TYPE_MessageClass* state variable

Value	R/A [a]
“*e-Mail*”	*A*
“*SMS*” (Default)	*A*
“*MMS*”	*A*
“*Instant Message*”	*A*
“*WebpageURL*”	*A*
[a] *R* = required, *A* = allowed, *X* = Non-standard.	

where:

- “*e-Mail*” refers to an e-Mail Message;
- “*SMS*” refers to an SMS Message;
- “*MMS*” refers to an MMS Message;
- “*Instant Message*” refers to an instant Message (e.g., within a chat Session);
- “WebpageURL” refers to a notification for an webpage sync messa

Vendor specific extensions are allowed for the values of this argument.

5.3.8 *A_ARG_TYPE_MessageFolder*

This state variable contains the possible values for the folders where Messages are kept and organized by the *Messaging* service.

This state variable is introduced to provide type information for various action arguments that refers to folders of Messages.

The possible values of this state variable are shown in the table below.

Table 4 — allowedValueList for the *A_ARG_TYPE_MessageFolder* state variable

Value	R/A [a]
"*Received*" (DEFAULT)	*R*
"*Outgoing*"	*R*
"*Sent*"	*R*
"*Deleted*"	*A*
"*ClosedSession*"	*A*
[a] *R* = required, *A* = allowed, *X* = Non-standard.	

where:

- The "*Received*" folder contains the Messages that have been received by the *Messaging* service;
- The "*Outgoing*" folder contains the Messages that are in the process of being sent by the *Messaging* service;
- The "*Sent*" folder contains the Messages that have been sent by the *Messaging* service;
- The "*Deleted*" folder contains the Messages that have been deleted from all other folders;
- The "*ClosedSession*" folder contains information related to all the closed and terminated Sessions. The Session informatiation includes Session ID, Session Class, and Messages exchanged during the Session;

5.3.9 *A_ARG_TYPE_MessageStatus*

This state variable contains the current status of a Message. The status includes whether a Message has been read or yet to be read. This state variable is used to provide type information for arguments of various actions (i.e., *GetMessages()*). The possible values of this state variable are shown in the table below.

Table 5 — allowedValueList for the *A_ARG_TYPE_MessageStatus* state variable

Value	R/A [a]
"*Read*"	*R*
"*Unread*"	*R*
[a] *R* = required, *A* = allowed, *X* = Non-standard.	

where:

- "*Read*" means the Message has been read by a TelCP by invocation of the *ReadMessage()* action, or by using the local user interface of the device.
- "*Unread*" means the Message has not been read yet.

5.3.10 *A_ARG_TYPE_Message*

This state variable defines an XML fragment that contains the details of a Message which includes the ID of the Message, the ID of the Session, the ID of the reply Message, information about the recipients, the subject, the text for the Message, and any attachment if exist, etc.

5.3.10.1 XML Schema Definition

This is a string containing an XML fragment. The XML fragment in this argument shall validate against the XML schema for *Message* in the XML namespace "urn:schemas-upnp-org:phone:messaging" which is located at "http://www.upnp.org/schemas/phone/messaging-v2.xsd".

The imported XML namespace "urn:schemas-upnp-org:phone:peer" is here referenced for the Complex Type *peerType*.

5.3.10.2 Description of fields in the *Message* structure

```
<?xml version="1.0" encoding="UTF-8"?>
<messaging:message
   xsi:schemaLocation="urn:schemas-upnp-org:phone:messaging
   http://www.upnp.org/schemas/phone/messaging-v2.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:messaging="urn:schemas-upnp-org:phone:messaging"
   xmlns:peer="urn:schemas-upnp-org:phone:peer">
   <messageID>ID of the Message(i.e., abcdef.123456)</messageID>
   <sessionID>ID of the Session</sessionID>
   <messageClass>Class of the Message</messageClass>
   <messageFolder>Folder where the Message is stored</messageFolder>
   <messageStatus>Status of the Message</messageStatus>
   <replyMessageID>ID of the replied Message</replyMessageID>
   <recipientsList>
      <recipientTo>
         Recipient information (i.e.,
         <peer:id>jane.doe@acme.com</peer:id>
         <peer:contactInstanceId>12</peer:contactInstanceId>)
      </recipientTo>
      <!-- Any other recipientTo (if any) go here.-->
      <recipientCc>
         Cc Recipient information (i.e.,
         <peer:id>Bull.Dog@dogs.com</peer:id>)
      </recipientCc>
      <!-- Any other recipientCc (if any) go here.-->
      <recipientBcc>
         Bcc Recipient information (i.e.,
         <peer:id>Fox.Terrier@dogs.com</peer:id>
         <peer:name>Fox Terrier</peer:name>
         <peer:contactInstanceId>4</peer:contactInstanceId>)
      </recipientBcc>
      <!-- Any other recipientBcc (if any) go here.-->
      <recipientFrom>
         Sender information (i.e.,
         <peer:id>The.Sender@fromhere.com</peer:id>
         <peer:name>Bob the Sender</peer:name>)
      </recipientFrom>
   </recipientsList>
   <subject>Message title</subject>
   <text>Message text</text>
   <attachments>
      <attachment>
         <uri>
         URI of the attachment(i.e.,http://192.168.1.3/file1.pdf)
         </uri>
         <size>Size of the attachment</size>
         <mimeType>MIME type of the attachment</mimeType>
      </attachment>
      <!-- Any other attachment (if any) go here.-->
   </attachments>
   <dateSent>Date when the Message is sent</dateSent>
   <dateReceived>Date when the Message is received</dateReceived>
</messaging:message>
```

<xml>

Required. Case Sensitive.

<message>

Required. Shall include a namespace declaration for the Complex Type <peerType> (“urn:schemas-upnp-org:phone:peer” as from the Annex A) and a namespace declaration for the *Messaging* service Schema (“urn:schemas-upnp-org:phone:messaging”). This element includes the following sub elements:

<messageID>

Required. Indicates the unique identifier of the Message (see 5.3.6).

<sessionID>

Allowed, Indicates the unique identifier of the Session which the Message belongs to (see 5.3.12). This element shall exist if the case of Session Mode Messaging.

<messageClass>

Required. Indicates the class of the Message (see 5.3.7)

<messageFolder>

Required. Indicates the folder in which the Message is currently stored (see 5.3.8).

<messageStatus>

Required. Indicates the current status of the Message (see 5.3.9).

<replyMessageID>

Allowed. Indicates the id of the Message which is being replied.

<recipientsList>

Required. Indicates the list of recipients for a Message as defined in 5.3.14. This element should reflect the current list of participants in a Session Mode Messaging.

<recipientTo>

Allowed. peer:peerType, Indicates the recipient to whom the Message is sent. This element shall exist if the value of the Message class is either SMS or MMS or Instant Message. In the case of e-Mail messages, at least one of the following elements shall exist: <recipientTo>, <recipientCc> or <recipientBcc>. This element can appear zero or more times.

<recipientCc>

Allowed. peer:peerType, Indicates the recipient to whom the Message is copied (CC); This element can appear zero or more times.

<recipientBcc>

Allowed. peer:peerType, Indicates the recipient to whom the Message is blind copied (BCC). This element can appear zero or more times.

<recipientFrom>

Required. peer:peerType, Indicates the sender of the Message.

<subject>

Allowed. xsd:string, Indicates the title of the Message.

<text>

Allowed. xsd:string, Includes the text content of the Message.

<attachments>

Allowed. Includes a list of attachments. This element shall exist if the value of the message class is MMS. It has the following sub elements.

<attachment>

Required. Includes the information of the file attached to the Message. Thbis element can appear one or more times. It has the following sub elements.

<uri>

Required. xsd:anyURI, Indicates the identifier for the resource as defined by [4].

<size>

Allowed. xsd:nonNegativeInteger, Indicates the size in Bytes of the resource.

<mimeType>

Allowed. xsd:string, Indicates the MIME type of the resource as defined by [3].

<dateSent>

Allowed. xsd:dateTime, Indicates the date when the Message is sent by the *Messaging* service as specified by [6]. This element shall exist in the sent Message.

<dateReceived>

Allowed. xsd:dateTime, Indicates the date when the Message is received by the *Messaging* service as specified by [6]. This element shall exist in the received Message.

5.3.11 *A_ARG_TYPE_MessageList*

This state variable defines an XML fragment that contains a list of Messages, each one with some or all of its details (including the ID of the Message).

This state variable is introduced to provide type information for various action arguments that return a list of Messages.

5.3.11.1 XML Schema Definition

This is a string containing an XML fragment. The XML fragment in this argument shall validate against the XML schema for *MessageList* in the XML namespace "urn:schemas-upnp-org:phone:messaging" which is located at "http://www.upnp.org/schemas/phone/messaging-v2.xsd".

The imported XML namespace "urn:schemas-upnp-org:phone:peer" is here referenced for the Complex Type *peerType*.

5.3.11.2 Description of fields in the *MessageList* structure

```
<?xml version="1.0" encoding="UTF-8"?>
<messaging:messageList
   xsi:schemaLocation="urn:schemas-upnp-org:phone:messaging
   http://www.upnp.org/schemas/phone/messaging-v2.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:messaging="urn:schemas-upnp-org:phone:messaging"
   xmlns:peer="urn:schemas-upnp-org:phone:peer">
   <message>
      <messageID>ID of the Message</messageID>
      <messageClass>Class of the Message</messageClass>
      <messageFolder>Message Folder</messageFolder>
      <messageStatus>Status of the Message</messageStatus>
      <recipientsList>
         <recipientTo>
            Recipient information (i.e.,
            <peer:id>jane.doe@acme.com</peer:id>
            <peer:contactInstanceId>12</peer:contactInstanceId>)
         </recipientTo>
         <!-- Any other recipientTo (if any) go here.-->
            <recipientFrom>
               Sender information (i.e.,
```

```
                <peer:id>The.Sender@fromhere.com</peer:id>
                <peer:name>Bob the Sender</peer:name>)
            </recipientFrom>
        </recipientsList>
        <subject>Message title</subject>
        <!-- Can have additional details -->
    </message>
    <!-- Any other message (if any) go here.-->
</messaging:messageList>
```

<xml>

Required. Case Sensitive.

<messageList>

Required. Shall include a namespace declaration for the Complex Type <peerType> (“urn:schemas-upnp-org:phone:peer” as from the Annex A) and a namespace declaration for the *Messaging* service Schema (“urn:schemas-upnp-org:phone:messaging”). This element includes the following sub elements:

<message>

Allowed. Contains the details of a Message as specified in 5.3.10. This element can appear zero or more times.

5.3.12 *A_ARG_TYPE_SessionID*

This state variable contains an identifier that uniquely identifies a Messaging Session.

The Session identifier (*SessionID*) is a unique alphanumerical identifier that shall contain only characters from the sets: “0..9”, “A..Z”, “a..z”, generated by the UPnP device hosting the *Messaging* service.

Example:

Some examples of valid values for this variable are:

- 654321
- abcdef
- ses0123
- XYZ123

5.3.13 *A_ARG_TYPE_SessionClass*

This state variable contains the value for the Session class. The possible values for this state variable are listed in the Table 6.

Table 6 — allowedValueList for the *A_ARG_TYPE_SessionClass* state variable

Value	R/A [a]
“*e-Mail*”	*A*
“*SMS*”	*A*
“*MMS*”	*A*
“*Chat*” (DEFAULT)	*A*
“*File-Transfer*”	*A*
“*WebpageSync*”	*A*
“*Mixed*”	*A*

Value	R/A [a]
[a] R = required, A = allowed, X = Non-standard	

where:

- "*e-Mail*" refers to an e-Mail Messaging Session;
- "*SMS*" refers to an SMS Messaging Session;
- "*MMS*" refers to an MMS Messaging Session;
- "*Chat*" refers to an instant Messaging Session, or IM Session;
- "*File-Transfer*" refers to a Session for transferring files;
- "*WebpageSync*" refers to a Session for synchronizing web pages;
- "*Mixed*" refers to a Messaging Session that can contain different classes of Messages including file transfer;

Vendor specific extensions are allowed for the values of this argument.

5.3.14 *A_ARG_TYPE_RecipientsList*

This state variable defines an XML fragment that contains the list of recipients for a Message or a Messaging Session.

A recipient can either be a contact or a group of contacts, and is identified by its address. It can also include a name and identification from the Address Book of the *Phone Data Model*, if implemented.

5.3.14.1 XML Schema Definition

This is a string containing an XML fragment. The XML fragment in this argument shall validate against the XML schema for *RecipientsList* in the XML namespace "`urn:schemas-upnp-org:phone:messaging`" which is located at "`http://www.upnp.org/schemas/phone/messaging-v2.xsd`".

The imported XML namespace "`urn:schemas-upnp-org:phone:peer`" is here referenced for the Complex Type `<peerType>`.

5.3.14.2 Description of fields in the *RecipientsList* structure

```
<?xml version="1.0" encoding="UTF-8"?>
<messaging:recipientsList
   xsi:schemaLocation="urn:schemas-upnp-org:phone:messaging
   http://www.upnp.org/schemas/phone/messaging-v2.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:messaging="urn:schemas-upnp-org:phone:messaging"   xmlns:peer="urn:schemas-
upnp-org:phone:peer">
   <recipientTo>
      Recipient information (i.e.,
      <peer:id>MyGroupURI</peer:id>
      <peer:name>MyGroup</peer:name>
      <peer:contactGroupId>37</peer:contactgroupId>)
   </recipientTo>
   <!-- Any other recipientTo (if any) go here.-->
   <recipientCc>
      Cc Recipient information (i.e.,
      <peer:id>Bull.Dog@dogs.com</peer:id>)
   </recipientCc>
   <!-- Any other recipientCc (if any) go here.-->
   <recipientBcc>
      Bcc Recipient information (i.e.,
```

```
      <peer:id>Fox.Terrier@dogs.com</peer:id>
      <peer:name>Fox Terrier</peer:name>
      <peer:contactInstanceId>4</peer:contactInstanceId>)
   </recipientBcc>
   <!-- Any other recipientBcc (if any) go here.-->
   <recipientFrom>
      Sender information (i.e.,
      <peer:id>The.Sender@fromhere.com</peer:id>
      <peer:name>Bob the Sender</peer:name>)
   </recipientFrom>
</messaging:recipientsList>
```

<xml>

Required. Case Sensitive.

<recipientsList>

Required. Shall include a namespace declaration for the Complex Type <peerType> ("urn:schemas-upnp-org:phone:peer" as from the Annex A) and a namespace declaration for the *Messaging* service Schema ("urn:schemas-upnp-org:phone:messaging"). This element includes the following sub elements:

<recipientTo>

Allowed. peer:peerType, Indicates the recipient to whom the Message is sent. This element can appear zero or more times.

<recipientCc>

Allowed. peer:peerType, Indicates the recipient to whom the Message is copied (CC); This element can appear zero or more times.

<recipientBcc>

Allowed. peer:peerType, Indicates the recipient to whom the Message is blind copied (BCC); This element can appear zero or more times.

<recipientFrom>

Required. peer:peerType, Indicates the sender of the Message or the creator of the Messaging Session.

5.3.15 *A_ARG_TYPE_SessionInfo*

This state variable defines an XML fragment that contains the details of a Messaging Session which includes the ID of the Session, the Session class, the information about the recipients, the subject, the status of the Session, and the list of Messages, etc.

5.3.15.1 XML Schema Definition

This is a string containing an XML fragment. The XML fragment in this argument shall validate against the XML schema for *SessionInfo* in the XML namespace "urn:schemas-upnp-org:phone:messaging" which is located at "http://www.upnp.org/schemas/phone/messaging-v2.xsd".

The imported XML namespace "urn:schemas-upnp-org:phone:peer" is here referenced for the Complex Type *peerType*.

5.3.15.2 Description of fields in the *SessionInfo* structure

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



```
   xsi:schemaLocation="urn:schemas-upnp-org:phone:messaging
   http://www.upnp.org/schemas/phone/messaging-v2.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:messaging="urn:schemas-upnp-org:phone:messaging"   xmlns:peer="urn:schemas-
upnp-org:phone:peer">
   <sessionID>ID of the Session (i.e., ses000372)</sessionID>
   <sessionClass>Class of the Session</sessionClass>
   <subject>Title of the Session</subject>
   <sessionStatus>Status of the Session</sessionStatus>
   <supportedContentType>
      Supported content type list (i.e., image/jpeg,video/mpeg)
   </supportedContentType>
   <recipientsList>
      <recipientTo>
         Recipient information (i.e.,
         <peer:id>john.doe@acme.com</peer:id>
      </recipientTo>
      <!-- Any other recipientTo (if any) go here.-->
      <recipientCc>
         Cc Recipient information (i.e.,
         <peer:id>diego.armando@biloxi.com</peer:id>)
      </recipientCc>
      <!-- Any other recipientCc (if any) go here.-->
      <recipientBcc>
         Bcc Recipient information (i.e.,
         <peer:id>Bob.Cat@cats.com</peer:id>
         <peer:name>Bob</peer:name>
         <peer:contactInstanceId>4</peer:contactInstanceId>)
      </recipientBcc>
      <!-- Any other recipientBcc (if any) go here.-->
      <recipientFrom>
         Session creator information (i.e.,
         <peer:id>Bull.Dog@dogs.com</peer:id>)
      </recipientFrom>
   </recipientsList>
   <listOfMessages>
      <messageID>ID of a Message</messageID>
      <!-- Any other messageID (if any) go here.-->
   </listOfMessages>
   <dateStarted>Date when the Session started</dateStarted>
   <dateLastMessage>Date of the last Message</dateLastMessage>
   <dateEnded>Date when the Session terminated</dateEnded>
</messaging:sessionInfo>
```


`<xml>`

Required. Case Sensitive.

`<sessionInfo>`

Required. Shall include a namespace declaration for the Complex Type <peerType> (“urn:schemas-upnp-org:phone:peer” as from the Annex A) and a namespace declaration for the *Messaging* service Schema (“urn:schemas-upnp-org:phone:messaging”). This element contains the following elements:

`<sessionID>`

Required. Indicates the identifier that uniquely identifies the Messaging Session (see 5.3.12).

`<sessionClass>`

Required. Indicates the class of Messaging Session (see 5.3.13).

`<subject>`

Allowed. xsd:string, Indicates the current title of the Messaging Session.

`<sessionStatus>`

Required. Indicates the current status of the Session (see 5.3.19).

`<supportedContentType>`

Allowed. Includes a list of supported contents that can be shared among the participants in the Session (see 5.3.17).

`<recipientsList>`

Required. Indicates the list of recipients for the Messaging Session (see 5.3.14).

`<recipientTo>`

Allowed. peer:peerType, Indicates the recipient to whom the Messages are sent. This element shall exist if the value of the Session class is either SMS or MMS or Instant Message. In the case of e-Mail Sessions, at least one of the following elements shall exist: `<recipientTo>`, `<recipientCc>` or `<recipientBcc>`. This element can appear zero or more times.

`<recipientCc>`

Allowed. peer:peerType, Indicates the recipient to whom the Messages are copied (CC); This element can appear zero or more times.

`<recipientBcc>`

Allowed. peer:peerType, Indicates the recipient to whom the Messages are blind copied (BCC); This element can appear zero or more times.

`<recipientFrom>`

Required. peer:peerType, Indicates the creator of the Messaging Session.

`<listOfMessages>`

Allowed. Includes the list of Message IDs sorted in a chronological order. It has the following sub elements:

`<messageID>`

Allowed. Indicates the unique identifier of the Message. It shall be in the format SessionID.MessageID (see 5.3.6). This element can appear zero or more times.

`<dateStarted>`

Allowed. xsd:dateTime, Indicates the date when the Session was started.

`<dateLastMessage>`

Allowed. xsd:dateTime, Indicates the date when a Message was last exchanged in the Session.

`<dateEnded>`

Allowed. xsd:dateTime, Indicates the date when the Session was terminated.

5.3.16 *A_ARG_TYPE_SessionsList*

This state variable defines an XML fragment that contains a list of Sessions, each one with some or all of its details (including the ID of the Session).

This state variable is introduced to provide type information for various action arguments that return a list of Sessions.

5.3.16.1 XML Schema Definition

This is a string containing an XML fragment. The XML fragment in this argument shall validate against the XML schema for *SessionsList* in the XML namespace "`urn:schemas-upnp-org:phone:messaging`" which is located at "`http://www.upnp.org/schemas/phone/messaging-v2.xsd`".

The imported XML namespace "urn:schemas-upnp-org:phone:peer" is here referenced for the Complex Type *peerType*.

5.3.16.2 Description of fields in the *SessionsList* structure

```
<?xml version="1.0" encoding="UTF-8"?>
<messaging:sessionsList
   xsi:schemaLocation="urn:schemas-upnp-org:phone:messaging
   http://www.upnp.org/schemas/phone/messaging-v2.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:messaging="urn:schemas-upnp-org:phone:messaging"   xmlns:peer="urn:schemas-
upnp-org:phone:peer">
   <sessionInfo>
      <sessionID>ID of the Session</sessionID>
      <sessionClass>Class of the Session</sessionClass>
      <sessionStatus>Status of the Session</sessionStatus>
      <recipientsList>
      <recipientTo>
         Recipient information
      </recipientTo>
      <!-- Any other recipientTo (if any) go here.-->
      <recipientCc>
         Cc Recipient information
      </recipientCc>
      <!-- Any other recipientCc (if any) go here.-->
      <recipientBcc>
         Bcc Recipient information
      </recipientBcc>
      <!-- Any other recipientBcc (if any) go here.-->
      <recipientFrom>
         Session creator information
      </recipientFrom>
   </recipientsList>
   </sessionInfo>
   <!-- Any other sessionInfo (if any) go here.-->
</messaging:sessionsList>
```

<xml>

Required. Case Sensitive.

<sessionsList>

Required. Shall include a namespace declaration for the Complex Type <peerType> ("urn:schemas-upnp-org:phone:peer" as from the Annex A) and a namespace declaration for the *Messaging* service Schema ("urn:schemas-upnp-org:phone:messaging"). This element contains the following elements:

<sessionInfo>

Allowed. Includes the details of the Session as defined in 5.3.15. This element can appear zero or more times.

5.3.17 *A_ARG_TYPE_SupportedContentType*

This state variable contains a list of content types supported by the Messaging Session.

The format of this state variable is a comma separated values of MIME types [3].

Example:

Some example of valid values for this state variable are:

- image/jpeg
- image/png,video/avi,video/mpeg

5.3.18 *A_ARG_TYPE_Subject*

This state variable contains the title of a Session.

5.3.19 *A_ARG_TYPE_SessionStatus*

This state variable contains the current status of a Messaging Session. The possible values of this state variable are shown in Table 7.

Table 7 — allowedValueList for the *A_ARG_TYPE_SessionStatus* state variable

Value	R/A [a]
"*Pending*"	*R*
"*Running*"	*R*
"*Parked*"	*R*
"*Closed*"	*A*
[a] *R* = required, *A* = allowed, *X* = Non-standard	

where:

- "*Pending*" refers to the state of a Session which has been initiated, but has not been accepted yet by a TelCP or a remote peer;
- "*Running*" refers to the state of a Session which is active and at least one TelCP is participating in the Session.
- "*Parked*" refers to the state of a Session which has no TelCP participating in the Session.
- "*Closed*" refers to the state of a Session which is currently inactive. This is due to the fact that the Session has been closed by a local TelCP or by a remote peer.

5.3.20 *A_ARG_TYPE_FileInfoList*

This state variable defines an XML fragment that contains information of a file transfer Session.

5.3.20.1 XML Schema Definition

This is a string containing an XML fragment. The XML fragment in this argument shall validate against the XML schema for *FileInfoList* in the XML namespace "`urn:schemas-upnp-org:phone:messaging`" which is located at "`http://www.upnp.org/schemas/phone/messaging-v2.xsd`".

5.3.20.2 Description of fields in the *FileInfoList* structure

```
<?xml version="1.0" encoding="UTF-8"?>
<messaging:fileInfoList
   xsi:schemaLocation="urn:schemas-upnp-org:phone:messaging
   http://www.upnp.org/schemas/phone/messaging-v2.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:messaging="urn:schemas-upnp-org:phone:messaging">
   <sessionID>ID of the Session</sessionID>
   <fileInfo>
      <fileType>Type of the file</fileType>
      <fileSize>Size of the file</fileSize>
      <fileLink>
         Link to the file (i.e., http://tcp.com/song.mp3)
      </fileLink>
      <progressNotif time=notification period>
         Progress notification switch
      </progressNotif>
```



```
    </fileInfo>
    <!-- Any other fileInfo (if any) go here.-->
</messaging:fileInfoList>
```

<xml>

Required. Case Sensitive.

<fileInfoList>

Required. Shall include a namespace declaration for the *Messaging* service Schema ("urn:schemas-upnp-org:phone:messaging"). This element contains the following elements and attributes:

<sessionID>

Required. Indicates the identifier that uniquely identifies the File Transfer Session (see 5.3.12).

<fileInfo>

Required. Includes the metadata of the file. It can contain the following sub elements and can have one or more instances.

<fileType>

Required. xsd:string, Indicates the type of the file (e.g., Audio-file or content type of the file).

<fileSize>

Required. xsd:nonNegativeInteger, indicates the size of the file in bytes.

<fileLink>

Required. xsd:anyURI, indicates the URL for the file.

<progressNotif>

Required. xsd:boolean, Indicates the type of notification required (e.g., periodic notification or notification after completion of the file transfer). If set to 1, the notification period can be specified by the following attribute:

time

Allowed. xsd:positiveInteger, Indicates the notification period of the file transfer which is in progress. This attribute shall exist if the <ProgressNotif> element is set to 1.

5.4 Eventing and Moderation

Table 8 — Event Moderation

Variable Name	Evented	Moderated Event	Max Event Rate a (seconds)	Logical Combination	Min Delta per Event b
NewMessages	*YES*	*YES*	1		
SessionUpdates	*YES*	*YES*	1		
A_ARG_TYPE_MessagingCapabilities	*No*	*No*			
A_ARG_TYPE_TelephonyServerIdentity	*No*	*No*			
A_ARG_TYPE_MessageID	*No*	*No*			
A_ARG_TYPE_MessageClass	*No*	*No*			
A_ARG_TYPE_MessageFolder	*No*	*No*			
A_ARG_TYPE_MessageStatus	*No*	*No*			
A_ARG_TYPE_Message	*No*	*No*			
A_ARG_TYPE_MessageList	*No*	*No*			
A_ARG_TYPE_SessionID	*No*	*No*			
A_ARG_TYPE_SessionClass	*No*	*No*			
A_ARG_TYPE_RecipientsList	*No*	*No*			
A_ARG_TYPE_SessionInfo	*No*	*No*			
A_ARG_TYPE_SessionsList	*No*	*No*			
A_ARG_TYPE_SupportedContentType	*No*	*No*			
A_ARG_TYPE_Subject	*No*	*No*			
A_ARG_TYPE_SessionStatus	*No*	*No*			
A_ARG_TYPE_FileInfoList	*No*	*No*			

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

b (N) * (allowedValueRange Step).

5.4.1 Eventing of *NewMessages*

The moderation of this state variable takes into consideration for both Page and Session Mode Messaging.

This state variable can be evented when a new incoming Message arrives. However, this state variable shall not be evented at a higher rate than 1 s. If multiple Messages arrive in a 1 s interval, then all the events shall be accumulated into a single event Message and the event Message will be sent after 1 s.

In order to have a security aware *Messaging* service implementation the state variable *NewMessages* may event only the *messageID* when a new incoming message arrives while it may return full set of information only to authorized TelCPs in response to *GetNewMessages()* action invoked via TLS tunnel.

5.4.2 Eventing of *SessionUpdates*

This state variable can be evented when one of the following events occurrs within a Session:

- A new Session request has arrived;

- A “Pending” Session has been accepted;
- A Session has been closed;
- A Session has been modified;
- A participant is typing a Message (if supported);
- A TelCP or a participant has left the Session;
- A new TelCP or a new participant has joined the Session;
- A file transfer has been started or cancelled;
- A file transfer progress notification is available;
- A download of a file to transfer has failed.

However, this state variable shall not be evented at a higher rate than 1 s. If multiple Messages arrive in a 1 s interval, then all the events shall be accumulated into a single event Message and the event Message will be sent after 1 s.

In order to have a security aware *Messaging* service implementation the state *variableSessionUpdates* may only event *sessionID* when the status of the Messaging Session has changed while it may return full set of information only to authorized TelCPs in response to *GetSessionUpdates()* action invoked via TLS tunnel.

5.5 Actions

Table 9 lists the actions of the *Messaging* service.

The *Messaging* service requires the implementation of all the actions necessary for delivering the complete set of functions for Page Mode Messaging, while Session Mode Messaging (and corresponding specific and dedicated actions) is left optional.

Table 9 — Actions

Name	Device R/A [a]	Control Point R/A [b]
GetTelephonyIdentity()	*R*	*R*
GetMessagingCapabilities()	*R*	*R*
GetNewMessages()	*A*	*A*
SearchMessages()	*A*	*A*
ReadMessage()	*R*	*R*
SendMessage()	*R*	*R*
DeleteMessage()	*R*	*R*
CreateSession()	*A*	*A*
ModifySession()	*A*	*A*
AcceptSession()	*A*	*A*
GetSessionUpdates()	*A*	*A*
GetSessions()	*A*	*A*
JoinSession()	*A*	*A*
LeaveSession()	*A*	*A*
CloseSession()	*A*	*A*
StartFileTransfer()	*A*	*A*
CancelFileTransfer()	*A*	*A*

Name	Device R/A [a]	Control Point R/A [b]
GetFileTransferSession()	A	A

a For a device this column indicates whether the action shall be implemented or not, where R = required, A = allowed, CR = conditionally required, CA = conditionally allowed, X = Non-standard, add -D when deprecated (e.g., R-D, A-D).

b For a control pont this column indicates whether a control point shall be capable of invoking this action, where R = required, A = allowed, CR = conditionally required, CA = conditionally allowed, X = Non-standard, add -D when deprecated (e.g., R-D, A-D).

5.5.1 GetTelephonyIdentity()

This action returns the identity of the Telephony Server. If the action succeeds then the output argument TelephonyIdentity contains the unique identity of the Telephony Server. If the identity of the Telephony Server has not been assigned by the time this action is invoked then this action will fail with an error code.

5.5.1.1 Arguments

Table 10 — Arguments for GetTelephonyIdentity()

Argument	Direction	relatedStateVariable
TelephonyIdentity	OUT	A_ARG_TYPE_TelephonyServerIdentity

5.5.1.2 Argument Descriptions

The value of the output argument TelephonyIdentity shall contain a valid URI as specified in [4]. The value follows the standard URI definitions. In case of SIP, the SIP URI [5] is used. In case of a generic resource identified by a telephone number, the TEL URI [7] is used.

The examples of SIP URI as the value of the output argument TelephonyIdentity are as follows:

sip:alice@atlanta.com
sip:alice:secretword@atlanta.com;transport=tcp
sips:alice@atlanta.com?subject=project%20x&priority=urgent
sip:+1-212-555-1212:1234@gateway.com;user=phone
sips:1212@gateway.com
sip:alice@192.0.2.4

The `sip:` URI scheme (`sips:`, for resources to be contacted securely) is defined in [5]. The general form, is: sip:user:password@host:port;uri-parameters?headers

The examples of TEL URI as the value of the output argument TelephonyIdentity are as follows:

Tel:+1-201-555-0123 (This URI points to a phone number in the United States. The hyphens are included to make the number more human readable; they separate country, area code and subscriber number.)

tel:7042;phone-context=example.com (This URI describes a local phone number valid within the context "example.com".)

tel:863-1234;phone-context=+1-914-555 (This URI describes a local phone number that is valid within a particular phone prefix.)

The URI scheme tel [7] describes any resources identified by telephone numbers. A telephone number is a string of decimal digits that uniquely indicates the network termination point. The number contains the information necessary to route the call to this point.

Dealing with conversion and compatibility rules among SIP/SIPS URI and TEL URI is described in [5], 19.1.6. Implementers shall be aware of these recommendations.

5.5.1.3 Service Requirements

None.

5.5.1.4 Control Point Requirements When Calling The Action

None.

5.5.1.5 Dependency on Device State

None.

5.5.1.6 Effect on Device State

None.

5.5.1.7 Errors

Table 11 — Error Codes for *GetTelephonyIdentity()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
606	Action not Authorized	The CP does not have privileges to invoke this action.
714	Identity does not exist	The identity of the TS has not been assigned yet by the telephony service in the WAN side.

5.5.2 *GetMessagingCapabilities()*

This action allows a TelCP to discover the set of capabilities and features that are supported by the *Messaging* service.

The *Messaging* service can support a subset of the messaging features. For example, the *Messaging* service can support only a subset of the Message classes or a subset of Session classes for a Session Mode Messaging. Therefore, a TelCP can discover the features supported by the *Messaging* service by invoking the *GetMessagingCapabilities()* action.

5.5.2.1 Arguments

Table 12 — Arguments for *GetMessagingCapabilities()*

Argument	Direction	relatedStateVariable
SupportedCapabilities	*OUT*	*A_ARG_TYPE_MessagingCapabilities*

5.5.2.2 Argument Descriptions

The output argument *SupportedCapabilitites* contains the features supported by the *Messaging* service. The argument follows the XML structure as defined in 5.3.2 *A_ARG_TYPE_MessagingCapabilities*.

5.5.2.3 Service Requirements

None.

5.5.2.4 Control Point Requirements When Calling The Action

None.

5.5.2.5 Dependency on Device State

None.

5.5.2.6 Effect on Device State

None.

5.5.2.7 Errors

Table 13 — Error Codes for *GetMessagingCapabilities()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
600-699	TBD	See Control clause in [1].

5.5.3 *GetNewMessages()*

This action allows a TelCP to read the value of the *NewMessages* state variable.This action may be used by a TelCP which does not subscribe to the *Messaging* service for the event notifications.

5.5.3.1 Arguments

Table 14 — Arguments for *GetNewMessages()*

Argument	Direction	relatedStateVariable
NewMessages	*OUT*	*NewMessages*

5.5.3.2 Argument Descriptions

The output argument *NewMessages* contains the value of the *NewMessages* state variable. The value of the *NewMessages* state variable includes information about the incoming Messages that have not been read yet.

If there are no new Messages then the output argument will include an XML fragment of type *NewMessages* without any NewMessage element.

5.5.3.3 Service Requirements

None.

5.5.3.4 Control Point Requirements When Calling The Action

None.

5.5.3.5 Dependency on Device State

None.

5.5.3.6 Effect on Device State

None.

5.5.3.7 Errors

Table 15 — Error Codes for *GetNewMessages*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
606	Action not Autohrized	The CP does not have privileges to invoke this action.

5.5.4 *SearchMessages()*

This action allows a TeICP to search for a list of Messages that satisfy the search criteria specified by the input arguments of this action. Based on the search criteria specified in the input arguments, this action can perform the following functions:

- Retrieve a complete list of Messages for a given Message class (e.g., E-Mail, SMS, MMS, Instant Message, etc.) and in a specific Message folder (e.g., Received, Outgoing, Sent, etc.).
- Retrieve a complete list of unread (or read) Messages of a given Message class (e.g., E-Mail, SMS, MMS, Instant Message, etc.), and in a specific Message folder (e.g., Received, Outgoing, Sent, etc.).
- Retrieve a complete list of Messages exchanged during a particular Session.

5.5.4.1 Arguments

Table 16 — Arguments for *SearchMessages()*

Argument	Direction	relatedStateVariable
MessageClass	*IN*	*A_ARG_TYPE_MessageClass*
MessageFolder	*IN*	*A_ARG_TYPE_MessageFolder*
MessageStatus	*IN*	*A_ARG_TYPE_MessageStatus*
SessionID	*IN*	*A_ARG_TYPE_SessionID*
MessageList	*OUT*	*A_ARG_TYPE_MessageList*

5.5.4.2 Argument Descriptions

The input argument *MessageClass* specifies the class of Messages to be searched. If the value of this argument is the empty string (“”), then the *Messaging* service will search for all the classes of Messages.

The input argument *MessageFolder* specifies the folder to search for Messages (e.g., Received folder, Sent folder, any folder, etc.). If the value of this argument is the empty string (“”), then the *Messaging* service will search in all the folders of Messages.

The input argument *MessageStatus* specifies whether to search for read or unread Messages. If the value of this argument is the empty string (“”), then the *Messaging* service will search for both read and unread Messages.

The input argument *SessionID* identifies the Session. If the value of this argument is not an empty string the *Messaging* service will only search for Messages that belongs to the specific Session.

The output argument *MessageList* is an XML fragment as specified in 5.3.11 which will only contain the Messages that satisfy the criteria specified in the input arguments.

5.5.4.3 Service Requirements

The *Messaging* Service will respond to this action with the list of Messages in MessageList structure, the *Messaging* Service shall include only required elements of the Message in the MessageList structure. The required elements are specified in 5.3.11.

5.5.4.4 Control Point Requirements When Calling The Action

Since the service only return the required elements for the Messages, if a TeICP requires the complete details of the Messages, then it can use the *ReadMessage()*action with the Message ID in the input argument to retrieve the details of the Message.

5.5.4.5 Dependency on Device State

None.

5.5.4.6 Effect on Device State

None.

5.5.4.7 Errors

Table 17 — Error Codes for *SearchMessages()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
606	Action not Autohrized	The CP does not have privileges to invoke this action.
702	Invalid Class	The Message class is invalid.
705	Invalid Message status	The Message status is invalid.
706	Invalid Session id	The Session with the given id does not exist.
707	Invalid folder	The folder is invalid.

5.5.5 *ReadMessage()*

This action allows a TeICP to retrieve a Message as identified by MessageID. This action also allows to retrieve a specific Message from a stored Session as indentified by the Session ID.

If the requested Message does not exists in the *Messaging* service, then *Messaging* service responds with an error.

5.5.5.1 Arguments

Table 18 — Arguments for *ReadMessage()*

Argument	Direction	relatedStateVariable
MessageID	*IN*	*A_ARG_TYPE_MessageID*
MessageRequested	*OUT*	*A_ARG_TYPE_Message*

5.5.5.2 Argument Descriptions

The input argument *MessageID* uniquely identifies the Message. The MessageID also includes the SessionID in the case of Session Mode Messaging. This allows a TelCP to retrieve a specific Message from a Session identified by the SessionID.

The output argument *MessageRequested* is an XML fragment as specified in 5.3.10 which contains all the details of the requested Message.

5.5.5.3 Service Requirements

None.

5.5.5.4 Control Point Requirements When Calling The Action

None.

5.5.5.5 Dependency on Device State

None.

5.5.5.6 Effect on Device State

When a *ReadMessage()* action is successful in the case of unread Messages, the value of the status of the Message changes to "Read".

5.5.5.7 Errors

Table 19 — Error Codes for *ReadMessage()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
606	Action not Autohrized	The CP does not have privileges to invoke this action.
701	Invalid Message ID	The Message with the given ID does not exist.

5.5.6 *SendMessage()*

This action allows a TelCP to send a Message. This action applies to both page and Session mode Messaging. This action returns immediately once invoked without waiting for the *Messaging* service to successfully sends the Message to the remote user as specified by the TelCP.

5.5.6.1 Arguments

Table 20 — Arguments for *SendMessage()*

Argument	Direction	relatedStateVariable
MessageToSend	*IN*	*A_ARG_TYPE_Message*
MessageID	*OUT*	*A_ARG_TYPE_MessageID*

5.5.6.2 Argument Descriptions

The input argument *MessageToSend* includes all the details of the Message including the recipients, the subject of the Message, the body of the Message, the attachments, etc. as specified in 5.3.10.

The output argument *MessageID* includes the unique identifier for the Message as created by the *Messaging* service. This unique identifier can be used to reference the Message.

5.5.6.3 Service Requirements

Once this action is invoked, then the *Messaging* service stores the submitted Message into the *Outgoing* folder. If the *Messaging* service successfully sends the Message to the remote party, then the Message is moved from the *Outgoing* folder to the *Sent* folder. The Message will remain in the *Outgoing* folder until it is successfully sent to the remote party.

The *Messaging* service shall generate a unique Message ID for the Message if the action is successful.

If the class element of the Message is set to the empty string, then the *Messaging* service shall decide whether to send the Message as a SMS, or MMS, or email or Instant Message by examining the recipients list, attachments and length of the Message, etc.

The *Messaging* service sets the value of the status of the Message as “Unread” in the *Outgoing* folder. After the Message is sent successfully to the remote party the Message is moved to the *Sent* folder and the status of the Message is set as “Read” in the *Sent* folder.

5.5.6.4 Control Point Requirements When Calling The Action

The *MessageToSend* input argument shall include all the mandatory details for the requested Message class as defined in 5.3.10.

The Message ID, the Message Folder, and the Message Status elements are managed by the *Messaging* service so the TelCP shall provide an empty string in the Message structure of the Message.

In the case of Session Mode Messaging, the *MessageClass* element in the Message structure of the *MessageToSend* input argument shall match with the current Session class as identified by the SessionID. If the TelCP wants to send the Message with a different Message Class, then the one specified Session class, then the TelCP first should change the Session class to “Mixed” by the *ModifySession()* action before invoking this action.

5.5.6.5 Dependency on Device State

None.

5.5.6.6 Effect on Device State

None.

5.5.6.7 Errors

Table 21 — Error Codes for *SendMessage()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
606	Action not Autohrized	The CP does not have privileges to invoke this action.
703	Invalid Message	The input Message is invalid.

5.5.7 *DeleteMessage()*

This action allows a TelCP to delete a Message as identified by Message ID.

If the *Deleted* folder is supported by the *Messaging* service, then the deleted Message is moved to the *Deleted* folder, otherwise the Message will be permanently removed. The request for deleting a Message from the *Deleted* folder will also remove the Message permanently.

5.5.7.1 Arguments

Table 22 — Arguments for *DeleteMessage()*

Argument	Direction	relatedStateVariable
MessageID	*IN*	*A_ARG_TYPE_MessageID*

5.5.7.2 Argument Descriptions

The input argument *MessageID* uniquely identifies the Message to be deleted.

5.5.7.3 Service Requirements

According to the Message lifecycle state diagram defined in 5.6.1.1, the execution of this action has the following effects.

- If the *Deleted* folder is supported by the *Messaging* service, then the deleted Message is moved to the *Deleted* folder, otherwise the Message will be permanently removed.
- The request for deleting the Message from the *Deleted* folder will also remove the Message permanently.
- A deletion of the Message does not change the status of the Message.
-

5.5.7.4 Control Point Requirements When Calling The Action

None.

5.5.7.5 Dependency on Device State

None.

5.5.7.6 Effect on Device State

None.

5.5.7.7 Errors

Table 23 — Error Codes for *DeleteMessage()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
606	Action not Autohrized	The CP does not have privileges to invoke this action.
701	Invalid Message ID	The Message with the given ID does not exist.

5.5.8 *CreateSession()*

This action allows a TelCP to create a new Messaging Session with a specific Session Class and a set of contacts.

If successful the *Messaging* service returns a unique identifier for the Messaging Session.

5.5.8.1 Arguments

Table 24 — Arguments for *CreateSession()*

Argument	Direction	relatedStateVariable
SessionClass	*IN*	*A_ARG_TYPE_SessionClass*
SessionRecipients	*IN*	*A_ARG_TYPE_RecipientsList*
Subject	*IN*	*A_ARG_TYPE_Subject*
SupportedContentType	*IN*	*A_ARG_TYPE_SupportedContentType*
SessionID	*OUT*	*A_ARG_TYPE_SessionID*

5.5.8.2 Argument Descriptions

The input argument *SessionClass* allows a TelCP to specify the class of the Session (e.g., SMS, MMS, Chat, etc.).

The input argument *SessionRecipients* includes the list of recipients for the Messaging Session.

The input arguments *Subject* and *SupportedContentType* allows a TelCP to specify the title of the Session and the intended content types for the Session. These arguments are only used for the Session Mode Messaging.

The output argument *SessionID* uniquely identifies the Messaging Session as assigned by the *Messaging* service.

5.5.8.3 Service Requirements

In the case of a valid request with SessionClass as *Chat/FileTransfer,* the *Messaging* service will create a new SessionID and send the Session invitations to the recipients. The *Messaging* service keeps the association between the local Session ID (Session ID of the LAN Messaing Session) and the global Session ID (the Session identifier on the WAN side) in order to correctly route the incoming and outgoing Messages.

If the Session Class value is the empty string then *Messaging* service shall decide the Session class based on the first Message exchange within the Messaging Session.

5.5.8.4 Control Point Requirements When Calling The Action

None.

5.5.8.5 Dependency on Device State

None.

5.5.8.6 Effect on Device State

None.

5.5.8.7 Errors

Table 25 — Error Codes for *CreateSession()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
606	Action not Autohrized	The CP does not have privileges to invoke this action.
704	Invalid recipients	One or more recipients are invalid.
710	Invalid Session class	The Session class is invalid.

5.5.9 *ModifySession()*

This action allows a TelCP to modify a Messaging Session which includes adding or removing contacts, or changing the subject, or the class, or the supported content types of the Session.

5.5.9.1 Arguments

Table 26 — Arguments for *ModifySession()*

Argument	Direction	relatedStateVariable
SessionID	*IN*	*A_ARG_TYPE_SessionID*
SessionRecipientsToAdd	*IN*	*A_ARG_TYPE_RecipientsList*
SessionRecipientsToRemove	*IN*	*A_ARG_TYPE_RecipientsList*
Subject	*IN*	*A_ARG_TYPE_Subject*
SupportedContentType	*IN*	*A_ARG_TYPE_SupportedContentType*
SessionClass	*IN*	*A_ARG_TYPE_SessionClass*

5.5.9.2 Argument Descriptions

The input argument *SessionID* includes the unique identifier of the Session to modify.

The input arguments *SessionRecipientsToAdd* and *SessionRecipientsToRemove* contain the list of recipients to add or remove within the current Session.

The input arguments *Subject* and *SupportedContentType* allow the TelCP to modify the main topic and the types of contents that can be shared among the participants within the Session.

The input argument *SessionClass* allows the TelCP to modify the class of the Session.

5.5.9.3 Service Requirements

If the input arguments contain valid values, the *Messaging* service shall modify only the changed parameter of the Session.

A Session can only change its initial class to "*Mixed*" since the Session will contain different classes of Messages (e.g., if an "*SMS*" Session is modified by a TelCP into an "*MMS*" Session, the resulting status of that Session is turned into "*Mixed*").

The class of a "*Mixed*" Session can not be modified.

5.5.9.4 Control Point Requirements When Calling The Action

A TelCP needs to fill the input parameters it doesn't want to modify with the empty string.

5.5.9.5 Dependency on Device State

None.

5.5.9.6 Effect on Device State

None.

5.5.9.7 Errors

Table 27 — Error Codes for *ModifySession()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
606	Action not Autohrized	The CP does not have privileges to invoke this action.
704	Invalid recipients	One or more recipients are invalid.
706	Invalid Session ID	The Session with the given ID does not exist.

5.5.10 *AcceptSession()*

This action allows a TelCP to accept a request for a new Messaging Session. When a TelCP receives the event for an incoming Session request, the TelCP can accept the incoming Session by invoking this action.

5.5.10.1 Arguments

Table 28 — Arguments for *AcceptSession()*

Argument	Direction	relatedStateVariable
SessionID	*IN*	*A_ARG_TYPE_SessionID*

5.5.10.2 Argument Descriptions

The input argument *SessionID* includes the unique identifier of the Session that the TelCP wants to accept.

5.5.10.3 Service Requirements

If the *SessionID* is valid, the *Messaging* service will accept the Session.

Only a Session that is currently "*Pending*" can be accepted by a TelCP. If the request is successfully accepted, the Session status shall be changed to "*Running*".

5.5.10.4 Control Point Requirements When Calling The Action

None.

5.5.10.5 Dependency on Device State

None.

5.5.10.6 Effect on Device State

None.

5.5.10.7 Errors

Table 29 — Error Codes for *AcceptSession()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
606	Action not Autohrized	The CP does not have privileges to invoke this action.
706	Invalid Session ID	The Session with the given ID does not exist.

5.5.11 *GetSessionUpdates()*

This action allows a TelCP to read the value of the *SessionUpdates* state variable, that is used to provide notifications of changes to the status of the *Messaging* Sessions.

This action can be used by a TelCP that doesn't subscribe for notifications from the *Messaging* service.

5.5.11.1 Arguments

Table 30 — Arguments for *GetSessionUpdates()*

Argument	Direction	relatedStateVariable
SessionUpdates	*OUT*	*SessionUpdates*

5.5.11.2 Argument Descriptions

The output argument *SessionUpdates* contains the value of the *SessionUpdates* state variable. The value of the *SessionUpdates* state variable includes information about active Sessions.

5.5.11.3 Service Requirements

None.

5.5.11.4 Control Point Requirements When Calling The Action

None.

5.5.11.5 Dependency on Device State

None.

5.5.11.6 Effect on Device State

None.

5.5.11.7 Errors

Table 31 — Error Codes for *GetSessionUpdates()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
606	Action not Autohrized	The CP does not have privileges to invoke this action.

5.5.12 *GetSessions()*

This action allows a TelCP to retrieve the Messaging Sessions available on the *Messaging* service either identified by the Session ID or filtered by the *SessionClass* and by the *SessionStatus* input arguments.

5.5.12.1 Arguments

Table 32 — Arguments for *GetSessions()*

Argument	Direction	relatedStateVariable
SessionID	*IN*	*A_ARG_TYPE_SessionID*
SessionClass	*IN*	*A_ARG_TYPE_SessionClass*
SessionStatus	*IN*	*A_ARG_TYPE_SessionStatus*
SessionsList	*OUT*	*A_ARG_TYPE_SessionsList*

5.5.12.2 Argument Descriptions

The input argument *SessionID* includes the unique identifier of the Session that the TelCP wants to retrieve. If the value of this argument is the empty string (“”), then the *Messaging* service will retrieve all the Sessions filtered by *SessionClass* and *SessionStatus*. If a valid Session id is specified the other input arguments are irrelevant.

The input argument *SessionClass* specifies the class of Sessions to be retrieved. If the value of this argument is the empty string (“”), then the *Messaging* service will retrieve information about Sessions for all classes.

The input argument *SessionStatus* specifies the value of the status of the Sessions to be retrieved. If the value of this argument is the empty string (“”), then the *Messaging* service will retrieve information about all the Sessions regardless of their status.

The Sessions that match the specified criteria are listed in the *SessionsList* output argument.

5.5.12.3 Service Requirements

When responding to this action, in the resulting list of Sessions the *Messaging* service will provide at least those details of Sessions that are defined as required XML elements in the related state variable section. However, additional optional elements can be included by the *Messaging* service in order to add further details.

5.5.12.4 Control Point Requirements When Calling The Action

None.

5.5.12.5 Dependency on Device State

None.

5.5.12.6 Effect on Device State

None.

5.5.12.7 Errors

Table 33 — Error Codes for *GetSessions()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
606	Action not Authorized	The CP does not have privileges to invoke this action.
706	Invalid Session ID	The Session with the given ID does not exist.
710	Invalid Session class	The Session class is invalid.
711	Invalid Session status	The Session status is invalid.

5.5.13 *JoinSession()*

This action allows a TelCP to join a “*Parked*” or a “*Running*” Session.

5.5.13.1 Arguments

Table 34 — Arguments for *JoinSession()*

Argument	Direction	relatedStateVariable
SessionID	*IN*	*A_ARG_TYPE_SessionID*

5.5.13.2 Argument Descriptions

The input argument *SessionID* includes the unique identifier of the Session that the TelCP wants to join.

5.5.13.3 Service Requirements

This action can be successfully invoked only on "*Parked*" or "*Running*" Sessions. When invoked on a Session with a status other than "*Parked*" or "*Running*" the action will fail with an appropriate error code.

If the request is successfully accepted by the *Messaging* service on a "*Parked*" Session, the status of the Session shall change to "*Running*".

5.5.13.4 Control Point Requirements When Calling The Action

None.

5.5.13.5 Dependency on Device State

None.

5.5.13.6 Effect on Device State

None.

5.5.13.7 Errors

Table 35 — Error Codes for *JoinSession()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
606	Action not Autohrized	The CP does not have privileges to invoke this action.
706	Invalid Session ID	The Session with the given ID does not exist.
708	Invalid action	The action is invalid in the current context.

5.5.14 *LeaveSession()*

This action allows a TelCP to leave a running Session without closing the Session.

5.5.14.1 Arguments

Table 36 — Arguments for *LeaveSession()*

Argument	Direction	relatedStateVariable
SessionID	*IN*	*A_ARG_TYPE_SessionID*

5.5.14.2 Argument Descriptions

The input argument *SessionID* includes the unique identifier of the Session that the TelCP wants to leave.

5.5.14.3 Service Requirements

Upon receiving this request, the *Messaging* service keeps locally the Session context, without leaving the corresponding remote Session. This allows another TelCP, or the same TelCP, to join the Session later.

This action can be successfully invoked only on "*Running*" Sessions. When invoked on a Session with a different status this action will fail with an appropriate error code.

Once the TelCP leaves the Session the status of the Session is changed to "*Parked*" if no other TelCPs are active in the Session.

5.5.14.4 Control Point Requirements When Calling The Action

None.

5.5.14.5 Dependency on Device State

None.

5.5.14.6 Effect on Device State

None.

5.5.14.7 Errors

Table 37 — Error Codes for *LeaveSession()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
606	Action not Autohrized	The CP does not have privileges to invoke this action.
706	Invalid Session ID	The Session with the given ID does not exist.
708	Invalid action	The action is invalid in the current context.

5.5.15 *CloseSession()*

This action allows a TelCP to close a Messaging Session. If this action is invoked on a "*Pending*" Session then the Session request is refused and the Session is closed.

When the Session is closed the Session is moved to the "*ClosedSession*" folder. It is left to the implementation of the service how long the closed Sessions are kept in the "*ClosedSession*" folder.

5.5.15.1 Arguments

Table 38 — Arguments for *CloseSession()*

Argument	Direction	relatedStateVariable
SessionID	*IN*	*A_ARG_TYPE_SessionID*

5.5.15.2 Argument Descriptions

The input argument *SessionID* includes the unique identifier of the Session that the TelCP wants to close.

5.5.15.3 Service Requirements

If the action is successful then the value of the Session status shall is set to "Closed".

5.5.15.4 Control Point Requirements When Calling The Action

None.

5.5.15.5 Dependency on Device State

None.

5.5.15.6 Effect on Device State

None.

5.5.15.7 Errors

Table 39 — Error Codes for *CloseSession()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
606	Action not Autohrized	The CP does not have privileges to invoke this action.
706	Invalid Session ID	The Session with the given ID does not exist.

5.5.16 *StartFileTransfer()*

This action allows a TelCP to initiate a file transfer Session with a WAN side user.

5.5.16.1 Arguments

Table 40 — Arguments for *StartFileTransfer()*

Argument	Direction	relatedStateVariable
FileInfoList	*IN*	*A_ARG_TYPE_FileInfoList*

5.5.16.2 Argument Descriptions

The input argument *FileInfoList* includes the details of the files to transfer and also Session related information for initiating the file transfer Session with a WAN user. This input argument includes the information of the file such as the type, size, and link of the file.

The *FileInfo* argument also includes the notification type. Sometimes TelCP may require periodic notification about the progress of the transfer, or the completion of the file transfer. If the value of the XML element *ProgressNotif* is set to “1” then the TelCP will get continuous notification of the progress of the transfer. If the value of the XML element *ProgressNotif* is set to “0” then the TelCP will get the notification after the file transfer is completed.

The element *SessionID* uniquely identifies the file transfer Session.

5.5.16.3 Service Requirements

Once the *Messaging* service receives this action with valid arguments, then it will send the appropriate response to this action and also retrieve the file from the TelCP using the link provided in the *FileInfo* argument.

The *Messaging* service can retrieve the file or a chunk of the file and then it will establish the WAN file transfer Session, and also notify the TelCPs about the progress of the file transfer Session as requested in the initial request.

When the file is not available for fetching from the link then, the *Messaging* service will notify the TelCP with the status “file can not be downloaded”, with SessionUpdates state variable.

Once the file transfer Session is successfully established, the *Messaging* service will notify the file transfer progress as requested by the TelCP in the *StartFileTransfer()* action.

5.5.16.4 Control Point Requirements When Calling The Action

None.

5.5.16.5 Dependency on Device State

None.

5.5.16.6 Effect on Device State

None.

5.5.16.7 Errors

Table 41 — Error Codes for *StartFileTransfer()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
606	Action not Autohrized	The CP does not have privileges to invoke this action.
706	Invalid Session ID	The Session with the given ID does not exist.

5.5.17 *CancelFileTransfer()*

This action is used to cancel a file transfer Session. The TelCP can use this action to close an ongoing file transfer Session and to remove all the Session related information.

5.5.17.1 Arguments

Table 42 — Arguments for *CancelFileTransfer()*

Argument	Direction	relatedStateVariable
SessionID	*IN*	*A_ARG_TYPE_SessionID*

5.5.17.2 Argument Descriptions

This input argument *SessionID* contains the unique identifier of the ongoing file transfer Session.

5.5.17.3 Service Requirements

If successful, the *Messaging* service also cancels the ongoing file transfer Session in the WAN side and also notifies the file transfer cancellation by sending an event with the *SessionStatus* state variable.

5.5.17.4 Control Point Requirements When Calling The Action

This action should be invoked only on a “*File-Transfer*” or a “*Mixed*” Session.

5.5.17.5 Dependency on Device State

None.

5.5.17.6 Effect on Device State

None.

5.5.17.7 Errors

Table 43 — Error Codes for *CancelFileTransfer()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
606	Action not Autohrized	The CP does not have privileges to invoke this action.
706	Invalid Session ID	The Session with the given ID does not exist.
709	No active file transfer	No active file transfer is ongoing within the Session.

5.5.18 *GetFileTransferSession()*

This action is used to retrieve the metadata of closed file transfer Session stored in the *ClosedSession* folder.

5.5.18.1 Arguments

Table 44 — Arguments for *GetFileTransferSession()*

Argument	Direction	relatedStateVariable
SessionID	*IN*	*A_ARG_TYPE_SessionID*
FileInfoList	*OUT*	*A_ARG_TYPE_FileInfoList*

5.5.18.2 Argument Descriptions

The input argument *SessionID* contains the unique identifier of the file transfer Session to be retrieved.

The output argument *FileInfoList* contains the metadata of the retrieved file transfer Session.

5.5.18.3 Service Requirements

If successful, the *Messaging* service retrieves the file transfer Session metadata from the *ClosedSession* folder.

5.5.18.4 Control Point Requirements When Calling The Action

This action should be invoked only on a "*File-Transfer*" Sessions that have been closed.

5.5.18.5 Dependency on Device State

None.

5.5.18.6 Effect on Device State

None.

5.5.18.7 Errors

Table 45 — Error Codes for *GetFileTransferSession()*

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
606	Action not Autohrized	The CP does not have privileges to invoke this action.
706	Invalid Session ID	The Session with the given ID does not exist.
710	Invalid Session class	The Session class is invalid.

5.5.19 Error Code Summary

Table 46 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 46 — Error Code Summary

ErrorCode	errorDescription	Description
400-499	TBD	See Control clause in [1].
500-599	TBD	See Control clause in [1].
606	Action not Autohrized	The CP does not have privileges to invoke this action.
700		Reserved for future extensions.
701	Invalid Message ID	The Message with the given ID does not exist.
702	Invalid Message class	The Message class is invalid.
703	Invalid Message	The input Message is invalid.
704	Invalid recipients	One or more recipients are invalid.
705	Invalid Message status	The Message status is invalid.
706	Invalid Session ID	The Session with the given ID does not exist.
707	Invalid folder	The folder is invalid.
708	Invalid action	The action is invalid in the current context.
709	No active file transfer	No active file transfer is ongoing within the Session.
710	Invalid Session class	The Session class is invalid.
711	Invalid Session status	The Session status is invalid.
714	Identity does not exist	The identity of the TS has not been assigned yet by the telephony service in the WAN side.

Note: 800-899 Error Codes are not permitted for standard actions. See Control clause in [1] for more details.

5.6 Service Behavioral Model

5.6.1 State Diagrams

5.6.1.1 Message Lifecycle

Figure 2 describes the lifecycle of a Message. The Message is moved among the folders depending on invoked actions or some events.

a) When a new Message is received, the *Messaging* service stores the Message in the "*Received*" folder.

b) A Message stored in the "*Received*" folder can be deleted by a TelCP by invoking the DeleteMessage() action with the ID of the Message. After this action succeeded, the Message is moved into the "*Deleted*" folder.

c) A Message stored in the "*Deleted*" folder can be permanently removed by invoking the DeleteMessage() action with the ID of the Message.

d) A Message can be sent by invoking the SendMessage() with the details of the Message to send. After this action succeeded, the Message is moved into the "*Outgoing*" folder.

e) A Message stored in the "*Outgoing*" folder is moved into the "*Sent*" folder after the Message is sent successfully to the remote party.

f) A Message stored in the "*Outgoing*" or in the "*Sent*" folder can be deleted by a TelCP by invoking the DeleteMessage() action with the ID of the Message. After this action succeeded the Message is moved into the "*Deleted*" folder.

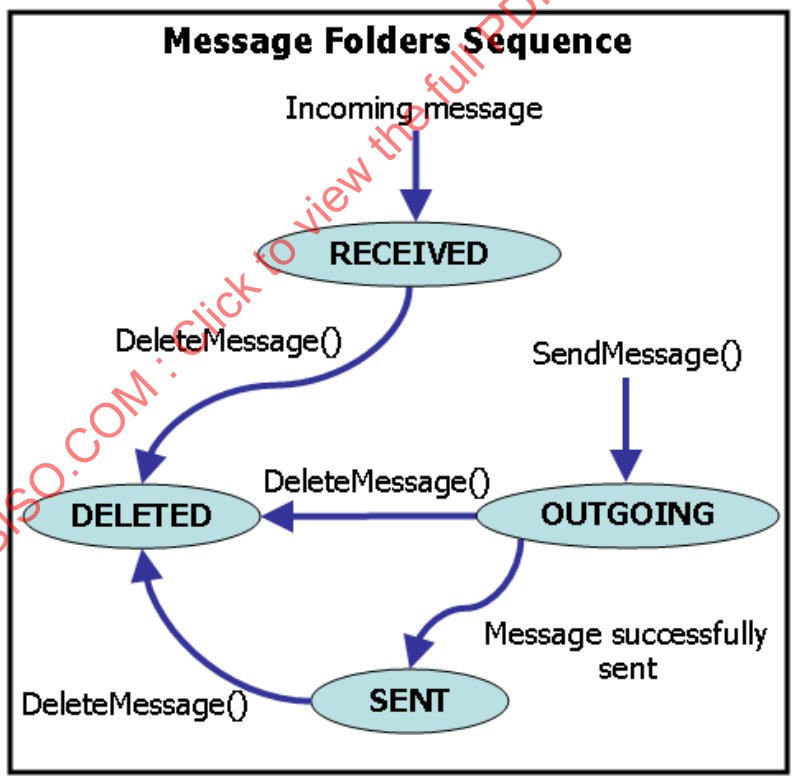

Figure 2 — Lifecycle of Messages

5.6.1.2 Session Lifecycle

Figure 3 describes the lifecycle of a Session.

a) A TelCP creates a Session by invoking the CreateSession() action. After this action is successful, the new Session is created and its status is initialized as "*Pending*".

b) When the *Messaging* service receives the Session request from the remote party, the *Messaging* service creates a new Session and its status is initialized as "*Pending*".

c) A TelCP accepts a Session by invoking the AcceptSession() action. After this action is successful, the status of the Session is updated to "*Running*".

d) A TelCP joins a Session ("*Parked*" or "*Running*") by invoking the JoinSession() action. After this action is successful, the status of the Session is updated to "*Running*".

e) A TelCP leaves a "*Running*" Session by invoking the LeaveSession() action. After this action is successful, the status of the Session is updated to "*Parked*" if no other TelCPs are participating in the Session.

f) A TelCP closes a Session by invoking the CloseSession() action. After this action is successful, the status of the Session is updated to "*Closed*".

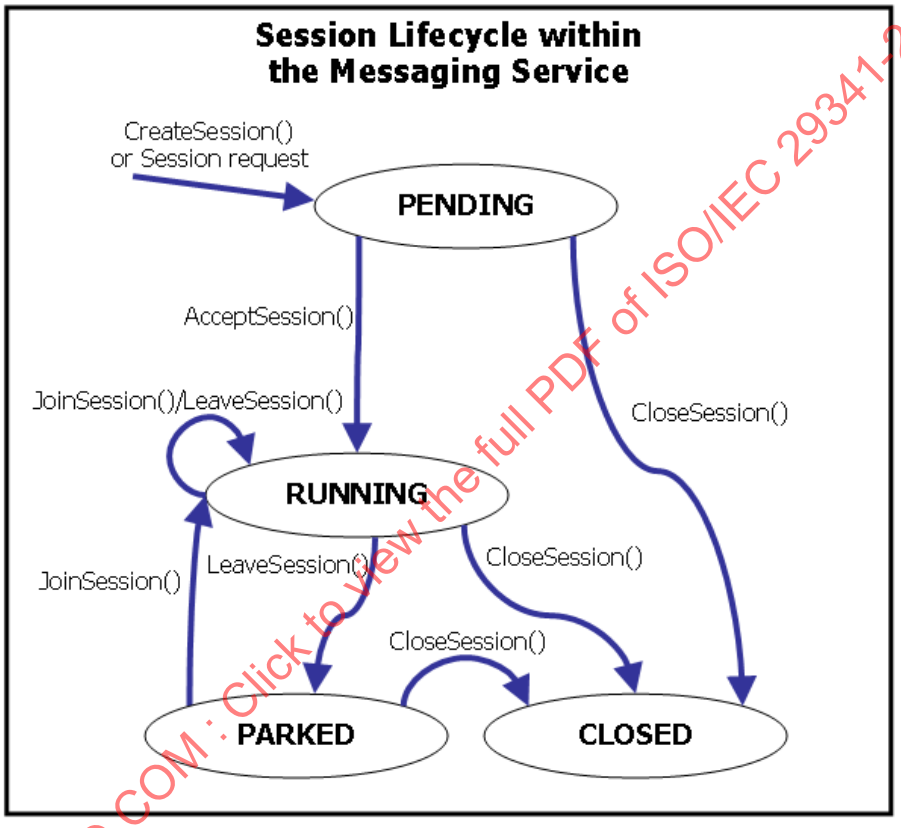

Figure 3 — Lifecycle of Sessions

6 XML Service Description

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

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

      <action>
         <name>GetTelephonyIdentity</name>
         <argumentList>
            <argument>
               <name>TelephonyIdentity</name>
               <direction>out</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_TelephonyServerIdentity
```

```
            </relatedStateVariable>
         </argument>
      </argumentList>
   </action>

   <action>
      <name>GetMessagingCapabilities</name>
      <argumentList>
         <argument>
            <name>SupportedCapabilities</name>
            <direction>out</direction>
            <relatedStateVariable>
               A_ARG_TYPE_MessagingCapabilities
            </relatedStateVariable>
         </argument>
      </argumentList>
   </action>

   <action>
      <name>GetNewMessages</name>
      <argumentList>
         <argument>
            <name>NewMessages</name>
            <direction>out</direction>
            <relatedStateVariable>
               NewMessages
            </relatedStateVariable>
         </argument>
      </argumentList>
   </action>

   <action>
      <name>SearchMessages</name>
      <argumentList>
         <argument>
            <name>MessageClass</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_MessageClass
            </relatedStateVariable>
         </argument>
         <argument>
            <name>MessageFolder</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_MessageFolder
            </relatedStateVariable>
         </argument>
         <argument>
            <name>MessageStatus</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_MessageStatus
            </relatedStateVariable>
         </argument>
         <argument>
            <name>SessionID</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_SessionID
            </relatedStateVariable>
         </argument>
         <argument>
            <name>MessageList</name>
            <direction>out</direction>
            <relatedStateVariable>
               A_ARG_TYPE_MessageList
            </relatedStateVariable>
```



```
            </argument>
        </argumentList>
    </action>

    <action>
        <name>ReadMessage</name>
        <argumentList>
            <argument>
                <name>MessageID</name>
                <direction>in</direction>
                <relatedStateVariable>
                    A_ARG_TYPE_MessageID
                </relatedStateVariable>
            </argument>
            <argument>
                <name>MessageRequested</name>
                <direction>out</direction>
                <relatedStateVariable>
                    A_ARG_TYPE_Message
                </relatedStateVariable>
            </argument>
        </argumentList>
    </action>

    <action>
        <name>SendMessage</name>
        <argumentList>
            <argument>
                <name>MessageToSend</name>
                <direction>in</direction>
                <relatedStateVariable>
                    A_ARG_TYPE_Message
                </relatedStateVariable>
            </argument>
            <argument>
                <name>MessageID</name>
                <direction>out</direction>
                <relatedStateVariable>
                    A_ARG_TYPE_MessageID
                </relatedStateVariable>
            </argument>
        </argumentList>
    </action>

    <action>
        <name>DeleteMessage</name>
        <argumentList>
            <argument>
                <name>MessageID</name>
                <direction>in</direction>
                <relatedStateVariable>
                    A_ARG_TYPE_MessageID
                </relatedStateVariable>
            </argument>
        </argumentList>
    </action>

    <action>
        <name>CreateSession</name>
        <argumentList>
            <argument>
                <name>SessionClass</name>
                <direction>in</direction>
                <relatedStateVariable>
                    A_ARG_TYPE_SessionClass
                </relatedStateVariable>
            </argument>
            <argument>
```

```
            <name>SessionRecipients</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_RecipientsList
            </relatedStateVariable>
         </argument>
         <argument>
            <name>Subject</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_String
            </relatedStateVariable>
         </argument>
         <argument>
            <name>SupportedContentType</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_SupportedContentType
            </relatedStateVariable>
         </argument>
         <argument>
            <name>SessionID</name>
            <direction>out</direction>
            <relatedStateVariable>
               A_ARG_TYPE_SessionID
            </relatedStateVariable>
         </argument>
      </argumentList>
   </action>

   <action>
      <name>ModifySession</name>
      <argumentList>
         <argument>
            <name>SessionID</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_SessionID
            </relatedStateVariable>
         </argument>
         <argument>
            <name>SessionRecipientsToAdd</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_RecipientsList
            </relatedStateVariable>
         </argument>
         <argument>
            <name>SessionRecipientsToRemove</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_RecipientsList
            </relatedStateVariable>
         </argument>
         <argument>
            <name>Subject</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_String
            </relatedStateVariable>
         </argument>
         <argument>
            <name>SupportedContentType</name>
            <direction>in</direction>
            <relatedStateVariable>
               A_ARG_TYPE_SupportedContentType
            </relatedStateVariable>
         </argument>
```



```
            <argument>
               <name>SessionClass</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_SessionClass
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>AcceptSession</name>
         <argumentList>
            <argument>
               <name>SessionID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_SessionID
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>GetSessionUpdates</name>
         <argumentList>
            <argument>
               <name>SessionUpdates</name>
               <direction>out</direction>
               <relatedStateVariable>
                  SessionUpdates
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>

      <action>
         <name>GetSessions</name>
         <argumentList>
            <argument>
               <name>SessionID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_SessionID
               </relatedStateVariable>
            </argument>
            <argument>
               <name>SessionClass</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_SessionClass
               </relatedStateVariable>
            </argument>
            <argument>
               <name>SessionStatus</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_SessionStatus
               </relatedStateVariable>
            </argument>
            <argument>
               <name>SessionsList</name>
               <direction>out</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_SessionsList
               </relatedStateVariable>
            </argument>
         </argumentList>
```

```
</action>

<action>
   <name>JoinSession</name>
   <argumentList>
      <argument>
         <name>SessionID</name>
         <direction>in</direction>
         <relatedStateVariable>
            A_ARG_TYPE_SessionID
         </relatedStateVariable>
      </argument>
   </argumentList>
</action>

<action>
   <name>LeaveSession</name>
   <argumentList>
      <argument>
         <name>SessionID</name>
         <direction>in</direction>
         <relatedStateVariable>
            A_ARG_TYPE_SessionID
         </relatedStateVariable>
      </argument>
   </argumentList>
</action>

<action>
   <name>CloseSession</name>
   <argumentList>
      <argument>
         <name>SessionID</name>
         <direction>in</direction>
         <relatedStateVariable>
            A_ARG_TYPE_SessionID
         </relatedStateVariable>
      </argument>
   </argumentList>
</action>

<action>
   <name>StartFileTransfer</name>
   <argumentList>
      <argument>
         <name>FileInfoList</name>
         <direction>in</direction>
         <relatedStateVariable>
            A_ARG_TYPE_FileInfoList
         </relatedStateVariable>
      </argument>
   </argumentList>
</action>

<action>
   <name>CancelFileTransfer</name>
   <argumentList>
      <argument>
         <name>SessionID</name>
         <direction>in</direction>
         <relatedStateVariable>
            A_ARG_TYPE_SessionID
         </relatedStateVariable>
      </argument>
   </argumentList>
</action>

<action>
```



```
            <name>GetFileTransferSession</name>
            <argumentList>
               <argument>
                  <name>SessionID</name>
                  <direction>in</direction>
                  <relatedStateVariable>
                     A_ARG_TYPE_SessionID
                  </relatedStateVariable>
               </argument>
               <argument>
                  <name>FileInfoList</name>
                  <direction>out</direction>
                  <relatedStateVariable>
                     A_ARG_TYPE_FileInfoList
                  </relatedStateVariable>
               </argument>
            </argumentList>
         </action>

      </actionList>

      <serviceStateTable>

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

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

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

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

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

         <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_MessageClass</name>
            <dataType>string</dataType>
            <allowedValueList>
               <allowedValue>e-Mail</allowedValue>
               <allowedValue>SMS</allowedValue>
               <allowedValue>MMS</allowedValue>
               <allowedValue>Instant Message</allowedValue>
               <allowedValue/>
            </allowedValueList>
         </stateVariable>

         <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_MessageFolder</name>
            <dataType>string</dataType>
            <allowedValueList>
               <allowedValue>Received</allowedValue>
               <allowedValue>Outgoing</allowedValue>
               <allowedValue>Sent</allowedValue>
               <allowedValue>Deleted</allowedValue>
```