WV-042 Client-Server Protocol Session and Transactions

Candidate Version 1.2 – 22 May 2004

Open Mobile Alliance
OMA-IMPS-WV-CSP-V1_2-20040522-C

Continues the Technical Activities
Originated in the Wireless Village Initiative

Contents

1. Scope

The Wireless Village Instant Messaging and Presence Service (IMPS) includes four primary features:

- Presence
- Instant Messaging
- Groups
- Shared Content

Presence is the key enabling technology for IMPS. It includes client device availability (my phone is on/off, in a call), user status (available, unavailable, in a meeting), location, client device capabilities (voice, text, GPRS, multimedia) and searchable personal statuses such as mood (happy, angry) and hobbies (football, fishing, computing, dancing). Since presence information is personal, it is only made available according to the user's wishes - access control features put the control of the user presence information in the users' hands.

Instant Messaging (IM) is a familiar concept in both the mobile and desktop worlds. Desktop IM clients, two-way SMS and two-way paging are all forms of Instant Messaging. Wireless Village IM will enable interoperable mobile IM in concert with other innovative features to provide an enhanced user experience.

Groups or chat are a fun and familiar concept on the Internet. Both operators and end-users are able to create and manage groups. Users can invite their friends and family to chat in group discussions. Operators can build common interest groups where end-users can meet each other online.

Shared Content allows users and operators to setup their own storage area where they can post pictures, music and other multimedia content while enabling the sharing with other individuals and groups in an IM or chat session.

These features, taken in part or as a whole, provide the basis for innovative new services that build upon a common interoperable framework.

2. References

2.1. Normative References

[CREQ] "Specification of WAP Conformance Requirements". Open Mobile Alliance™. WAP-221-CREQ. URL:http://www1.wapforum.org/tech/terms.asp?doc=WAP-221-CREQ-20010425-a.pdf

[CSP DTD] "WV-043 Client-Server Protocol DTD and Examples Version 1.2". Open Mobile Alliance. May 2004. URL: http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-CSP_DTD-V1_2-20040522-C.pdf

[CSP SCR] "WV-048 Client-Server Protocol Static Conformance Requirement Version 1.2". Open Mobile Alliance. May 2004. URL: http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-CSP_SCR-V1_2-20040522-C.pdf

[CSP Trans] "WV-044 Client-Server Protocol Transport Bindings Version 1.2". Open Mobile Alliance. May 2004. URL: http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-CSP_Transport-V1_2-20040522-C.pdf

[E.164] ITU-T Recommendation E.164 (05/97) The international public telecommunication numbering plan URL:http://www.itu.int/rec/recommendation.asp?type=items&lang=e&parent=T-REC-E.164-199705-I

[FIPS 180-1] "Secure Hash Standard", April 1995 URL:http://csrc.nist.gov/publications/fips/fips180-1/fip180-1.pdf

[IANA] Character sets registered at IANA (MIBenum assignments) URL:http://www.iana.org/assignments/character-sets

[PA] "WV-049 Presence Attributes Version 1.2". Open Mobile Alliance. May 2004. URL: http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-PA-V1_2-20040522-C.pdf

[RFC1321] "The MD5 Message-Digest Algorithm", April 1992. URL:http://www.ietf.org/rfc/rfc1321.txt?number=1321

[RFC2045] "Multipurpose Internet Mail Extensions (MIME) Part one: Format of Internet Message Bodies". Section 6.8 "Base64 Content-Transfer-Encoding". URL:http://www.ietf.org/rfc/rfc2045.txt?number=2045

[RFC2046] Borenstein N., and N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part Two: Media Types", November 1996. URL:http://www.ietf.org/rfc/rfc2046.txt?number=2046

[RFC2119] "Keywords for using RFCs to Indicate Requirements levels", Bradner, S. URL:http://www.ietf.org/rfc/rfc2119.txt?number=2119

[RFC2234] "Augmented BNF for Syntax Specifications: ABNF". D. Crocker, Ed., P. Overell. November 1997. URL:http://www.ietf.org/rfc/rfc2234.txt

[RFC2396] Uniform Resource Identifiers (URI): Generic Syntax URL:http://www.ietf.org/rfc/rfc2396.txt?number=2396

[RFC2426] vCard MIME Directory Profile URL:http://www.ietf.org/rfc/rfc2426.txt?number=2426

[RFC2445] Internet Calendaring and Scheduling Core Object Specification (iCalendar) URL:http://www.ietf.org/rfc/rfc2445.txt?number=2445

[RFC2778] "A Model for Presence and Instant Messaging", February 2000 URL:http://www.ietf.org/rfc/rfc2778.txt?number=2778

[RFC822] "Standard for the Format of ARPA Internet Text Messages", August 1982. URL:http://www.ietf.org/rfc/rfc0822.txt?number=822

[SSP SCR] "WV-055 SSP – Server-Server Protocol Static Conformance Requirement Version 1.2". Open Mobile Alliance. December 2002. URL:http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-

SSP_SCR-V1_2-20030117-D.pdf

[TS23040] 3rd Generation Partnership Project; Technical Specification Group Terminals; Technical Realization of the Short Message Service (SMS) (Release 5), 3GPP TS 23.040 v5.4.0", June 2002 URL:ftp://ftp.3gpp.org/Specs/archive/23_series/23.040/23040-540.zip

[TS23140] 3rd Generation Partnership Project; Technical Specification Group Terminals; Multimedia Messaging Service (MMS); Functional Description; Stage 2 (Release 5), 3GPP TS 23.140 v5.4.0", September 2002 URL:ftp://ftp.3gpp.org/specs/archive/23_series/23.140/23140-540.zip

[UUID] Steven Miller, "DEC/HP Network Computing Architecture Remote Procedure Call Run-Time Extension Specification Version OSF TX1.0.11", July 23, 1992

[VCAL10] "vCalendar - The Electronic Calendaring and Scheduling Format", version 1.0, The Internet Mail Consortium (IMC), September 18, 1996, URL:http://www.imc.org/pdi/vcal-10.doc

[VCARD21] "vCard - The Electronic Business Card", version 2.1,The Internet Mail Consortium (IMC), September 18, 1996, URL:http://www.imc.org/pdi/vcard-21.doc

[WAPMMS] "Wireless Application Protocol - MMS Encapsulation Protocol, Version 01-June-2001", URL:http://www1.wapforum.org/tech/terms.asp?doc=WAP-209-MMSEncapsulation-20010601-a.pdf

[XML] "Extensible Markup Language 1.0 (Second Edition)", W3C recommendation, 6-October-2000 URL:http://www.w3.org/TR/2000/REC-xml-20001006.pdf

2.2. Informative References

[Arch] "WV-040 System Architecture Model Version 1.2". Open Mobile Alliance. May 2004. URL: http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-Arch-V1_2-20040522-C.pdf

[FeaFun] "WV-041 Features and Functions Version 1.2". Open Mobile Alliance. May 2004. URL: http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-Features_Functions-V1_2-20040522-C.pdf

[CSP] "WV-042 Client-Server Protocol Session and Transactions Version 1.2". Open Mobile Alliance. May 2004. URL: http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-CSP-V1_2-20040522-C.pdf

[CSP DTD] "WV-043 Client-Server Protocol DTD and Examples Version 1.2". Open Mobile Alliance. May 2004. URL: http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-CSP_DTD-V1_2-20040522-C.pdf

[CSP Trans] "WV-044 Client-Server Protocol Transport Bindings Version 1.2". Open Mobile Alliance. May 2004. URL: http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-CSP_Transport-V1_2-20040522-C.pdf

[CSP DataType] "WV-045 Client-Server Protocol Data Types Version 1.2". Open Mobile Alliance. May 2004. URL: http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-CSP_DataTypes-V1_2-20040522-C.pdf

[CSP SMS] "WV-046 Client-Server Protocol SMS Binding Version 1.2". Open Mobile Alliance. May 2004. URL: http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-CSP_SMS-V1_2-20040522-C.pdf

[CSP WBXML] "WV-047 Client-Server Protocol Binary Definition and Examples Version 1.2". Open Mobile Alliance. May 2004.

URL: http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-CSP_WBXML-V1_2-20040522-C.pdf

[CSP SCR] "WV-048 Client-Server Protocol Static Conformance Requirement Version 1.2". Open Mobile Alliance. May 2004.
URL: http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-CSP_SCR-V1_2-20040522-C.pdf

[PA] "WV-049 Presence Attributes Version 1.2". Open Mobile Alliance. May 2004.
URL:http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-PA-V1_2-20040522-C.pdf

[PA DTD] "WV-050 Presence Attribute DTD and Examples Version 1.2". Open Mobile Alliance. May 2004.
URL: http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-PA_DTD-V1_2-20040522-C.pdf

[CLP] "WV-051 Command Line Protocol Version 1.2". Open Mobile Alliance. May 2004.
URL: http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-CLP-V1_2-20040522-C.pdf

[SSP] "WV-052 SSP - Server-Server Protocol Semantics Document Version 1.2". Open Mobile Alliance. May 2004.
URL: http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-SSP-V1_2-20040522-C.pdf

[SSP Syntax] "WV-053 Server-Server Protocol XML Syntax Document Version 1.2". Open Mobile Alliance. May 2004.
URL: http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-SSP_Syntax-V1_2-20040522-C.pdf

[SSP Trans] "WV-054 SSP - Transport Binding Version 1.2". Open Mobile Alliance. May 2004.
URL: http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-SSP_Transport-V1_2-20040522-C.pdf

[SSP SCR] "WV-055 SSP – Server-Server Protocol Static Conformance Requirement Version 1.2". Open Mobile Alliance. May 2004.
URL: http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-SSP_SCR-V1_2-20040522-C.pdf

[WAPARCH] “WAP Architecture, Version 12-July-2001”. Open Mobile Alliance™. WAP-210-WAPArch.
URL:http://www1.wapforum.org/tech/terms.asp?doc=WAP-210-WAPArch-20010712-a.pdf

3. Terminology and Conventions

3.1. Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2. Definitions

None.

3.3. Abbreviations

CSP	Client-Server Protocol
SAP	Service Access Points
WV	Wireless Village

4. Introduction

This document describes the transactions and the information elements in transactions that are used to provide Wireless Village services and the interoperability of different implementations.

5. WV session

5.1. Session management

The Wireless Village session is a framework in which the WV services are provided to the WV client. The WV session is transport-independent. If the transport connection is broken, the client can reconnect and continue the session. The client device MAY be even turned on and off and the session MAY still be continued.

The WV session is established when the client logs in and terminated when either the client logs out or the SAP decides to disconnect the session. The session is identified by a *Session-ID*. In addition, the WV client provides at login phase a client-unique *session cookie* for the session, which is used by the server to trigger communications in some cases.

The client and the server MUST use the same version of the WV protocol that was used in the login primitive in every transaction throughout the whole session. The versions supported on the server side MAY be retrieved any time – it is independent from WV sessions. Please refer to chapter 5.2 Version Management and 6.3 Version Discovery Transaction for further information of this mechanism.

If the server for some reason does not accept the WV version that the client used in LoginRequest, the client MAY initiate version discovery transaction to retrieve the list of supported versions from the server, and select a WV version (to be used in the login phase again) that is supported on both client and server side.

The authentication of the user is done at the login phase. The authentication is, in general, considered to be valid throughout the session. However, the server MAY, at any time disconnect the session and request the client to re-login. In this case, the client provides the old session-ID. After valid authentication, the WV server MAY accept to *re-establish* the old session.

During the WV session, both the client and SAP have to maintain *session context*. The session context contains dynamic information of the services the client is currently using. The WV specifications do not explicitly define what is the actual state of services the session context contains, but it assumes that the negotiated services for the session are valid throughout the session.

The session context in the WV server and the WV client are tied together by the services the user is currently using (subscribed presence attributes, joined groups, authorized presence attributes, etc). The WV server and WV client MAY assume that the link between the contexts is valid throughout the WV session as well as reestablished session.

Associated with each session is an auto logout timer value. The client can request a particular value or request an infinite time-to-live time. The server can set any timer value and the client MUST obey that. Server implementers are however advised not to select a lower time-to-live value then the one requested by the client.

When the session is terminated and a *new* session is established, the WV server MAY still maintain some services, such as subscribed presence attributes. This is an implementation issue in the WV server.

5.2. Version Management

Each set of the Wireless Village specifications defines a version of the Wireless Village Client-Server protocol. XML, and WBXML based bindings use specific namespaces to identify which version of the protocol is being used for each of the following specific purposes:

- Session management;
- Transaction management (see 4.3); and
- Presence Attributes as defined in [PA]

Sessions are established using matched sets of namespaces defined in a specific version of the Wireless Village specifications. When a backward compatible minor version of the protocol is defined, the specifications MAY indicate that the Transaction management and Presence Attributes namespaces defined in the minor version MAY be used in conjunction with the Session management namespace from the associated major version.

SMS binding does not use namespaces to identify different versions of the WV specifications: it uses only two digits in the preamble. Thus the version discovery is very much simplified in SMS binding.

The client maintains the same WV version (and hence namespaces) that was used in the login transaction throughout the whole duration of the session.

The Version Discovery transaction provides a mechanism by which a client can discover the protocol versions implemented by a specific server. The mechanism is described in details in chapter 6.3 Version Discovery Transaction.

5.3. Addressing

5.3.1. Introduction

The Wireless Village addressing model introduces an unique WV address space. The definition of the addresses is based on the URI format [RFC2396]. The addressable entities are:

- User
- Contact list
- User group (private and public)
- Content (private and public)

Use of other address spaces MAY be used to interoperate with other systems, but their use is out of the scope of Wireless Village specifications.

In addition to the user, the WV client the user is using MAY be addressed as well. In this specification version, the client identification is defined but its exact semantics and use cases are left for next WV specification release.

5.3.2. Generic address format

The generic address syntax is based on URI [RFC2396]. The wv-schema in the URI indicates the use of wv-addressing space. The generic syntax is defined as follows:

```
Address = ["wv:"] [User-ID] ["/" Resource] ["@" Domain]
Resource  = Group-ID | Contact-List-ID | Content-ID
Domain    = sub-domain *(“.” sub-domain)
```

where User-ID refers to the identification of the WV user, Domain identifies the WV server domain, and Resource further identifies the referred public or private resource within the domain. The sub-domain is defined in [RFC822].

When the User-ID is present without the Resource, the address refers to the user. When User-ID is present with Resource, the address refers to the private resource of the user. When User-ID is not present, the Resource MUST always be present and then the address refers to a public resource within the domain.

The Domain part is OPTIONAL. When it is not present, the address refers implicitly to the home domain.

The schema part is OPTIONAL. When it is not present the default schema “wv:” is assumed.

The addresses are case insensitive.

5.3.3. Address encoding

As per URI [RFC2396], certain reserved characters MUST be escaped if they occur within the User-ID, Resource, or Domain portions of a Wireless Village address. This includes the characters “;”, “?”, “:”, “&”, “=”, “+”, “$” and “,”. For example, a valid Wireless Village address for the user “$mith” in the “server.com” domain is:

```
wv:%24mith@server.com
```

Certain characters are not allowed in the User-ID portion of Wireless Village addresses (see 4.2.4, below). This includes the characters "/", "@", "+", " " and TAB. This restriction is independent of the encoding of a User-ID within a Wireless Village address. For example, this Wireless Village address is not permissible:

```
wv:john%40aol.com@server.com
```

This address is not permissible because after URI-decoding, the User-ID portion contains a forbidden character ("@"). If a server's internal representation of a username permits the occurrence of forbidden characters, such characters MUST be double-escaped when they occur in a Wireless Village address, such that they do not occur unescaped in the User-ID portion after URI-decoding, or they MUST be escaped via some other scheme that does not employ forbidden characters.

5.3.4. User addressing

CSP uses User-IDs to uniquely identify any WV User. The User-ID is syntactically equivalent to an e-mail address, and as such is subject to the same restrictions for character set, as described in "Standard for the Format of ARPA Internet Text Messages" [RFC822]. The User ID is either a local User ID which is the domain that the client is logged on (home-domain) or an external User ID, which is on another domain.

The User-ID either refers to the Internet-type address or to a mobile number of the user. If the User-ID refers to the mobile number of the user, the user name always starts either with a digit or with a '+' sign. A user name referring to Internet-type address MAY not start with a '+' sign or digit.

The syntax of the User-ID is defined as follows:

```
User-ID     = Mobile-Identity | Internet-Identity
Internet-Identity           = *alpha
Mobile-Identity             = (digit | "+") *digit
digit       = "0" | "1" | "2" |"3" |"4" |"5" |"6" |"7" |"8" | "9"
alpha       = Any non-control ASCII character (decimal 32 – 126, inclusive) except specials
specials    = "/" | "@" | "+" | " " | TAB
```

When the User-ID refers to the mobile number address, the User-ID preceded with '+' sign refers to the international numbering in The International Public Telecommunication Numbering Plan [E.164]. Without '+' sign, it refers to the national numbering in the [E.164].

Examples:

```
Local-User-ID:          wv:yuriyt
                        wv:+1234567890
                        wv:4567890
External-User-ID:       wv:Jon.Smith@imps.com
                        wv:+1234567890@imps.com
                        wv:4567890@imps.com
```

The users MAY also be identified by screen names, nicknames and aliases. These identifiers explicitly and implicitly refer to the User-ID.

ScreenName – the combination of a name a user chooses in a group session, and the Group-ID itself. The user MAY have different ScreenNames on different occasions as well as on different groups. The ScreenName is always connected to a group.

NickName – A name that is used internally in a client to hide the UserID of contacts. When a ContactList is stored on the server the NickName MUST have a space. It is not possible to address a NickName.

Alias – The name a user suggest others to use as NickName. Part of the User Presence.

5.3.5. Contact List Addressing

The CSP uses Contact List IDs to uniquely identify any contact list of any user. The contact list ID is based on the generic address syntax. The contact list MAY be a public contact list ora private contact list. The syntax is defined as follows:

Contact-List-ID = *alpha

Examples for the contact list IDs are:

```
wv:john/colleagues@imps.com
wv:/managers
wv:john/friends
wv:/managers@imps.com
```

5.3.6. User Group

The CSP uses a Group ID to uniquely identify a group. The Group-ID is based on the generic address syntax. The user group MAY be public user group or private user group. The syntax of the group ID is defined as follows:

Group-ID = *alpha

Examples of the Group-IDs are:

```
wv:john/mygroup@imps.com
ww:john/mygroup
wv:/technicalforum
wv:/technical_forum@imps.com
```

5.3.7. Content Addressing

The CSP uses a Content ID to uniquely identify a content. The Content ID is based on the generic address syntax. The syntax of the Content ID is as follows:

Content-ID = *alpha

Examples of the Content IDs are:

```
wv:john/WV_presentation
wv:john/WV_presentation@imps.com
wv:/wvspec
wv:/wvspec@imps.com
```

5.3.8. Client Addressing

The client-ID is a unique identifier of the WV client that the user is currently using. It identifies the WV client as an application and its location that accesses the WV services. The client-ID is intended to allow:

- Multiple access from the same user
- Direct application-to-application communication.

The Client-ID consists of

- An OPTIONAL application identifier as a URL identifying the application and its location.
- An OPTIONAL mobile device identity (such as international mobile number defined in [E.164]).

The semantics and use cases for the Client-ID are left for future versions of WV specifications.

5.4. Transaction management

A WV transaction is a basic communication mechanism between a WV client and a WV SAP. A transaction consists of a request and a response primitive usually. (Exceptions: disconnect, 4-way login.) The purpose of the transaction is to exchange data between the entities or request an operation: usually both within the same transaction. The transactions MAY originate from either WV client or WV SAP.

The transaction consists of request message and response message. Initiator of the transaction SHOULD always expect the Status primitive as the result of transaction even if it is not specified explicitly in the description of transaction. This behaviour is used to notify the initiator about error caused by the request. The only exception is the Disconnect request from server.

The initiating entity, the WV client or the SAP, allocates a transaction identifier that the responding entity returns in the response message. This links together the requesting message and response message. The originator of the transaction is responsible for maintaining the uniqueness of the transaction identifiers in a particular direction (C->S or S->C) within a session.

The response to a request message SHOULD be received within 20 seconds from the initiation of the transaction. During or after that period, the requesting entity MAY *resend* the request message using the same transaction identifier. The responding entity SHOULD guarantee that the requested operation or data is carried out only once, even if multiple request messages with the same transaction identifier are received.

The transactions are sequential until the capability negotiation is completed.

Sequential means that one transaction MUST be complete (closed) before a next one is started (open). A transaction is considered as closed when a the final response primitive has been received, a time out waiting for a response primitive has occurred, or the underlying transport has been detected as “broken.”

Multiple transactions mean that two or more transactions are open during the same time. The transactions can be done using the same transport connection or on separate transport connections.

6. Fundamental Primitives

6.1. Status Primitive

If an error occurs in the processing party while processing a transaction, it SHALL respond to the other party with a Status primitive instead of the expected response primitive. The Status primitive is also used as the expected response primitive in some successful transactions.

The Result structure always contains one of the status Codes specified in this document. It MAY also contain a Description string and if necessary, a Detailed description explaining the error. The DetailedResult can be used with every error code but has its main use when the error concerns a specific MessageID, UserID, GroupID, ScreenName or Domain. An example is when sending a message to several recipients and one UserID is invalid. The Result could then have status code 201 (Partially successful) and the DetailedResult naming the bad UserID together with status code 531 (Unknown user).

The status code 201 (Partially successful) is used when only part of the request was successfully processed. In this case the DetailedResult with all errors MUST be included but the result of successful parts of the transaction MAY be omitted. An example is the creation of a contact list with initial contacts. If the server creates the contact list, but fails to add one or more of the contacts, the 201 (Partially successful) with DetailedResult for the bad contacts SHOULD be returned. If the server cannot create the contact list at all there was no partial success and the Result code MAY be for example 701 (Contact list already exists).

When no part of the transaction was successfully processed but the error cannot be described by only one status code, the code 900 (Multiple errors) is used. In this case DetailedResult with all errors MUST be included. By definition there can be no successful parts and that is the difference from status code 201 (Partially successful).

If the request is malformed and cannot be processed, the error code 400 (Bad Request) is used.

Information Element	Req	Type	Description
Message-Type	M	Status	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	O	String	Session ID for session.
Client-ID	O	Structure	Identifies the requesting IM client. Unique (for this user) identifier.
Result	M	Structure	The result of the transaction.

Table 1. Information elements in Status primitive

6.2. PollingRequest Primitive

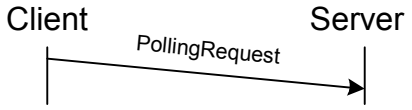

Figure 1. Polling request

The polling request is used in the context of the server pushing data to the Client. When a client has received CIR from a CIR channel it needs to send the PollingRequest to the server to enable the server to insert the server initiated transaction in the reply. The exact cases for the use of poll messages are elaborated in the transport-binding document.

The polling request follows the basic message structure, but it carries no transaction-ID. Thus, the transaction-ID SHALL be empty.

Information Element	Req	Type	Description
Message-Type	M	PollingRequest	Message identifier.
Transaction-ID	M	String	Empty content, since Transaction-

			ID is not carried.
Session-ID	M	String	Session ID for session.

Table 2. Information elements in PollingRequest primitive

6.3. Version Discovery Transaction

Figure 2. Version Discovery

From time to time, it MAY be necessary for a client to evaluate the best protocol version to use in sessions with a particular server. This is achieved using the Version Discovery transaction, which determines the protocol versions that are mutually acceptable to both the client and server.

The client initiates the Version Discovery transaction by sending a VersionDiscoveryRequest to the intended server.

When the server receives the VersionDiscoveryRequest, it examines the client's proposed versions (if any).

- If the client proposes different versions, the server selects the version that it implements. It includes this version in the VersionDiscoveryResponse. If the server supports none of the namespaces that the client proposed in a particular set, then the server returns an empty result.
- If the client proposes no versions, then the server includes all of the versions.

If the server knows of other servers that better support the namespaces proposed by the client, then the server MAY return the URLs (MSISDNs) of those servers to the client.

Servers that only implement version 1.1 of the Wireless Village specifications return an error indication when they received a VersionDiscoveryRequest. Clients that receive such an error response MAY choose to assume that the server only supports version 1.1 protocols.

The Version Discovery transaction does not follow the basic message structure; it is a dedicated transaction that is meant to be consistent with all future versions of the WV protocol. All servers MUST implement this transaction, while it is OPTIONAL for the clients.

Primitive	Direction
VersionDiscoveryRequest	Client → Server
VersionDiscoveryResponse	Client ← Server

Table 3. Primitive directions for Version Discovery

Information Element	Req	Type	Description
Version-List	O	Structure	List of versions supported by the client.

Table 4. Information elements in VersionDiscoveryRequest primitive

Information Element	Req	Type	Description
Version-List	C	Structure	List of versions implemented by the

			server. If the Version-List was present in the request, only those versions are included in this list that are also implemented by the server.
Other-Servers	O	Structure	List of access points of servers that better support the namespaces proposed by the client.

Table 5. Information elements in VersionDiscoveryResponse primitive

6.4. Logging in

6.4.1. Transactions

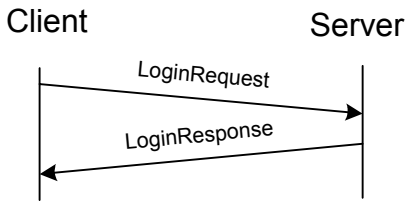

Figure 3. Logging in

In order to use the IM services the user MUST log in into a Service Access Point.

After the server processed this request, it sends a login response message to the client; which will contain the details of the login operation. When the login operation into the Service Access Point is not successful, a status message indicates the login failure instead of the login response message. The login response sent by the server MAY also indicate that the client needs to perform a Client Capability Request.

After a successful login a service negotiation performed. After the service negotiation the agreed services available to the user are: Presence Service, IM Service, Group Service, Content Service.

When the session is re-initiated, client capability negotiation is not performed.

The client MAY choose either a two-way access control or a four-way access control. If the client chooses the two-way access control, the LoginRequest contains the element “Password-String” with password in plain text. The server responds with either success or failure or further authorization. If the client chooses the four-way access control, the LoginRequest contains neither element “Password-String” nor element “Digest-Bytes”. Instead, the LoginRequest contains the element “Supported Digest Schema”. The server responds with the challenge “nonce” based on the “digest schema”.

The server can choose not to use any authentication scheme; instead rely on authentication in the mobile network. In this case the Digest-Schema element indicates “PWD”, and the Nonce element is not present in the LoginResponse primitive.

The client then sends the LoginRequest again with the element “Digest-Bytes” which is the BASE64-encoded result string based on the “schema” hash function on the concatenation of the challenge “nonce” and the password:

Digest-Bytes = hash_function(nonce + password)

The server finally responds with either “success” or “failure”.

Even if the client chooses the two-way access control, the server still can send a response with error code 401. That means the server requires further authorization of this request. In this case, the response message contains the available authorization scheme “digest schema” with the challenge “nonce” for the scheme.

Following schemes can be used as “digest schema” to generate the challenge “nonce”:

The MD5 Scheme

The client concatenates the challenge with the password, and performs a MD5 hash on the resulting string. The client then SHOULD repeat a request with the resulting data as a string encoded by BASE64.

The MD4 & MD6 Scheme

The same algorithm as for the MD5 Scheme except the hash algorithm is replaced with MD4 or MD6.

The SHA Scheme

The client concatenates the challenge with the password, and performs a SHA hash on the resulting string. The client then SHOULD repeat a request with the resulting data as a string encoded by BASE64.

6.4.2. Error conditions

Generic error conditions:

- Service unavailable. (503)
- Version not supported. (505)

LoginRequest error conditions:

- Further authorization needed to use the server. (401)
- Invalid password. (409)
- The particular user is not allowed to use the server. (403)
- Unknown user. (531)
- Already logged in. (603)
- SessionID, UserID and ClientID not matching. (422)
- No matching digest scheme supported (543)

6.4.3. Primitives and information elements

Primitive	Direction
LoginRequest	Client → Server
LoginResponse	Client ← Server

Table 6. Primitive directions for Logging in

Information Element	Req	Type	Description
Message-Type	M	LoginRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction, set by client.
User-ID	M	String	Identifies the requesting User.
Client-ID	M	Structure	Identifies the requesting WV client. Unique (for this user) identifier.
Password-String	C	String	The password digest corresponding to the User-ID
Digest-Bytes	C	String	The digest is BASE64 encoded.
Supported-Digest-Schema	C	String	A list of supported digest schema (PWD, SHA, MD4, MD5, MD6)
Session-ID	C	String	The session-ID when session reestablishment is requested.
Time-To-Live	O	Integer	Time requested between client to server messages before client is considered disconnect. If information element is not

			present client is requesting an infinite time-to-live time. Indicated in seconds.
Session-Cookie	M	String	The session cookie used by WV SAP to initiate communications within the session (max length 50)

Table 7. Information elements in LoginRequest primitive

Information Element	Req	Type	Description
Message-Type	M	LoginResponse	Message identifier.
Transaction-ID	M	String	Identifies the request transaction.
Client-ID	M	Structure	Identifies the requesting WV client. Unique (for this user) identifier.
Result	M	Structure	Result of the login request.
Nonce	C	String	Random string generated by server for password digest. The string is not BASE64 encoded.
Digest-Schema	C	String	Type of digest schema to use.
Session-ID	C	String	Session-ID. String generated by the server to identify this session. Session-ID is supplied with all following requests to the server. Present only if login was successful.
Keep-Alive-Time	C	Integer	Auto logout timer value in seconds. The server can set any timer value and the client MUST obey that. Each message transaction resets the Keep-Alive-Time timer. Present only if login was successful.
Client-Capability-Request	C	Boolean	Informs the Client that it needs to perform a Client Capability Request transaction. Present only if login was successful.

Table 8. Information elements in LoginResponse primitive

6.5. Logging out and Disconnecting

6.5.1. Transactions

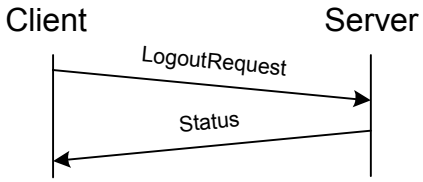

Figure 4. Logging Out

The user can log out from the WV services by using the LogoutRequest message. The server responds with a Status message.

If the user is an active member (e.g. joined) to one or more discussion groups when the logout request is issued, the server (the client if server-initiated disconnect) will automatically remove (leave group) the user from the discussion group.

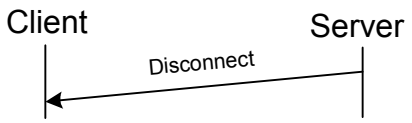

Figure 5. Server Initiated Disconnection

Whenever the server disconnects a client, it sends a Disconnect message to the client. The server SHOULD disconnect a client if the session Time-To-Live timer has been exceeded. The server MAY also disconnect the client for some other reason. The server sends the Disconnect message to the client containing the Session-ID, and the Result containing code and descriptive text. The transaction-ID is present in the primitive for compatibility reason only (with other primitives) – the client MUST ignore its content.

6.5.2. Error conditions

Generic error conditions:

- Service unavailable. (503)
- Not logged in. (604)

LogoutRequest error conditions:

- None except the generic error conditions.

Disconnect error conditions:

- Forced logout. (601)
- Session expired. (600)

6.5.3. Primitives and information elements

Primitive	Direction
LogoutRequest	Client → Server
Status	Client ← Server
Disconnect	Client ← Server

Table 9. Primitive directions for Logging Out

Information Element	Req	Type	Description
Message-Type	M	LogoutRequest	Message identifier
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Session ID for session.

Table 10. Information elements in LogoutRequest

Information Element	Req	Type	Description
Message-Type	M	Disconnect	Message identifier.
Transaction-ID	M	String	Identifies the transaction. Present if client initiated request.
Session-ID	M	String	Session ID for session.
Result	M	Structure	Indicates the code and description why the disconnection happened.

Table 11. Information elements in Disconnect primitive

6.6. Keep Alive

6.6.1. Transactions

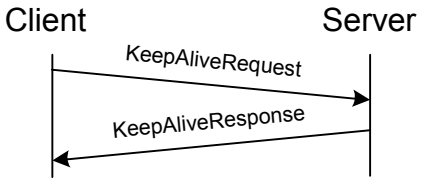

Figure 6. Keep alive transaction

KeepAliveRequest is sent by the client to reset the keep-alive timer when no other messages were sent. Optionally the client MAY apply for a new timeout value. The server responds with KeepAliveResponse message, which optionally contains a new timeout value if the timeout value requested by the client is not satisfactory.

Not only the KeepAliveRequest, but also any other messages from the same session reset the timer on the server side.

6.6.2. Error conditions

Generic error conditions:

- Service unavailable. (503)
- Not logged in. (604)

KeepAliveRequest error conditions:

- New timeout value not accepted – old value is in use. (605)

6.6.3. Primitives and information elements

Primitive	Direction
KeepAliveRequest	Client → Server
KeepAliveResponse	Client ← Server

Table 12. Primitive directions for keep alive transaction

Information Element	Req	Type	Description
Message-Type	M	KeepAliveRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Session ID for session.
Time-To-Live	O	Integer	Indicates the new time-to-live of the session in seconds.

Table 13. Information elements in KeepAliveRequest primitive

Information Element	Req	Type	Description

Message-Type	M	KeepAliveResponse	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Session ID for session.
Result	M	Structure	The result of the request.
Keep-Alive-Time	O	Integer	Indicates the new time-to-live of the session in seconds.

Table 14. Information elements in KeepAliveResponse primitive

6.7. Get Service Provider Info

6.7.1. Transactions

Figure 7. Get Service Provider Info transaction

The Get Service Provider information retrieves information about the Service Provider. The name of the provider as well as a multimedia message MAY be used as a splash screen or "about information", or link to a web/wap page that might contain more useful information. This transaction can be done without login in to the server.

6.7.2. Error conditions

Generic error conditions:

- Service not supported. (405)
- Service unavailable. (503)
- Service not agreed. (506)
- Not logged in. (604)

GetSPInfoRequest error conditions:

- ClientID not matching this user. (422)

6.7.3. Primitives and information elements

Primitive	Direction
GetSPInfoRequest	Client → Server
GetSPInfoResponse	Client ← Server

Table 15. Primitive directions for Get Service Provider Info

Information Element	Req	Type	Description
Message-Type	M	GetSPInfoRequest	Message identifier.

Transaction-ID	M	String	Identifies the transaction.
Session-ID	C	String	Session ID for session.
Client-ID	C	Structure	Identifies the requesting client.

Table 16. Information elements in Get Service Provider Info Request primitive

Information Element	Req	Type	Description
Message-Type	M	GetSPInfoResponse	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	C	String	Session ID for session.
Client-ID	C	Structure	Identifies the requesting client.
Name	M	String	Name of the service provider.
Logo	O	MMS	Service-provider specific image. (*e.g.*, logo)
Text	O	String	Descriptive text.
URL	O	String	Link to a web page.

Table 17. Information elements in Get Service Provider Info Response primitive

6.8. Service and Capability Request

6.8.1. Transactions

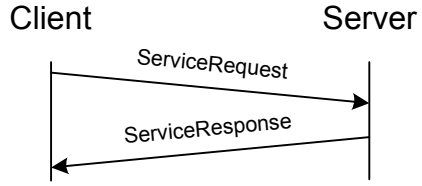

Figure 8. Service request

After successful login the client and server MUST set up the context of the session. Service negotiation MUST be done after the successful login transaction, and MAY be repeated during the session at any time.

The response to the client confirms what services the server supports and the client is allowed to use.

The goals of the service negotiation are:

- To discover the available services.

  In order to achieve this goal the client needs to know exactly which services are available on the server. If the client does not know which services are available on the server it can request it using the All-Function-Request information element ServiceRequest primitive. If requested, the ServiceResponse primitive will contain the service tree, which gives the information about the available services. Having the tree available the client can select which services it would like to use; it is ready to make an agreement.
- To agree what services will be used.

  The client and the server have to agree which features and functions will be used during the session. The client sends its service tree to the server in the Requested-Services element of the ServiceRequest primitive describing what kind of services it would like to use. The server processes the tree, and sends back the ServiceResponse primitive containing the inverted tree (meaning that the tree contains the features and functions the client has requested but not allowed to use). The client throughout the whole session cannot use any of the features and functions that the server did not agree to provide: all non-negotiated features and functions will fail. Non-

negotiated features and functions are the ones that were either not requested by the client, or not agreed by the server for some reason.

The ServiceRequest and ServiceResponse primitives carry the service tree. The service tree is a set of features and functions structured as a tree, WVCSPFeat being the root (which indicates all available features and functions). The tree MAY have multiple sub-trees (with multiple roots). There is no need to indicate the whole sub-tree when all features or functions are requested/denied under a specific sub-tree: only the sub-root element MUST be present in the tree. However, it is necessary to indicate the sub-tree when certain features (that are part of a sub-root element) are not requested.

Note that the service tree in SMS binding is handled differently.

The names of the elements are derived from the [CSP SCR] document.

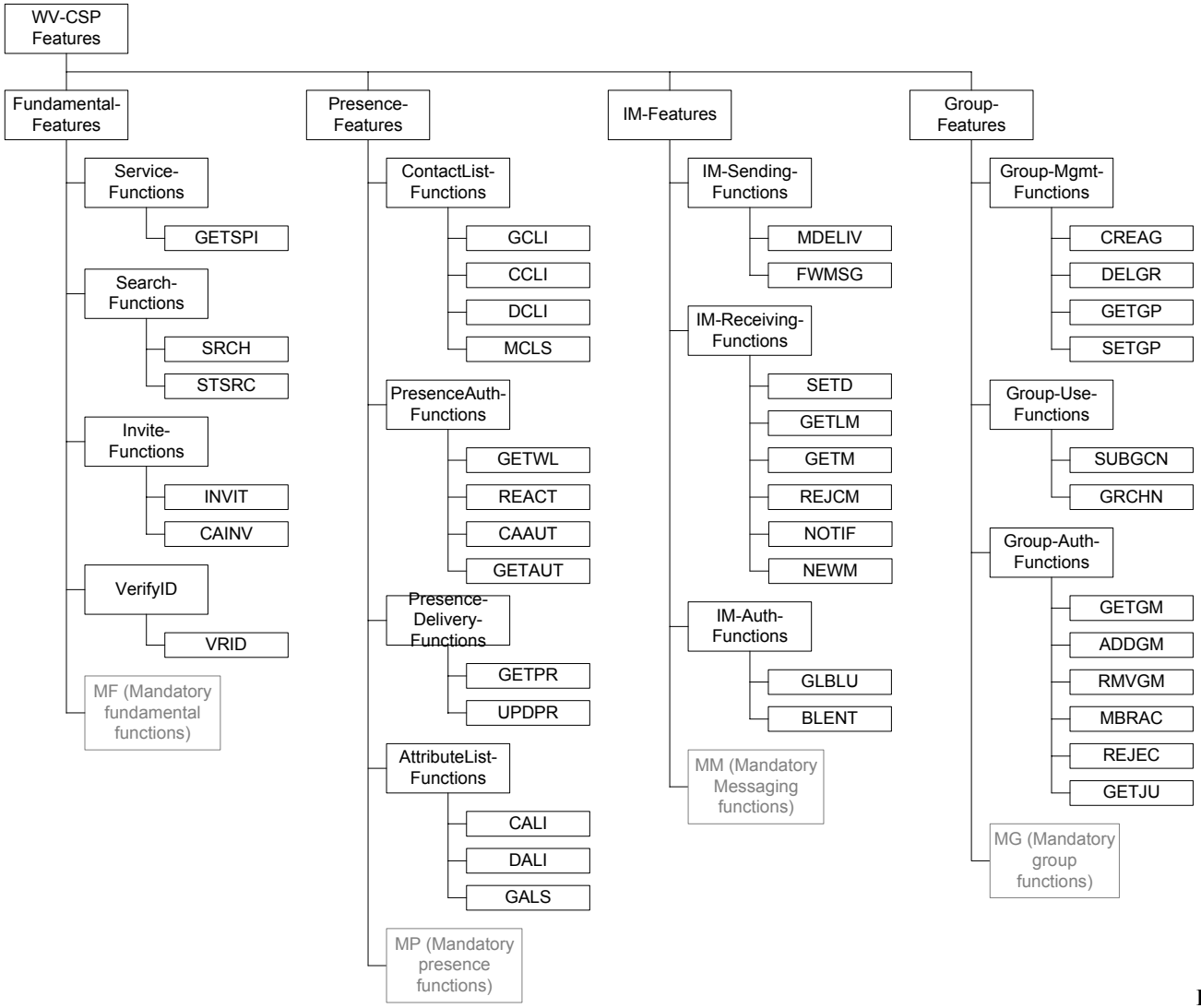

Figure 9. The service tree

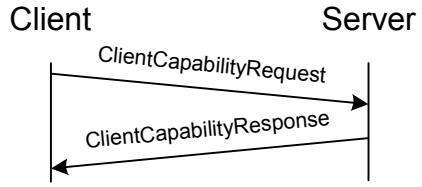

Figure 10. Client Capability Request

Client capability negotiation can be performed after the successful Login transaction. The server MUST maintain the client capability information during the session, and it MAY cache these capabilities between sessions- If the client capability negotiation is needed after login, the server MUST indicate it in the login response. The client capability negotiation can also be performed any time during a session.

The client capability request sets up the communication preferences for the session, for example the message delivery method.

The Client Capability Request contains an element "Requested Capabilities" that conveys the client capability information to the server. The client capability information includes:

- ClientType – the type of the client. See Table 7 in [PA].
- InitialDeliveryMethod – the initial IM delivery method that the recipient client prefers in the set of "PUSH" and "Notify/Get".
- AnyContent – A Boolean value indicating that the client accepts any content types.
- AcceptedContentType – the list of supported content types in the client device, such as "text/plain; charset=us-ascii". Applicable only when Any-Content is "No" or missing.
- AcceptedCharset – the list of supported character sets for plain text documents in the client device. Integer number assigned by IANA (see MIBenum numbers in [IANA]). Applicable only when Any-Content is "Yes".
- AcceptedTransferEncoding – the supported transfer encoding methods in the client device, such as "base64".
- AcceptedContentLength – the maximum content size when using "PUSH". Indicates the character (byte) count of the message content.
- SupportedBearer – the list of supported bearers (HTTP, WSP, SMS)
- MultiTrans – Integer value indicating the maximum number of primitives that the client can handle within the same transport message, as well as the maximum number of open transactions from both client and server side at any given time. The value MUST be greater than 0.
- ParserSize – the maximum character (byte) count of XML (WBXML, SMS - depending on the actual encoding) message size that the parser can handle. Multiple transactions in the same message and presence updates (many user in the same message) might generate large XML documents.
- SupportedCIRMethod – the list of supported CIR methods that are supported by the client.
- UDPPort – the client MAY indicate that it requests other than the default port for the standalone UDP/IP CIR method. It is a decimal integer number.
- ServerPollMin – integer value indicating the minimum time interval (in seconds) that MUST pass before two subsequent PollingRequest transactions.
- DefaultLanguage – The current language setting in the client. The language code is specifying that the client prefers to receive text information in the indicated language from the server. The

information is OPTIONAL – it is used to override the user profile/presence info language preference.

For the details of the "PUSH" and other message-related information, please refer to section 9. Instant Messaging Feature.

The Client Capability Response contains an element "Agreed Capabilities" that conveys the agreed capability information back to the client. The agreed capability information includes the following derived elements:

- SupportedBearer – the list of supported bearers (HTTP, WSP, SMS) that both the client and server support.
- SupportedCIRMethod – the list of CIR methods that both the client and server support.
- TCPAddress – If the client indicated that it supports STCP in the request, the server provides an IP address for standalone TCP/IP CIR method. It is an IP address.
- TCPPort – If the client indicated that it supports STCP in the request the server provides a port number if it is different from the default port for the standalone TCP/IP CIR method. Decimal integer number.
- ServerPollMin – integer value indicating the minimum time interval (in seconds) that MUST pass before two subsequent PollingRequest transactions.
- CIRHTTPAddress – A URL used for HTTP binding of CIR channel. See CSP Transport Binding specification for description on how to use this binding.

6.8.2. Error conditions

Generic error conditions:

- Service unavailable. (503)
- Not logged in. (604)

ServiceRequest error conditions:

- ClientID not matching this user. (422)

ClientCapabilityRequest error conditions:

- ClientID not matching this user. (422)

6.8.3. Primitives and information elements

Primitive	Direction
ServiceRequest	Client → Server
ServiceResponse	Client ← Server
ClientCapabilityRequest	Client → Server
ClientCapabilityResponse	Client ← Server

Table 18. Primitive directions for service request and capability request

Information Element	Req	Type	Description
Message-Type	M	ServiceRequest	Message identifier.
Transaction-ID	M	String	Transaction identifier.
Session-ID	M	String	Session ID for session.
Requested-Functions	C	Structure	Identifies the service elements and functions the client requests.

All-Functions-Request	M	Boolean	Request the server to send all services the it supports in the reply.

Table 19. Information elements in ServiceRequest primitive

Information Element	Req	Type	Description
Message-Type	M	ServiceResponse	Message identifier.
Transaction-ID	M	String	Transaction identifier.
Session-ID	M	String	Session ID for session.
Not-Available-Functions	C	Structure	Identifies the delta of the client requested and what is available for that user.
All-Functions	C	Structure	Identifies all of the functions that the server supports.

Table 20. Information elements in ServiceResponse primitive

Information Element	Req	Type	Description
Message-Type	M	CapabilityRequest	Message identifier.
Transaction-ID	M	String	Transaction identifier.
Session-ID	M	String	Session ID for session.
Requested-Capabilities	M	Structure	Identifies the capabilities requested by the client.

Table 21. Information elements in ClientCapabilityRequest primitive

Information Element	Req	Type	Description
Message-Type	M	CapabilityResponse	Message identifier.
Transaction-ID	M	String	Transaction identifier.
Session-ID	M	String	Session ID for session.
Agreed-Capabilities	M	Structure	Identifies the capabilities agreed by the server.

Table 22. Information elements in ClientCapabilityResponse primitive

7. Common features

7.1. General search transaction

7.1.1. Transactions

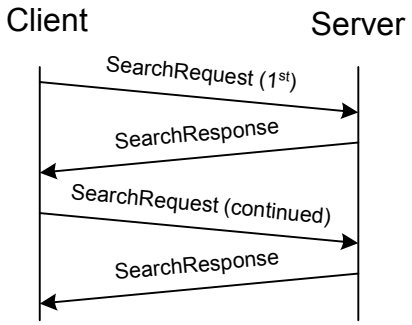

Figure 11. General search transactions

A user MAY search for users/groups based on various user/group properties. The user MAY limit the number of results to be retrieved at a time, and MAY continue the search and go through all the results.

The search is performed using a list of one or more Search-Pairs.

A Search-Pair consists of a Search-Element and a Search-String. The Search-Element indicates which property of the user/group SHALL be searched for the Search-String. When more than one search pairs are specified in the primitive, logical AND operation is assumed between the different pairs.

Every Search-Element MAY be present only once within the same search request.

The result of a user search is always user-ID. Similarly, the result of a group search is always group-ID.

Search-Element	Search-String
USER_ALIAS	The *Search-String* is a substring of a user's alias.
USER_ONLINE_STATUS	The Search-String is the online status value.
USER_EMAIL_ADDRESS	The *Search-String* is a substring of a user's e-mail address.
USER_FIRST_NAME	The *Search-String* is a substring of a user's firstname.
USER_ID	The *Search-String* is a substring of a user-ID.
USER_LAST_NAME	The *Search-String* is a substring of a user's lastname.
USER_MOBILE_NUMBER	The *Search-String* is a mobile number. [E.164]

Table 23. Search elements for user search

Search-Element	Search-String
GROUP_ID	The *Search-String* is a substring of a group-ID.
GROUP_NAME	The *Search-String* is a substring of a group's name (part of group properties).
GROUP_TOPIC	The *Search-String* is a substring of a group's topic (part of group properties).
GROUP_USER_ID_JOINED	The *Search-String* is a user-ID.
GROUP_USER_ID_OWNER	The *Search-String* is a user-ID. Search result contains the list of groups owned by the specified user.
GROUP_USER_ID_AUTOJOIN	The *Search-String* is a user-ID. Search result contains the list of groups that have AutoJoin property set to "T" for the

	specified user.

Table 24. Search elements for group search

The client sends the SearchRequest message to the server including the Search-Pair-List and the Search-Limit (maximum number of results at a time). The server responds with the SearchResponse message, which includes the Result of the search, and if the search was successful, it includes the Search-ID, the Search-Index (a continuation index to indicate where the search SHOULD be continued), the Search-Findings (the number of items found that match the criteria so far), and the Search-Results (the actual data).

The search result is restricted in the same way presence information is established when requested in other ways. If the searching user is not proactively authorized to see certain presence values for a user included in the search result, those presence values will not be included. If the unauthorized presence attribute is part of the search criteria, that user will not be included in the search result. This allows users that want to have certain presence attributes searchable to expose them through their default attribute list.

The user MAY continue the search. In this case the SearchRequest message includes only the Search-ID and the Search-Index. The server responds with the SearchResponse, but the message includes only the Result, the Search-Index, the Search-Findings and the Search-Results.

The client MAY modify the Search-Index value, so that the search MAY be continued at a different place. The Search-Index is valid until a new search is performed or the session ends (a previous search is invalidated when a new search is started).

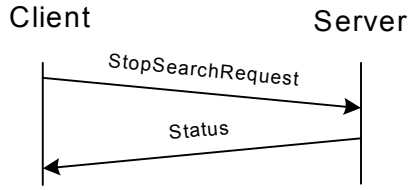

Figure 12. Stop search transaction

In order to indicate to the server that results are no further REQUIRED from a previously issued search request, the client sends StopSearchRequest message to the server containing the Search-ID. The server invalidates the indicated search, and replies with a Status message. The invalidated Search-ID cannot be used after invalidation.

7.1.2. Error conditions

Generic error conditions:

- Service not supported. (405)
- Service unavailable. (503)
- Service not agreed. (506)
- Not logged in. (604)

SearchRequest error conditions:

- Unable to parse criteria (Invalid Search-Element). (402)
- Initial search request was not sent (Invalid Search-ID). (424)
- Invalid Search-Index (out of range). (425)
- Search timeout (in case of continued search the subsequent request primitive is late). (535)
- Too many hits (536)
- Too broad search criteria (537)

StopSearchRequest error conditions:

- Initial search request was not sent (Invalid Search-ID). (424)

7.1.3. Primitives and information elements

Primitive	Direction
SearchRequest	Client → Server
SearchResponse	Client ← Server
StopSearchRequest	Client → Server
Status	Client ← Server

Table 25. Primitive directions for searching

Information Element	Req	Type	Description
Message-Type	M	SearchRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies the session.
Search-Pair-List	C	Structure	Searching criteria in terms of user or group properties. Present only in the 1st search request.
Search-Limit	C	Integer	Indicates the number of maximum search results can be received at a time. Present only in the 1st search request.
Search-ID	C	String	Uniquely identifies a search transaction. The server assigns this ID when the first search is performed, thus not present in the 1st search request.
Search-Index	C	Integer	Indicates that the results SHALL be sent starting from this particular index. Present only when the search is continued.

Table 26. Information elements in SearchRequest primitive

Information Element	Req	Type	Description
Message-Type	M	SearchResponse	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies the session.
Search-ID	C	String	Uniquely identifies a search transaction. The server assigns this ID when the 1st search is performed successfully.
Search-Findings	M	Integer	Indicates the number of the current findings.
Completion-Flag	M	Boolean	Indicates whether the client can expect new results. 'F' if server MAY provide new results (still searching), 'T' if new results will not be provided.
Search-Index	M	Integer	Indicates the particular index from which the next search can start. This provides the information to the user where to continue the search.
Search-Results	C	Structure	Search results.

Table 27. Information elements in SearchResponse primitive

Information Element	Req	Type	Description
Message-Type	M	StopSearchRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies the session.
Search-ID	M	String	Identifies the search to be invalidated.

Table 28. Information elements in StopSearchRequest primitive

7.2. Invitations

7.2.1. Transactions

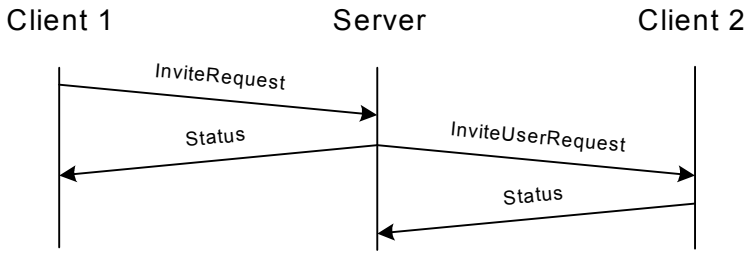

Figure 13. Invite user(s)

A user MAY invite other user(s) to join a group, to exchange messages, to share presence values list, and to share content and to request to be added to a group's member list. The client sends the InviteRequest message to the server containing the ID of the invitation, the type of the invitation, the ID of the subject, the list of user(s) to be invited specified by user-IDs or screen-names, and optionally the reason for the invitation (a short text), and his/her own screen-name. When requesting group membership, the Recipients element identifies the group ID.

The server responds with a Status message, and sends InviteUserRequest message to every user who has been invited. The InviteUserRequest message contains the ID of the invitation, the ID or the screen-name (if it was present in the request) of the inviter, the subject and the reason for the invitation (if requested). Each invited client replies with a Status message.

In case of group membership invitations, the server sends the InviteUserRequest to the administrators and moderators of the group.

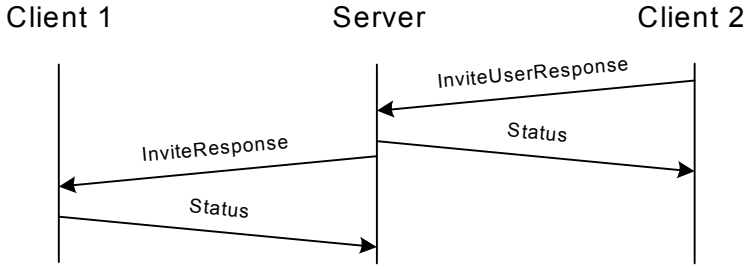

Figure 14. Invited users' response

The invitee MAY accept or reject the invitation. The invitee's client responds with the InviteUserResponse message to the server containing the ID of the invitation, the acceptance indicator, the ID of the subject, and optionally a short response text and the responding user's screen-name. When replying to a membership invitation, only the group ID is used to identify the sender. The server responds with a Status message.

The server will send the InviteResponse message to the inviter containing the ID of the invitation, the ID of the invitee (user-ID or screen-name), the acceptance indicator, the ID of the subject and (if the invitee specified it) the short response text. The inviter's client responds with a Status message.

While there is no mandatory requirement for an invited user to act according to the acceptance indicator of his/her response in the scope of this function, the invited user SHOULD act according to the response of the invited user.

The subject of the invitation MAY be a group, messaging, a shared content, presence or group membership. In case of presence the user MAY include a list of presence attributes that he/she is willing to share with the other party. Note that actual presence attribute sharing is not done; the transaction is only informational. Similarly, in case of group, messaging, or shared content invitations the actual action is not taken, the user MAY do it manually (the invitation is only informational).

7.2.2. Error conditions

Generic error conditions:

- Service not supported. (405)
- Service unavailable. (503)
- Service not agreed. (506)
- Not logged in. (604)

InviteRequest error conditions:

- Invalid invitation type. (402)
- Invalid invite-ID. (423)
- Delivery to recipient not available. (410)
- Delivery to recipient domain not available. (516)
- Recipient unknown (UserID or screen-name). (531)
- Recipient unknown (Contact list). (700)
- Invalid or unsupported presence value. (751)
- Group does not exist. (800)

InviteResponse error conditions:

- Client MAY ignore any error and respond with Successful. (200)

InviteUserRequest error conditions:

- Client MAY ignore any error and respond with Successful. (200)

InviteUserResponse error conditions:

- Invalid acceptance type. (402)
- Invalid invite-ID. (423)

7.2.3. Primitives and information elements

Primitive	Direction
InviteRequest	Client 1 → Server
Status	Client 1 ← Server
InviteResponse	Client 1 ← Server
Status	Client 1 → Server
InviteUserRequest	Client 2 ← Server
Status	Client 2 → Server
InviteUserResponse	Client 2 → Server
Status	Client 2 ← Server

Table 29. Primitive directions in invitation transaction

Information Element	Req	Type	Description

Message-Type	M	InviteRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Invite-ID	M	String	Identifies the invitation.
Invite-Type	M	Enumerated string	Indicates the type of the invitation. (Group (GR), Messaging (IM), Presence (PR), ShareContent (SC), Group Membership (GM))
Recipients	M	Structure	Identifies the user(s) to be invited. (user ID, contact list ID, screen name). If Invite-Type is GM, identifies the group ID.
Invite-Group	C	String	Identifies the related group. (Mandatory if InviteGroup or Group Membership, otherwise not present.)
Invite-Presence	C	Structure	Identifies the related presence attributes. (OPTIONAL if InvitePresence, otherwise not present.)
Invite-Content	C	Structure	Identifies the related shared content as a list of URLs. (OPTIONAL if InviteContent, otherwise not present.)
Invite-Reason	O	String	Textual description of the invitation.
Own-Screen-Name	O	Structure	Identifies the requesting user
Validity	O	Integer	Indicates the time interval in which the invitation is valid. Indicated in seconds.

Table 30. Information elements in InviteRequest primitive

Information Element	Req	Type	Description
Message-Type	M	InviteResponse	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the requestinguser's session.
Invite-ID	M	String	Identifies the invitation.
Invite-Acceptance	M	Boolean	Indicates the user's choice.
Sender	M	Structure	Identifies the responding user. (User ID, screen name). If InviteType was GM, identifies the group ID.
Invite-Response	O	String	Textual description of the response. Present if it was present in InviteUserResponse.

Table 31. Information elements in InviteResponse primitive

Information Element	Req	Type	Description
Message-Type	M	InviteUserRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the requested user's session.
Invite-ID	M	String	Identifies the invitation.
Invite-Type	M	Enumerated string	Indicates the type of the invitation. (Group (GR), Messaging (IM), Presence (PR), ShareContent (SC), Group Membership (GM))
Sender	M	Structure	Identifies the requesting user. (User ID, screen name)

Invite-Group	C	String	Identifies the related group. (Mandatory if InviteGroup or Group Membership.)
Invite-Presence	C	Structure	Identifies the related presence attributes.
Invite-Content	C	Structure	Identifies the related shared content as a list of URLs.
Invite-Reason	O	String	Textual description of the invitation.
Validity	O	Integer	Indicates the time interval in which the invitation is valid. Indicated in seconds.

Table 32. Information elements in InviteUserRequest primitive

Information Element	Req	Type	Description
Message-Type	M	InviteUserResponse	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Invite-ID	M	String	Identifies the invitation.
Invite-Acceptance	M	Boolean	Indicates the user's choice.
Sender	M	Structure	Identifies the requesting user. (User ID, screen name). This value SHOULD be the same as received in the Sender field of InviteUserRequest
Invite-Response	O	String	Textual description of the response.
Own-Screen-Name	O	Structure	Identifies the responding user.

Table 33. Information elements in InviteUserResponse primitive

7.3. Canceling invitations

7.3.1. Transactions

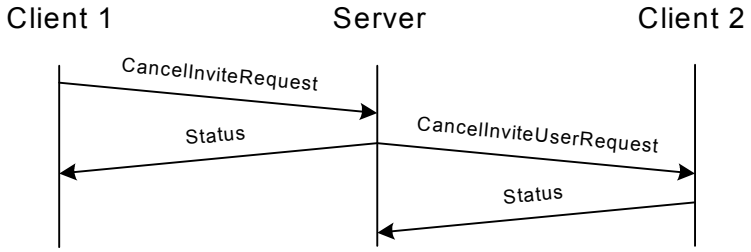

Figure 15. Canceling invitation

A user MAY cancel any previously requested invitation(s). The client sends the CancelInviteRequest message to the server containing a valid Invite-ID, the list of users (user-ID or screen-name) to be notified, and optionally a screen-name and/or a short textual description why the cancellation SHOULD take place. The server responds with a Status message to the client, and sends out CancelInviteUserRequest messages to all requested clients. All clients respond to the server with a Status message. If the invitation was to a group membership, the Recipients element is not present. Note that the cancellation request makes sense only in the scope of contact list invitations, group membership invitations and presence information sharing invitations.

7.3.2. Error conditions

Generic error conditions:

- Service not supported. (405)
- Service unavailable. (503)
- Service not agreed. (506)
- Not logged in. (604)

CancelInviteRequest error conditions:

- Invalid invite-ID. (423)
- Delivery to recipient not available. (410)
- Delivery to recipient domain not available. (516)
- Recipient unknown (UserID or screen-name). (531)
- Recipient unknown (Contact list). (700)

CancelInviteUserRequest error conditions:

- Client MAY ignore any error and respond with Successful. (200)

7.3.3. Primitives and information elements

Primitive	Direction
CancelInviteRequest	Client 1 → Server
Status	Client 1 ← Server
CancelInviteUserRequest	Client 2 ← Server
Status	Client 2 → Server

Table 34. Primitive directions in cancel invitation transaction

Information Element	Req	Type	Description
Message-Type	M	CancelInviteRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Invite-ID	M	String	Identifies the invitation.
Recipients	C	Structure	Identifies the user(s) to be cancelled. (User ID, contact list ID, screen name). Not present if Invite-Type was GM
Cancel-Reason	O	String	Textual description of the cancellation.
Cancelled-Content	C	String	Identifies content to be cancelled as a list of URLs.
Own-Screen-Name	O	Structure	Identifies the requesting user.

Table 35. Information elements in CancelInviteRequest primitive

Information Element	Req	Type	Description
Message-Type	M	CancelInvite UserRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.

Session-ID	M	String	Identifies the session.
Invite-ID	M	String	Identifies the invitation.
Sender	M	Structure	Identifies the requesting user. (User ID, screen name.)
Cancel-Reason	O	String	Textual description of the cancellation.
Cancelled-Content	C	String	Identifies content to be cancelled as a list of URLs.

Table 36. Information elements in CancelInviteUserRequest primitive

7.4. 6.4 Verify WV ID

7.4.1. 6.4.1 Transactions

A user MAY send to the server a list of ID to verify if their validity; the server responds with a status primitive. An ID is a structure containing a combination of user name (UID), resource (CLID or GID) and domain name (Domain).

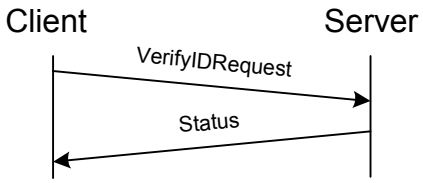

Figure 16. Verify ID

7.4.2. Error Conditions

- General address error (901)
- Unknown user (531).
- Contact list does not exist (700).
- Group does not exist (800).
- Domain not found (404).
- Service Not Supported (405)

7.4.3. Primitives and information elements

Primitive	Direction
VerifyIDRequest	Client → Server
Status	Client ← Server

Table 37. Primitive direction for Verify WV ID

Information Element	Req	Type	Description
Message-Type	M	VerifyIDRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies the session.
ID-List	M	Structure	Includes a list of ID structures to be verified.

Table 38. Information elements in VerifyIDRequest primitive

8. Presence Feature

The relation of contact list and attribute list is described in [FeaFun].

8.1. Contact List

8.1.1. Contact List Properties

There are two properties for Contact List:

- DisplayName: a free text string given by user that can be presented in the user interface of the client.
- Default: a string set by user, 'T' (true) indicates that the particular contact list is the default contact list and 'F' (false) indicates that the list is NOT the default contact list.

When the user creates his/her first contact list, the server automatically sets that contact list as the default (even if the user specifies that the 'Default' property SHOULD be set to 'F'). The server MAY also create the first list automatically.

When the user has more than one contact list in the system, the user MAY set any of his/her contact lists as the default contact list (see ListManageRequest primitive). When the user sets 'Default' property of a contact list to 'T', the 'Default' property of the previously default contact list MUST be set to 'F' automatically by the server.

If the user tries to set the 'Default' property of the default contact list to 'F', the server will silently ignore this. If the user deletes the default contact list, the server will select another contact list as default (which one is implementation-specific at the server).

8.1.2. Transactions

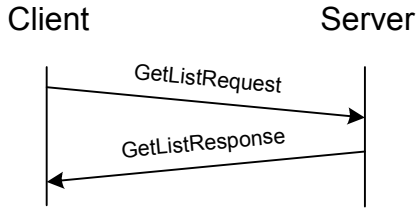

Figure 17. Get list of contact list IDs transaction

The user MAY retrieve the list of all his/her own contact list IDs at once. The default contact list ID is indicated in a separate information element. The client sends the GetListRequest message to the server. The server replies with a GetListResponse message, which contains the list of all contact list IDs. In case there is some error, the server will respond with a Status message instead of the expected GetListResponse message.

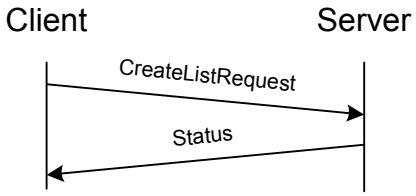

Figure 18. Create contact list transaction

A user is able to create more than one contact list, but there MAY be system specific limitations for the maximum number of lists per user.

The client sends the CreateListRequest message to the server including the ID of the contact list, and optionally any list properties and a list of initial users that SHOULD be added to the list. The server creates the contact list, and responds with a Status message.

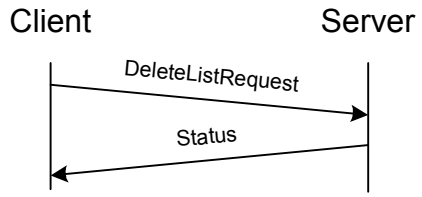

Figure 19. Delete contact list transaction

A user MAY delete a contact list at any time. The server SHOULD NOT unsubscribe the members implicitly, meaning that if a contact list that has been subscribed is deleted, the presence subscriptions SHOULD NOT be cancelled for the particular users.

The client sends the DeleteListRequest message to the server containing the ID of the contact list to be deleted. The server removes the indicated contact list, and responds with a Status message.

Figure 20. Manage contact list transaction

The user MAY retrieve one contact list at once, add and remove members, change the name of a contact list, and set the default contact list. The [CSP SCR] document describes the syntax. The client sends the ListManageRequest message to the server. The server performs the requested operations, and then replies with a ListManageResponse message. If there is an error, the server will respond with a Status message instead of the expected ListManageResponse message.

8.1.3. Error conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

GetListRequest error conditions:

- None except the generic error conditions.

CreateListRequest error conditions:

- Contact list already exists. (701)
- Invalid or unsupported contact list property. (752)
- Unknown user ID. (531)
- The maximum number of contact lists has been reached for the user (753)

DeleteListRequest error conditions:

- Contact list does not exist. (700)

ListManageRequest error conditions:

- Contact list does not exist. (700)
- Invalid or unsupported contact list property. (752)
- Unknown user ID. (531)
- The maximum number of contacts has been reached for the user (754)

8.1.4. Primitives and information elements

Primitive	Direction
GetListRequest	Client → Server
GetListResponse	Client ← Server
CreateListRequest	Client → Server
Status	Client ← Server
DeleteListRequest	Client → Server
Status	Client ← Server
ListManageRequest	Client → Server
ListManageResponse	Client ← Server

Table 39. Primitive directions for contact list management

Information Element	Req	Type	Description
Message-Type	M	GetListRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies the session.

Table 40. Information elements in GetListRequest primitive.

Information Element	Req	Type	Description
Message-Type	M	GetListResponse	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies the session.
Contact-List-ID-List	C	Structure	The list of all contact-list IDs except the default contact list. If empty or omitted, no contact lists exist.
Default-CList-ID	C	String	Identifies the default contact list. If omitted, no default contact list exists.

Table 41. Information elements in GetListResponse primitive.

Information Element	Req	Type	Description
Message-Type	M	CreateList Request	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies the session.
Contact-List-ID	M	String	Identifies the contact list to be created.
Contact-List-Props	O	Structure	The initial properties of the contact list.
User-Nick-List	O	Structure	Identifies the initial users to be added to the contact list (User-ID, Nickname).

Table 42. Information elements in CreateListRequest primitive

Information Element	Req	Type	Description
Message-Type	M	DeleteListRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies the session.
Contact-List-ID	M	String	Identifies the contact list to be deleted.

Table 43. Information elements in DeleteListRequest primitive

Information Element	Req	Type	Description
Message-Type	M	ListManageRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies the session.
Contact-List-ID	M	String	Identifies the contact list.
Add-Nick-List	C	Structure	Identifies the users to be added to the contact list. (User-ID, Nickname)
Remove-Nick-List	C	Structure	Identifies the users to be removed from the contact list (User-ID).
Contact-List-Props	C	Structure	The properties of the contact list to be set.
Receive-List	M	Boolean	Indicates if the client wants to receive the User-Nick-List in the ListManageResponse.

Table 44. Information elements in ListManageRequest primitive.

Information Element	Req	Type	Description
Message-Type	M	ListManage Response	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies the session.
Result	M	Structure	The result of the request.
Contact-List-Props	C	Structure	The properties of the contact list.
User-Nick-List	C	Structure	Contains the list of users on the contact list. (User-ID, Nickname)

Table 45. Information elements in ListManageResponse primitive.

8.2. Attribute list

8.2.1. Transactions

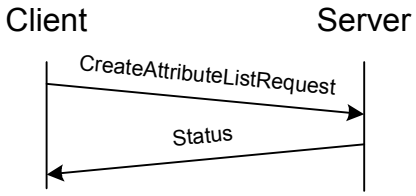

Figure 21. Create attribute list transaction

An attribute list is a set of presence attributes. The list set will define proactive authorization in the way described in [FeaFun]

A user is able to create user or contact-list specific attribute list(s), but only one attribute lists per user or per contact list. In order to modify an attribute list, it can be overwritten by creating another one for the same user-ID or contact-list-ID (there is no need to delete first).

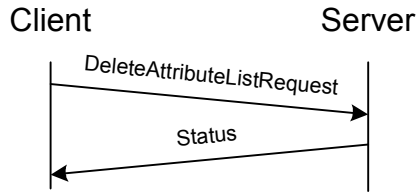

Figure 22. Delete attribute list transaction

A user MAY delete an attribute list from a user and/or from a contact list.

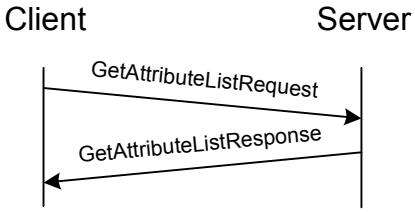

Figure 23. Get attribute list(s) transaction

The publisher MAY retrieve the attributes he/she associates with a specific contact list(s) or user(s), or the default attribute list. If no specific user(s) or contact list(s) are specified the response will include all user-specific and contact-list specific filters. The default attribute list is retrieved if the Default-List element indicates “Yes”.

Changing the proactive authorization will not cancel already active subscriptions. The subscriber will not be sent notifications of attributes not authorized, but if it is reauthorized he will be sent notifications without the need for re-subscribing.

8.2.2. Error conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

CreateAttributeListRequest error conditions:

- Unknown user ID. (531)
- Contact list does not exist. (700)
- Unknown presence attribute (not defined in [PA]). (750)
- The maximum number of attribute lists has been reached for the user (755)

DeleteAttributeListRequest error conditions:

- Unknown user ID. (531)
- Contact list does not exist. (700)

GetAttributeListRequest error conditions:

- Unknown user ID. (531)
- Contact list does not exist. (700)

8.2.3. Primitives and information elements

Primitive	Direction
CreateAttributeListRequest	Client → Server
Status	Client ← Server
DeleteAttributeListRequest	Client → Server
Status	Client ← Server
GetAttributeListRequest	Client → Server
GetAttributeListResponse	Client ← Server

Table 46. Primitive directions for attribute list management

Information Element	Req	Type	Description
Message-Type	M	CreateAttributeListRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies the session.
Presence-Attribute-List	M	Structure	A list of presence attributes. These will be authorized to the user.
User-ID-List	O	Structure	Identifies the user(s) to assign the attribute list association.
Contact-List-ID-List	O	Structure	Identifies the contact list(s) to assign the attribute list association.
Default-List	M	Boolean	Indicates if the attributes are targeted to the default attribute list in addition to the lists specified by the fields User-ID-List and Contact-List-ID-List above.

Table 47. Information elements in CreateAttributeListRequest primitive

Information Element	Req	Type	Description
Message-Type	M	DeleteAttributeListRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies the session.
Contact-List-ID-List	O	Structure	Identifies the contact list(s) to remove the attribute list association
User-ID-List	O	Structure	Identifies the user(s) to remove the attribute list association.
Default-List	M	Boolean	Indicates if the default attribute list SHOULD be cleared.

Table 48. Information elements in DeleteAttributeListRequest primitive

Information Element	Req	Type	Description
Message-Type	M	GetAttributeList	Message identifier.

		Request	
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies the session.
Default-List	M	Boolean	Indicates if the default attribute list is requested.
Contact-List-ID-List	O	Structure	Identifies the contact list(s) to retrieve the associated attribute lists for.
User-ID-List	O	Structure	Identifies the user(s) to retrieve the associated attribute lists for.

Table 49. Information elements in GetAttributeListRequest primitive.

Information Element	Req	Type	Description
Message-Type	M	GetAttributeList Response	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies the session.
Result	M	Structure	Result of the request.
Attribute-Association-List	O	Structure	The list of user, and contact-list presence attribute associations.
Default-Association-List	O	Structure	The list of presence attributes associated with the default list.

Table 50. Information elements in GetAttributeListResponse primitive.

8.3. Presence Information delivery

8.3.1. Subscribed Presence Transactions

8.3.1.1. Transactions

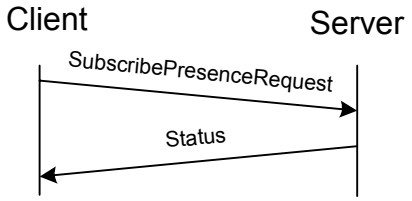

Figure 24. Subscribe presence transaction

The requesting client sends a SubscribePresenceRequest message to the server including the user(s) (specified by user-ID, or by Contact-List-ID, or both), the list of presence attributes to subscribe (if not present, then all attributes are requested). The server responds with a Status message.

When the subscription of presence information is complete, the requesting user will immediately receive the current presence information as a presence notification and future presence changes as subscribed to. The scope of the subscription is either a single user or a contact list referring to multiple users.

If the requesting client subscribes to a contact list, the requesting client MAY request the server to automatically subscribe to the presence attributes when a new user is added to this contact list, and automatically unsubscribe to the presence attributes when the contact list is deleted or when a user is removed from the contact list. Note that the automatic subscription / un-subscription is merely a characteristics of the subscription / un-subscription itself.

The requesting user MAY subscribe only part of the presence information and, correspondingly, the user whose presence information is subscribed MAY allow only part of the presence information to be delivered.

The subscription is not persistent through different sessions.

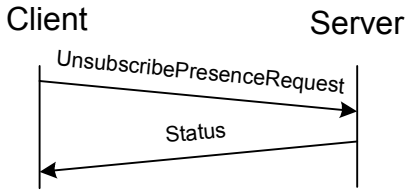

Figure 25. Unsubscribe presence transaction

When the requesting user does not want to receive any more the presence information, he MAY unsubscribe the presence information. The presence data of multiple users MAY be un-subscribed in one message. The server stops delivering all presence information for the un-subscribed user, even if the user is included in a contact list that is used for presence subscription.

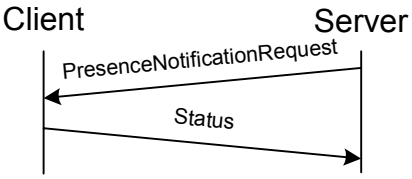

Figure 26. Presence notification

As long as authorized and subscribing, the requesting user will always receive new presence information when the other party updates its presence information. The server sends PresenceNotificationRequest message to the client containing the updated presence information. The client responds with a Status message.

Figure 27. Get watcher list transaction

The user MAY get the list of users that subscribe to his/her presence attributes. The requesting client sends a GetWatcherListRequest message to the server. The server responds with a GetWatcherListResponse message including the list of the subscribing users.

Note: There are no ways of retrieving exactly what attributes every user subscribes to, but the GetAttributeListRequest can tell what attributes they are authorized to see.

8.3.1.2. Error Conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)

- Service not supported. (405)

SubscribePresenceRequest error conditions:

- Unknown user ID. (531)
- Contact list does not exist. (700)
- Unknown presence attribute (not defined in [PA]). (750)
- Automatic subscription / unsubscription is not supported (760)

UnSubscribePresenceRequest error conditions:

- Unknown user ID. (531)
- Contact list does not exist. (700)

PresenceNotificationRequest error conditions:

- Client MAY ignore any error and respond with Successful. (200)

GetWatcherListRequest error conditions:

- None except the generic error conditions.

8.3.1.3. Primitives and information elements

Primitive	Direction
SubscribePresenceRequest	Client → Server
Status	Client ← Server
UnSubscribePresenceRequest	Client → Server
Status	Client ← Server
PresenceNotificationRequest	Client ← Server
Status	Client → Server
GetWatcherListRequest	Client → Server
GetWatcherListResponse	Client ← Server

Table 51. Primitive directions

Information Element	Req	Type	Description
Message-Type	M	SubscribePresence Request	Message identifier.
Transaction-ID	M	String	Identifies the subscription request transaction.
Session-ID	M	String	Session ID for session.
User-ID-List	C	Structure	Identifies the IM user(s).
Contact-List-ID-List	C	Structure	Identifies the set(s) of users for subscription.
Presence-Attribute-List	O	Structure	A list of presence attributes. An empty or missing list indicates all available presence attributes are desired.
Auto-Subscribe	M	Boolean	'T' means that the automatic subscription to the presence attributes is enabled when a new user is added to the contact list, and the automatic un-subscription to the presence attributes is also enabled when the contact list is deleted or when a user is removed from the contact list. 'F' means that the automatic subscription / un-subscription is disabled.

Table 52. Information elements in SubscribePresenceRequest primitive

Information Element	Req	Type	Description
Message-Type	M	UnSubscribe PresenceRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Session ID for session.
User-ID-List	C	Structure	Identifies the IM users(s).
Contact-List-ID-List	C	Structure	Identifies the set of users to be unsubscribed.

Table 53. Information elements in UnsubscribePresenceRequest primitive

Information Element	Req	Type	Description
Message-Type	M	PresenceNotificationRequest	Message identifier
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies the session
Presence-Value-List	M	Structure	List of User IDs and its corresponding presence values.

Table 54. Information elements in PresenceNotificationRequest primitive

Information Element	Req	Type	Description
Message-Type	M	GetWatcherList Request	Message identifier.
Transaction-ID	M	String	Identifies the subscription request transaction.
Session-ID	M	String	Session ID for session
History Period	O	Integer	Indicates the time period in seconds on the longest possible history of the watcher (from the time of request) that SHOULD be returned. In case of absence, it indicates the user request the watcher list at the time of the request only. The value 0 MUST NOT be used.
MaxWatcherList	O	Integer	Indicates the maximum number of Watcher element in GetWatcherListReponse.

Table 55. Information elements in GetWatcherListRequest primitive

Information Element	Req	Type	Description
Message-Type	M	GetWatcherList Response	Message identifier.
Transaction-ID	M	String	Identifies the subscription request transaction.
Session-ID	M	String	Session ID for session
History-Period	O	Integer	Indicates the time period in seconds in which the watcher history has been accumulated. This value MUST NOT be

			larger than the requested period. Absence indicates that the Watcher information element (see below) returns only the current subscribers at the time of the request even if the history log is requested. The value 0 MUST NOT be used.
Watcher	O	Structure	Identifies the watchers and their status from the history period. If this element is not present at all within the response, it indicates that the server does not give any watcher information at this time. The number of this element in GetWatcherListResponse MUST NOT be larger than Max Watcher List value in the corresponding GetWatcherListRequest.

Table 56. Information elements in GetWatcherListResponse primitive

8.3.2. Get Presence Transactions

8.3.2.1. Transactions

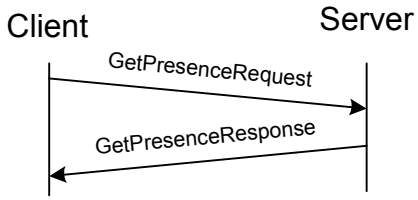

Figure 28. Get Presence transaction

A user MAY, if authorized, get another user's presence information at any time. The client sends a GetPresenceRequest message to the server containing the user-ID or a contact list name, and optionally the list of presence requested attributes.

Absence of the presence attribute list in the request indicates to the server that all available presence information is requested. If no information is available about a particular presence attribute the corresponding presence attribute is not returned in the GetPresenceResponse message.

The server responds with a GetPresenceResponse message containing the result of the request and the presence attributes.

Only those attributes that are pro- or reactively authorized will be distributed. See the [FeaFun] document for clarification on that process.

8.3.2.2. Error Conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

GetPresenceRequest error conditions:

- Unknown user ID. (531)

- Contact list does not exist. (700)
- Unknown presence attribute (not defined in [PA]). (750)

8.3.2.3. Primitives and information elements

Primitive	Direction
GetPresenceRequest	Client → Server
GetPresenceResponse	Client ← Server

Table 57. Primitive directions for getting presence

Information Element	Req	Type	Description
Message-Type	M	GetPresenceRequest	Message identifier
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Session ID for session
User-ID-List	C	Structure	List of identifications of the requested IM users.
Contact-List-ID-List	C	Structure	Identifies the set of Users IDs.
Presence-Attribute-List	O	Structure	A list of presence attributes. An empty or missing list indicates all available presence attributes are desired.

Table 58. Information elements in GetPresenceRequest primitive

Information Element	Req	Type	Description
Message-Type	M	GetPresenceResponse	Message identifier
Transaction-ID	M	String	Identifies the transaction.
Session-ID	O	String	Identifies the session
Result	M	Structure	Result of the request.
Presence-Value-List	O	Structure	List of User IDs and its corresponding presence values.

Table 59. Information elements in GetPresenceResponse primitive

8.3.3. Reactive presence authorization

8.3.3.1. Transactions

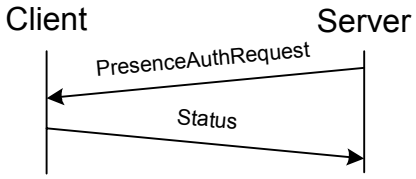

Figure 29. Reactive presence authorization request transaction

If a publisher requests reactive presence authorization, the server will send a PresenceAuthRequest message to the client containing the ID of the requesting user (which identifies the authorization request also) and the list of requested presence-attributes (not present if all of them are requested).

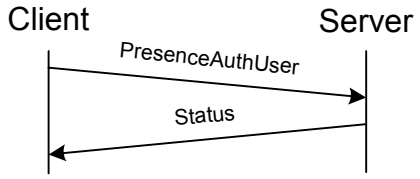

Figure 30. Reactive presence authorization of user transaction

The client responds to the server in a separate transaction with a PresenceAuthUser message that contains the user-ID of the requesting user, and a list of attributes to add to the list of granted or denied presence attributes. If the list is not present or empty then access is granted or denied to all presence attributes. The server replies with a Status message.

From this authorization status the requesting user MAY access the authorized presence information.

A new authorization will overwrite the existing one. Any attribute previously granted or denied that is not specified in the new authorization will not be changed. An exception is the empty list, which will overwrite all authorizations.

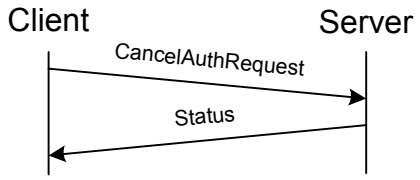

Figure 31. Cancel presence authorization transaction

A user MAY cancel a previous presence authorization. The client will send CancelAuthRequest message to the server containing the user-ID. The server responds with a Status message.

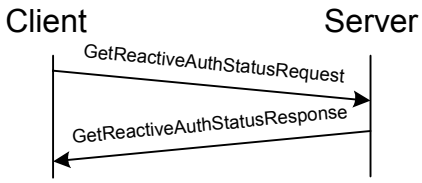

Figure 32. Get reactive authorization status transaction

A client MAY retrieve a list of users that he has granted or denied authorization to along with a list of pending reactive authorization requests. The client will send a GetReactiveAuthStatusRequest message to the server. The server will return the current status of the reactive authorization function.

8.3.3.2. Error Conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

PresenceAuthRequest error conditions:

- Client MAY ignore any error and respond with Successful. (200)

PresenceAuthUser error conditions:

- Unknown authorization request or user ID. (531)

CancelAuthRequest error conditions:

- Unknown authorization request or user ID. (531)

8.3.3.3. Primitives and information elements

Primitive	Direction
PresenceAuthRequest	Client ← Server
Status	Client → Server
PresenceAuthUser	Client → Server
Status	Client ← Server
CancelAuthRequest	Client → Server
Status	Client ← Server
GetReactiveAuthStatusRequest	Client → Server
GetReactiveAuthStatusResponse	Client ← Server

Table 60. Primitive directions for reactive presence authorization

Information Element	Req	Type	Description
Message-Type	M	PresenceAuthRequest	Message identifier
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies for session.
User-ID	M	String	Identification of the requesting IM user (and the authorization request).
Presence-Attribute-List	O	Structure	A list of presence attributes. An empty or missing list indicates all available presence attributes are desired.

Table 61. Information elements in PresenceAuthRequest primitive

Information Element	Req	Type	Description
Message-Type	M	PresenceAuthUser	Message identifier
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies the session.
User-ID	M	String	Identifies the authorization request (and the user).
Acceptance	M	Boolean	Indicates whether the user accepts (authorize) or declines (not authorize) the request.
Presence-Attribute-List	O	Structure	A list of presence attributes that is to be added to the granted list or denied list. An empty or missing list indicates that the Acceptance flag SHOULD apply to all presence attributes.

Table 62. Information elements in PresenceAuthUser primitive

Information Element	Req	Type	Description
Message-Type	M	CancelAuthRequest	Message identifier

Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies the session.
User-ID	M	String	Identifies the authorization request (and the user).

Table 63. Information elements in CancelAuthRequest primitive

Information Element	Req	Type	Description
Message-Type	M	GetReactiveAuth StatusRequest	Message identifier
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies the session.
User-ID-List	O	String	Identifies the user(s) to retrieve the reactive authorization status for.

Table 64. Information elements in GetReactiveAuthStatusRequest primitive

Information Element	Req	Type	Description
Message-Type	M	GetReactiveAuth StatusResponse	Message identifier
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Identifies the session.
ReactiveAuthStatus-List	M	Structure	Reactive authorization status list.

Table 65. Information elements in GetReactiveAuthStatusResponse primitive

8.3.4. Update Presence Transactions

8.3.4.1. Transactions

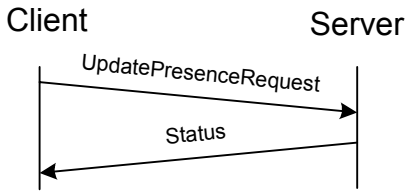

Figure 33. Update presence transaction

An owner of the presence data MAY update presence attributes and their values. Only the updated attributes and their values need to be carried in this primitive, the omitted attributes are not modified.

8.3.4.2. Error Conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

UpdatePresenceRequest error conditions:

- Unknown presence attribute (not defined in [PA]). (750)
- Unknown presence value (not defined in [PA]). (751)

8.3.4.3. Primitives and information elements

Primitive	Direction
UpdatePresenceRequest	Client → Server
Status	Client ← Server

Table 66. Primitive directions for UpdatePresenceRequest

Information Element	Req	Type	Description
Message-Type	M	UpdatePresence Request	Message identifier
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Session ID for session.
Update-Value-List	M	Structure	A list of presence values to update.

Table 67. Information elements in UpdatePresenceRequest

9. Instant Messaging Feature

9.1. Delivery Transactions

9.1.1. Send Message Transaction

9.1.1.1. Transactions

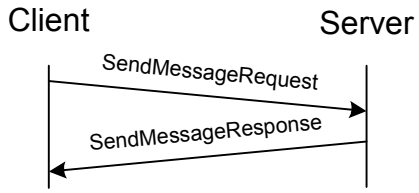

Figure 34. Send Message transaction

The user MAY send message to other user(s), to a group or to another user in a group. The client sends the SendMessageRequest to the instant messaging service. The element contains the recipient(s), message itself, and optionally delivery report request and validity. The date and time element of the Message-Info structure is not present. The server responds with the SendMessageResponse message, which contains the Message-ID in order to identify the message later.

One of the elements in the "SendMessageRequest" is the "Message-info", which includes Content-type, Transfer-encoding, Content-length, Sender, Date, Time and other useful information. The default Content-type is "text/plain; charset=utf-8". The default transfer encoding is "identity" meaning no encoding.

When the validity has expired, the message delivery is not REQUIRED, and the message SHALL be dropped without notice.

The sender MAY request a delivery report.

9.1.1.2. Error conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

SendMessageRequest error conditions:

- Unsupported content-type. (415)
- Domain not supported. (516)
- Contact list does not exist. (700)
- Recipient user does not exist. (531)
- Recipient user blocked the sender. (532)
- Recipient user is not logged in. (533)
- Message queue full. (507)
- Recipient group does not exist. (800)
- Sender has not joined the group (or kicked). (808)
- Private messaging is disabled in the group. (812)
- Private messaging is disabled for the recipient. (813)

9.1.1.3. Primitives and information elements

Primitive	Direction
SendMessageRequest	Client → Server
SendMessageResponse	Client ← Server

Table 68. Primitive directions for Send Message Transaction

Information Element	Req	Type	Description
Message-Type	M	SendMessageRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Session ID for session.
Delivery-Report-Request	M	Boolean	Indicates if the user wants delivery report.
Message-Info	M	Structure	Message information data. (MessageID or MessageURI, MIME-type, encoding, size, sender and recipient(s) (userID, clientID, screen name, group, contact-list), date and time, validity).
Content	C	String | Binary data	The content of the instant message.

Table 69. Information elements in SendMessageRequest

Information Element	Req	Type	Description
Message-Type	M	SendMessageResponse	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Session ID for session.
Result	M	Structure	Result of the SendMessageRequest
Message-ID	C	String	Server generated message ID for this message.

Table 70. Information elements in SendMessageResponse primitive

9.1.2. Set Delivery Method transaction

9.1.2.1. Transactions

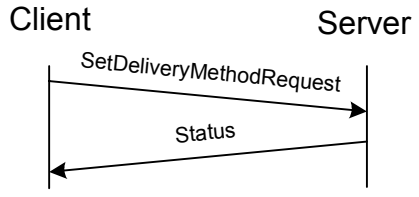

Figure 35. Set Delivery Method transaction

The recipient client indicates the REQUIRED message delivery method in the "Capability-List" element during the login phase. The client MAY change the message delivery method during the session. This is accomplished by using "SetDeliveryMethodRequest".

One important element in the "SetDeliveryMethodRequest" element is "AcceptedContentLength" that indicates the maximum message size that can be pushed to the client when using "Push". If the message size is larger than the "AcceptedContentLength", or the content type is application/vnd.wap.mms-message, the server SHALL use "Notify/Get" instead of "Push".

When the server delivers a message to the client, if the message body itself is stored in the server and the server is expected to deliver it directly, the server SHOULD choose either "Push" or "Notify/Get" depending on the message size and the default delivery method of the recipient.

The message MAY also be retrievable from some other message server, possibly a 3rd party content server. In this case, the server uses "Notify/Get" method no matter irrespective of the delivery method or size.

9.1.2.2. Error Conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

SetDeliveryMethodRequest error conditions:

- Group does not exist. (800)

9.1.2.3. Primitives and information elements

Primitive	Direction
SetDeliveryMethodRequest	Client → Server
Status	Client ← Server

Table 71. Primitive directions for Set Delivery Method Transaction

Information Element	Req	Type	Description
Message-Type	M	SetDeliveryMethod Request	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Session ID for session.
Delivery-Method	M	Enumerated character	Determines the type of message delivery. Push means that the complete message is transferred in the notification. Notify/Get means that only the message ID is transferred in the notification; the message is then retrieved using a get.
Accepted-Content-Length	O	Integer	Maximum size of message that can be pushed when using PUSH.
Group-ID	O	String	Group ID if delivery method refers to a group.

Table 72. Information elements in SetDeliveryMethodRequest

9.1.3. Get Message List Transactions

9.1.3.1. Transactions

Figure 36. Get Message List transaction

The server MAY offer space where undelivered messages or group history can be stored. The get message list transaction is used to get the Message-IDs of the undelivered messages to be used in a GetMessageRequest or RejectMessageRequest. Notified messages that the client did not fetch MAY be manually retrieved.

9.1.3.2. Error Conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

GetMessageListRequest error conditions:

- Group does not exist. (800)
- Group is not joined (or kicked). (808)
- History is not supported. (821)

9.1.3.3. Primitives and information elements

Primitive	Direction
GetMessageListRequest	Client → Server
GetMessageListResponse	Client ← Server

Table 73. Primitive directions for Get Message List transactions

Information Element	Req	Type	Description
Message-Type	M	GetMessageListRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Session ID for session.
Group-ID	C	String	Identifies the group to retrieve history.
Message-Count	O	Integer	The maximal number of message-info structures to be returned.

Table 74. Information elements in GetMessageListRequest

Information Element	Req	Type	Description
Message-Type	M	GetMessageList	Message identifier.

		Response	
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Session ID for session.
Message-Info-List	M	Structure	Message information data for each message. (MessageID or MessageURI, MIME-type, encoding, size, sender and recipient(s) (userID, clientID, screen name, group, contact-list), date and time, validity).

Table 75. Information elements in GetMessageListResponse

9.1.4. Reject Message Transactions

9.1.4.1. Transactions

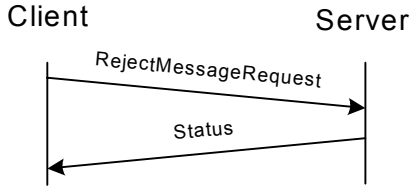

Figure 37. Reject Message transaction

If the server offers space where undelivered messages are stored, the user MAY need to reject unwanted messages. Server responds with Status indicating the outcome.

If the originating user did request a delivery report, it will be indicated for him/her that the message has been rejected.

9.1.4.2. Error Conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

RejectMessageRequest error conditions:

- Invalid Message-ID. (426)

9.1.4.3. Primitives and information elements

Primitive	Direction
RejectMessageRequest	Client → Server
Status	Client ← Server

Table 76. Primitive directions for Reject Message transactions

Information Element	Req	Type	Description

Message-Type	M	RejectMessageRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	Integer	Session ID for session
Message-ID-List	M	Structure	Identifies the messages to be removed.

Table 77. Information elements in RejectMessageRequest

9.1.5. Message Delivery Transactions

9.1.5.1. Transactions

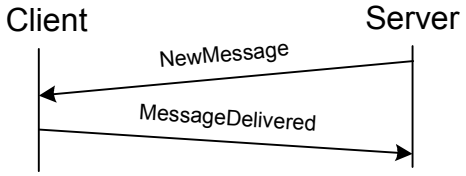

Figure 38. New Message transaction

The NewMessage primitive delivers new messages to the client. The client responds to the NewMessage primitive with a MessageDelivered and returns the transaction ID and message ID from the NewMessage.

If the content of the message body contains some reference URI's to the content of the 3rd party content server, the client SHOULD be prepared to take further steps to retrieve the actual content from the 3rd party content server.

9.1.5.2. Error Conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

NewMessage error conditions:

- Client will not accept the message delivery. (410)
- Client does not support the content type. (415)

MessageDelivered error conditions:

- Invalid Message-ID. (426)

9.1.5.3. Primitives and information elements

Primitive	Direction
NewMessage	Client ← Server
MessageDelivered	Client → Server

Table 78. Primitive directions for Message Delivery Transactions

Information Element	Req	Type	Description
Message-Type	M	NewMessage	Message identifier.

Transaction-ID	M	String	Identifies the transaction. Either generated by server (PUSH) or by client when doing a GetMessageRequest.
Session-ID	M	String	Session ID for session
Message-Info	M	Structure	Message information data. (MessageID or MessageURI, MIME-type, encoding, size, sender and recipient(s) (userID, clientID, screen name, group, contact-list), date and time, validity).
Content	C	String | Binary data	Message data. Not present if Message-Info contains a MessageURI.

Table 79. Information elements in NewMessage

Information Element	Req	Type	Description
Message-Type	M	MessageDelivered	Message identifier.
Transaction-ID	M	String	Identifies the transaction. Either generated by client (after a GetMessageRequest) or by server when client is responding to a NewMessage.
Session-ID	M	String	Session ID for session.
Message-ID	M	String	ID of message that has been delivered

Table 80. Information elements in MessageDelivered

9.1.6. Message Notification Transactions

9.1.6.1. Transactions

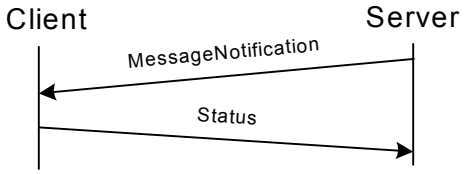

Figure 39. Message Notification transaction

The user MAY receive MessageNotification(s) from the server when the delivery method is “Notify/Get”. The server sends the MessageNotification message to the client containing the Message-Info whenever a new message arrives. The client responds with a Status message.

The Message-Info element includes information about the message. The information that carried is: the MIME-type, the encoding, the size, the sender, the date and time, and the validity of the message.

When the client receives a message notification that refers to the message using the "Message-ID" element, the message is retrievable from the WV IM service element.

The message MAY also be retrievable from some other message server instead of a WV IM service element, possibly a 3rd party content server. In this case, the message notification refers to the message using the "Message-URL" element instead of the "Message-ID" element.

When the client receives a message notification that refers to the message using the "Message-URL" element, the URL indicates the location of the message as well as the protocol that is used to retrieve the message. The client SHOULD provide a mechanism to the user to retrieve the message whenever feasible, but it is up to the client implementation to support the protocol and retrieval mechanisms not defined in these specifications.

When the message is referred with the "Message-URL" element, the message cannot be rejected.

9.1.6.2. Error Conditions

MessageNotification error conditions:

- Client MAY ignore any error and respond with Successful. (200)

9.1.6.3. Primitives and information elements

Primitive	Direction
MessageNotification	Client ← Server
Status	Client → Server

Table 81. Primitive directions for Message Notification Transaction

Information Element	Req	Type	Description
Message-Type	M	Message Notification	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Session ID for session.
Message-Info	M	Structure	Message information data. (MessageID or MessageURI, MIME-type, encoding, size, sender and recipient(s) (userID, clientID, screen name, group, contact-list), date and time, validity).

Table 82. Information elements in MessageNotification

9.1.7. Get Message Transactions

9.1.7.1. Transactions

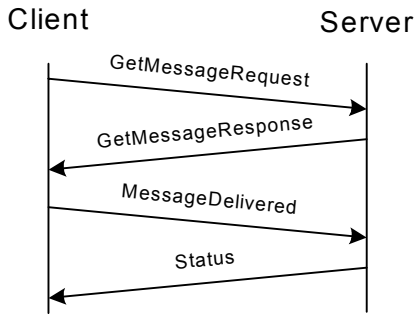

Figure 40. Get Message Transaction

GetMessageRequest is used to retrieve messages using a message ID. The server responds to the GetMessageRequest with a GetMessageResponse containing the requested instant message. The client sends MessageDelivered primitive to the server containing the Message-ID to indicate that the instant message has been successfully delivered. The server removes the delivered message from the server, and sends out the delivery report(s) if it was requested.

If the content of the message body contains some reference URI's to the content of the 3rd party content server, the client SHOULD be prepared to take further steps to retrieve the actual content from the 3rd party content server.

9.1.7.2. Error Conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

GetMessageRequest error conditions:

- Invalid Message-ID. (426)

9.1.7.3. Primitives and information elements

Primitive	Direction
GetMessageRequest	Client → Server
GetMessageResponse	Client ← Server
MessageDelivered	Client → Server
Status	Client ← Server

Table 83. Primitive directions for Get Message Transaction

Information Element	Req	Type	Description
Message-Type	M	GetMessageRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Session ID for session.
Message-ID	M	String	ID of the message to retrieve.

Table 84. Information elements in GetMessageRequest

Information Element	Req	Type	Description
Message-Type	M	GetMessage Response	Message identifier.
Transaction-ID	M	String	Identifies the transaction. Either generated by server (PUSH) or by client when doing a GetMessageRequest.
Session-ID	M	String	Session ID for session
Message-Info	M	Structure	Message information data. (MessageID or MessageURI, MIME-type, encoding, size, sender and recipient(s) (userID, clientID, screen name, group, contact-list), date and time, validity).
Content	C	String | Binary data	Message data. Not present if

			Message-Info contains a MessageURI.

Table 85. Information elements in GetMessageResponse

9.1.8. Delivery Status Report Transaction

9.1.8.1. Transactions

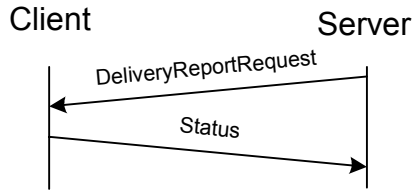

Figure 41. Delivery Status Report Transaction

A user who sends a message MAY request a delivery report. If the sender did request a delivery report, the server will send DeliveryReportRequest to the originating user when it delivered the message to each recipient client.

The delivery report can also inform the client about a delivery attempt that was unsuccessful due to detected error conditions on the receiving side.

9.1.8.2. Error conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

DeliveryReportRequest error conditions:

- Client MAY ignore any error and respond with Successful. (200)

DeliveryReportRequest error indications:

- Unsupported content-type. (415)
- Domain not supported. (516)
- Contact list does not exist. (700)
- Recipient user does not exist. (531)
- Recipient user blocked the sender. (532)
- Recipient user is not logged in. (533)
- Message queue full. (507)
- Recipient group does not exist. (800)
- Sender has not joined the group (or kicked). (808)
- Private messaging is disabled in the group. (812)
- Private messaging is disabled for the recipient. (813)
- Message has been rejected (538)
- Message has been forwarded (541)
- Message has expired (not delivered) (542)

9.1.8.3. Primitives and information elements

Primitive	Direction
DeliveryReportRequest	Client ← Server
Status	Client → Server

Table 86. Primitive directions for delivery status report transaction

Information Element	Req	Type	Description
Message-Type	M	DeliveryReport Request	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Result	M	Structure	Result of the delivery.
Delivery-Time	O	DateTime	Date and time of delivery.
Message-Info	M	Structure	Message information data. (MessageID or MessageURI, MIME-type, encoding, size, sender and recipient(s) (userID, clientID, screen name, group, contact-list), date and time, validity).

Table 87. Information elements in DeliveryReportRequest primitive

9.1.9. Forward message transaction

9.1.9.1. Transactions

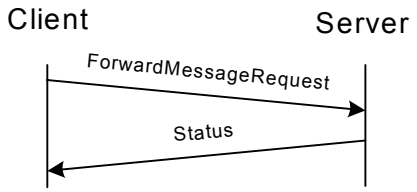

Figure 42. Forward message transaction

A user can forward a non-retrieved instant message. The client sends ForwardMessageRequest to the server containing the ID of the message to be forwarded. The server forwards the message to the requested recipients and it is removed from the user's inbox. The server responds with a Status message.

9.1.9.2. Error conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

DeliveryReportRequest error conditions:

- Invalid Message-ID. (426)
- Unsupported content-type. (415)
- Domain not supported. (516)

- Contact list does not exist. (700)
- Recipient user does not exist. (531)
- Recipient user blocked the sender. (532)
- Recipient user is not logged in. (533)
- Message queue full. (507)
- Recipient group does not exist. (800)
- Sender has not joined the group (or kicked). (808)
- Private messaging is disabled in the group. (812)
- Private messaging is disabled for the recipient. (813)

9.1.9.3. Primitives and information elements

Primitive	Direction
ForwardMessageRequest	Client → Server
Status	Client ← Server

Table 88. Primitive directions for forward message transaction

Information Element	Req	Type	Description
Message-Type	M	ForwardMessage Request	Message identifier.
Transaction-ID	M	String	Identifies the transaction.
Session-ID	M	String	Session ID for session
Message-ID	M	String	Identifies the message to forward.
Recipients	M	Structure	Identifies the user(s) to whom the instant message SHOULD be forwarded. (User ID, contact list ID, group ID, screen name).

Table 89. Information elements in ForwardMessageRequest

9.2. Access Control Transactions

9.2.1. Blocking Incoming Messages and Invitations Transaction

The concept of *blocking* means restricting message or invitation delivery from certain entities. These entities MAY be:

- user(s) specified by User-ID(s) or screen-name(s), and
- group(s) specified by group-ID(s).

The messages or invitations sent by the blocked entity are not delivered to the blocker.

The messages or invitations sent by the granted entity are delivered to the user.

Blocking is active for the blocked entities(s) until the user decides to turn off the use of the Blocked-Entity-List or to unblock (remove from the list) the particular entity.

Granting is active for the granted entities(s) until the user decides to turn off the use of the Granted-Entity-List or to un-grant (remove from the list) the particular entity.

The Blocked-Entity-List list has priority over the Granted-Entity-List; see the decision tree below:

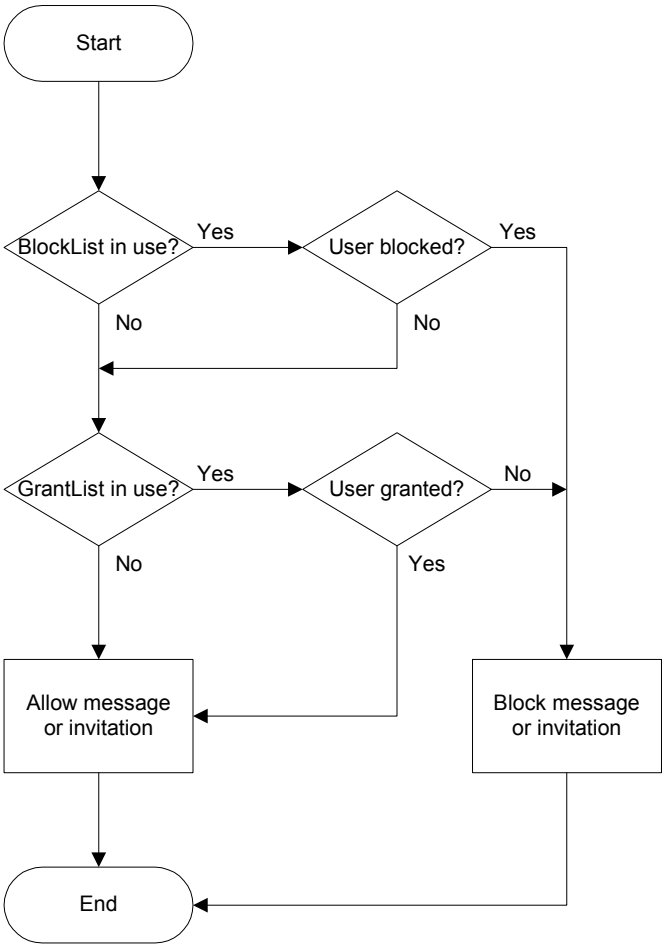

Figure 43. Blocking decision-tree

9.2.1.1. Transactions

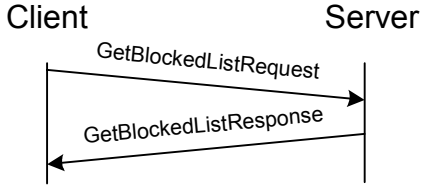

Figure 44. Get list of blocked entities transaction

A user MAY get his/her own list of blocked entities at any time. The client sends the GetBlockedListRequest message to the server. The server responds with the GetBlockedListResponse message, which includes the list of blocked and granted entities.

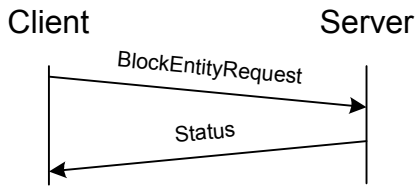

Figure 45. Block/unblock entities transactions

A user MAY block/un-block any other entity at any time. The client sends the BlockEntityRequest message to the server containing the list of entities to be blocked/unblocked and/or to be granted/un-granted. The server responds with a Status message to the client.

9.2.1.2. Error conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

GetBlockedListRequest error conditions:

- None except the generic error conditions.

BlockEntityRequest error conditions:

- Unknown user-ID or ScreenName. (531)
- Unknown group-ID. (800)

9.2.1.3. Primitives and information elements

Primitive	Direction
GetBlockedListRequest	Client → Server
GetBlockedListResponse	Client ← Server
BlockEntityRequest	Client → Server
Status	Client ← Server

Table 90. Primitive directions for block transactions

Information Element	Req	Type	Description
Message-Type	M	GetBlockedList Request	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.

Table 91. Information elements in GetBlockedListRequest primitive

Information Element	Req	Type	Description
Message-Type	M	GetBlockedList Response	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Blocked-Entity-List	C	Structure	The list of currently blocked entities.

Blocked-List-Inuse	M	Boolean	Indicates if the list of blocked entities is currently in use (active).
Granted-Entity-List	C	Structure	The list of currently granted users.
Granted-List-Inuse	M	Boolean	Indicates if the list of granted entities is currently in use (active)

Table 92. Information elements in GetBlockedListResponse primitive

Information Element	Req	Type	Description
Message-Type	M	BlockEntity Request	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Block-Entity-List	O	Structure	A list of entities to be added to the Blocked-Entity list.
Unblock-Entity-List	O	Structure	A list of entities to be removed from the Blocked-Entity list.
Blocked-List-Inuse	O	Boolean	Indicates if the list of blocked entities is currently in use (active).
Grant-Entity-List	O	Structure	A list of entities to be added to the Granted-Entity list.
Ungrant-Entity-List	O	Structure	A list of entities to be removed from the Granted-Entity list.
Granted-List-Inuse	O	Boolean	Indicates if the list of granted entities is currently in use (active)

Table 93. Information elements in BlockEntityRequest primitive

9.3. Message Content Format

"Text/plain; charset=UTF-8" is mandatory and default content type.

The suggested content types are:

vCard 2.1 as defined in [VCARD21],

vCalendar 1.0 as defined in [VCAL10],

application/vnd.wap.mms-message as defined in [WAPMMS].

EMS application/vnd.3gpp.sms as defined in [TS23040] and application/x-sms as defined in [TS23140]

The suggested content types, while not mandatory, are content types that are highly RECOMMENDED to further maximize the interoperability of the clients.

Note that the MMS standardization is an ongoing effort in 3GPP and WAP Forum. The RECOMMENDED MMS content-type "application/vnd.wap.mms-message" SHALL be consistent with the standardization effort, and support the standard.

The message MAY carry other types of content. In this case, the "Content-type" will be consistent with the MIME types that are standardized in IETF [RFC2045] [RFC2046] or WAP Forum.

While some content types are RECOMMENDED to facilitate interoperability, the server SHALL recognize the client capability through "Capability Negotiation" so as to ensure the interoperability of different types of content between clients.

10. Group Feature

10.1. Group models

The concept of *user group* means a discussion forum formed by two or more individuals (users) to exchange information, opinions, comments, thoughts about a particular issue, which is the *topic* of the particular group.

These user groups MAY be categorized by:

Ownership:

- Public (created and maintained by service provider),
- Private (created and maintained by a subscriber),

Membership:

- Open (any users MAY join the group),
- Restricted (only particular users MAY join the group).

The management of public groups is not within the scope of this document.

The User group transactions are divided to two categories:

- more or less Static user group transactions,
- and Dynamic user group transactions:

10.1.1.1. Static user group transactions

These transactions include user group management, and other transactions that are not used frequently:

- Creation, modification, and deletion of groups,
- Adding, removing group members,
- Setting access rights,
- Getting information about a group,
- Subscription to group change notification.

10.1.1.2. Dynamic user group transactions

These transactions include frequently used transactions:

- Joining, leaving a group,
- Inviting other users to a group,
- Rejecting users,
- Notification about group changes.

10.1.2. Private group model

A group is considered as *private*, if an ordinary IM user has created it, and that user maintains it.

10.1.3. Public group model

A group is considered as *public*, if the service provider has created it, and the service provider maintains it.

10.1.4. Access privileges

There are three levels of access privileges to the restricted groups:

- Administrator,
- Moderator,
- User.

Administrators can do anything in a group.

The creator of the particular group always has administrator privileges (administrator privileges cannot be removed) as long as the group exists. He/she cannot be rejected in that group.

Moderators can add/remove/reject members who are ordinary users but not moderators or administrators.

Users do not have any privileges other than to join/leave to/from the group, and to send/receive messages.

The following table describes the availability of transactions for each privilege level.

Y=fully available, N=not available.

Name	Administrators	Moderators	Users
Send/receive messages	Y	Y	Y
Send/receive private messages	Y	Y	Y
Create group	N/A	N/A	N/A
Delete group	Y	N	N
Join/leave group	Y	Y	Y
Get/add/remove group members	Y	Y	N
Get group properties	Y	Y	Y
Set group properties	Y	N	N
Get/set own properties	Y	Y	Y
Get/modify reject list	Y	Y	N
Subscribe group change	Y	Y	Y
Group change notification	Y	Y	Y
Modify member access rights	Y	N	N
Get list of joined users	Y	Y	Y

10.1.5. Group properties

The values of the group properties are defined by the owner, or by group member(s) with sufficient access rights. Only Administrators MAY modify these property values. Each group MAY have the following properties:

- Name: a string that is presented to the user as the name of the group (not necessarily same as GroupID!). Default value is an empty string. This is an OPTIONAL property (the client does not have to specify it in the CreateGroupRequest).
- Accesstype:

  open (for everyone) or

  restricted (members only).

  Default value is “open”. The client does not have to specify it in the CreateGroupRequest.

- Type[1]:
  - public (maintained by service provider) or
  - private (maintained by individual user(s)).
- PrivateMessaging:
  - T (sending private messages is enabled) or
  - F (sending private messages is disabled).

  Default value is “F”. The client does not have to specify it in the CreateGroupRequest.
- Searchable:
  - T (the group is subject to search) or
  - F (the group is no included in searching).

  Default value is “F”. The client does not have to specify it in the CreateGroupRequest.
- Topic: a string that describes the subject of discussion in the group. The topic is subject to searching if allowed. The default value is an empty string. This is an OPTIONAL property (the client does not have to specify it in the CreateGroupRequest).
- ActiveUsers[2]: an integer number that indicates the number of currently joined users.
- MaxActiveUsers: an integer number that indicates the maximum number of joined users at any given time. Default value is set by the group service. This is an OPTIONAL property (the client does not have to specify it in the CreateGroupRequest).
- WelcomeNote: a string that is presented as text to the user when he/she joins the group. The default value is an empty structure. This is an OPTIONAL property (the client does not have to specify it in the CreateGroupRequest).
- History:
  - T (message history is supported)
  - F (message history is not supported)

  If server supports the message history functionality, user/client can request it for a new or existing group.
- AutoDelete:
  - T (group will be automatically deleted when (1) all joined users have left; and (2) the validity “expires”.)
  - F (group will not be automatically deleted)
  - If not specified in CreateGroupRequest, the default value is “F”.
- Validity: a non-negative integer number that indicates the time period (in minutes) for which the group is valid. The default value is zero, which means that the group is not valid if and only if all joined user have left the group. The ‘Validity’ SHALL be ignored if ‘AutoDelete’ is ‘F’. Note that the generic XML element of the value of the group property is defined as a String. For this particular ‘Validity’ property, the String is the decimal representation of the non-negative integer number.

Each user MAY have his/her own properties for each group individually. These properties are:

[1] This property is read-only (determined by the server) and it cannot be modified.

[2] This property is read-only (monitored by the server) and it cannot be modified.

- PrivateMessaging:
  - T (sending private messages is enabled) or
  - F (sending private messages is disabled).

  Default value is “F”. The client does not have to specify it in the CreateGroupRequest.

- IsMember[3]:
  - T (the user is a member of the group) or
  - F (the user is not member of the group).
- PrivilegeLevel[4]:
  - User (general user),
  - Mod (moderator),
  - Admin (administrator).
- AutoJoin
  - T (server joins the client automatically to the particular group)
  - F (server does not join client automatically to the particular group)

  Default value is “F” for every user in every group.

- ShowID
  - T (server MAY disclosed the user ID to other users joined to this group)
  - F (server MAY NOT disclose the user ID to other users joined to this group)

  Default value is “F” for every user in every group.

The DTD is defined to allow custom group/own properties. The client and the server SHOULD ignore (without generating an error) the properties they are not able to process (not understood).

Both group members and joined users have own group properties. Properties of non-members MAY be discarded after the user leaves the group whereas members’ properties are kept on the server between separate group sessions.

10.2. Create group feature

10.2.1. Transactions

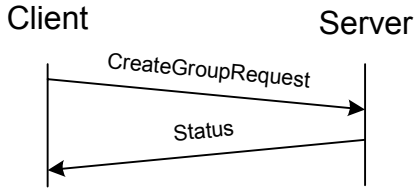

Figure 46. Create group transaction

[3] This property is read-only (determined by the server) and it cannot be modified.

[4] This property is read-only (determined by the server) and it cannot be modified.

A user MAY create a private user group at any time. The client sends the CreateGroupRequest message, which contains the name (ID), and the initial properties of the group. The server creates the group with the specified properties, and responds with a Status message. Optionally, the user can also join the newly created group.

10.2.2. Error Conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

CreateGroupRequest error conditions:

- Group already exists. (801)
- Invalid group attribute(s). (806)
- The maximum number of groups has been reached (user-limit). (814)
- The maximum number of groups has been reached (server-limit). (815)
- Cannot have searchable group without name or topic. (822)

10.2.3. Primitives and information elements

Primitive	Direction
CreateGroupRequest	Client → Server
Status	Client ← Server

Table 94. Primitive directions in create group transaction

Information Element	Req	Type	Description
Message-Type	M	CreateGroup	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Group-ID	M	String	Identifies the group to be created.
Group-Props	M	Structure	The properties of the group.
Join-Group	M	Boolean	A flag indicating that the user creating the group joins the group at the same time.
Screen-Name	O	Structure	Screen name of the user in the group.
Subscribe-Notif	M	Boolean	A flag indicating that the user wants to activate the group change notifications while joining the group.

Table 95. Information elements in CreateGroupRequest primitive

10.3. Delete group feature

10.3.1. Transactions

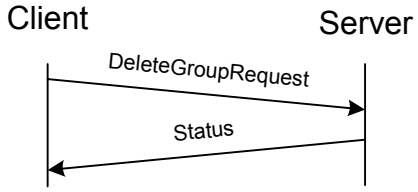

Figure 47. Delete group transaction

A user with sufficient access rights MAY delete a private user group at any time. The client sends the DeleteGroupRequest message, which contains the name (ID) of the group. The server SHOULD remove all currently joined users from the group (LeaveGroupResponse message), delete the specified group, and respond with a Status message.

10.3.2. Error Conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

DeleteGroupRequest error conditions:

- Group does not exist. (800)
- Group is public. (804)
- Insufficient user rights. (816)

10.3.3. Primitives and information elements

Primitive	Direction
DeleteGroupRequest	Client → Server
Status	Client ← Server

Table 96. Primitive directions in delete group transaction

Information Element	Req	Type	Description
Message-Type	M	DeleteGroup	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Group-ID	M	String	Identifies the group.

Table 97. Information elements in DeleteGroupRequest primitive

10.4. Join group feature

Since some users MAY be rejected and some groups MAY be restricted; a decision tree is provided to describe the behaviour:

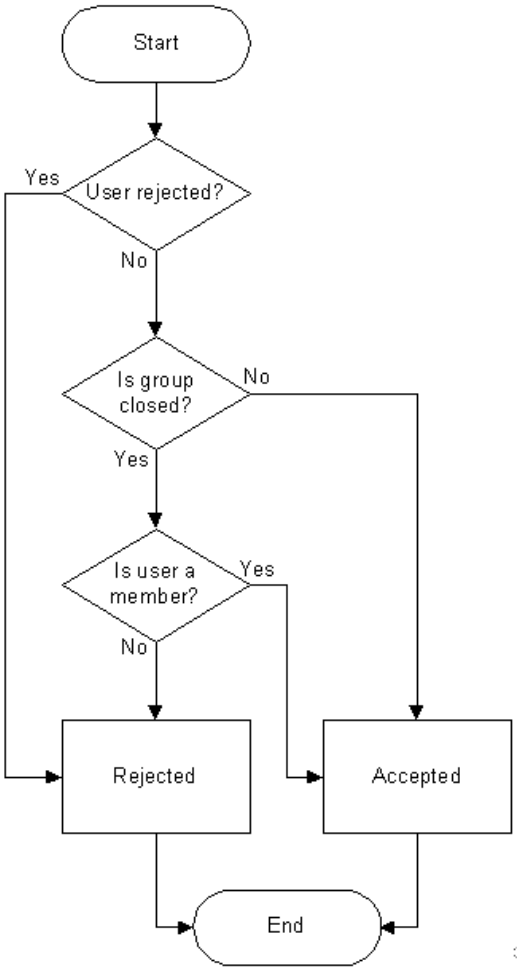

Figure 48. Joining decision-tree

10.4.1. Transactions

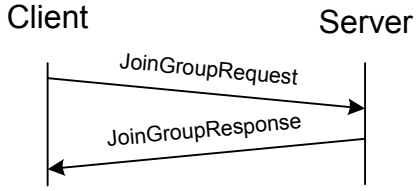

Figure 49. Join group transaction

A user MAY join a discussion group at any time. The client sends the JoinGroupRequest message to the server containing the ID of the group, his/her screen name shown during the discussion and the joined users' list request. The server responds with

the JoinGroupResponse message containing the list users, identified by their of screen names and user IDs of the currently joined users (if requested), and optionally a welcome note. If there is an error, the server responds with a Status message instead of the expected JoinGroupResponse message. The user-ID is present only if that user has set his ShowID own property to true.

After the user successfully joins the group, the user MAY receive and send messages from/to the particular group.

To retrieve previous messages (history) from the group, the get message list transaction MAY be utilized.

10.4.2. Error conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)
- Invalid/unsupported group properties (806)

JoinGroupRequest error conditions:

- Group does not exist. (800)
- Insufficient user rights. (816)
- User already joined. (807)
- User has been rejected. (809)
- Cannot join with the specified screen name; it is already in use. (811)
- The maximum number of allowed users has been reached. (817)

10.4.3. Primitives and information elements

Primitive	Direction
JoinGroupRequest	Client → Server
JoinGroupResponse	Client ← Server

Table 98. Primitive directions in join group transaction

Information Element	Req	Type	Description
Message-Type	M	JoinGroupRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Group-ID	M	String	Identifies the group in the transaction.
Screen-Name	O	Structure	Screen name of the user in the group.
Joined-Request	M	Boolean	Indicates if the user wants the list of currently joined users.
Subscribe-Notif	M	Boolean	A flag indicating that the user wants to activate the group change notifications while joining the group.
Own-Props	O	Structure	The list of the users' own group properties with the corresponding values.

Table 99. Information elements in JoinGroupRequest primitive

Information Element	Req	Type	Description
Message-Type	M	JoinGroupResponse	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Joined-Users-List	C	Structure	The list of the currently joined users (screen name, User ID). Present if it was requested.
Welcome-Text	O	Structure	A short text to be shown to the user when he/she has joined the group. The structure of the Welcome-Text includes { Content-type, OPTIONAL Content-encoding, Content-Data }.

Table 100. Information elements in JoinGroupResponse primitive

10.5. Leave group feature

10.5.1. Transactions

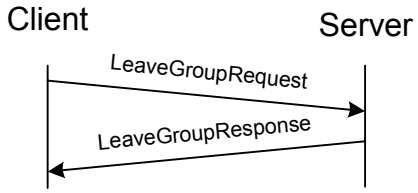

Figure 50. User initiated leave group transaction

A user MAY leave the joined discussion group at any time. The client sends the LeaveGroupRequest message to the server containing the ID of the group. The server responds with a LeaveGroupResponse message containing the reason code, which is own request is this case. If there is an error, the server responds with a Status message instead of the expected LeaveGroupResponse message.

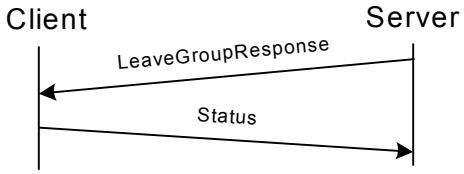

Figure 51. Server initiated leave group transaction

The server MAY initiate the group leaving also (user kicked out of the group, group deleted, etc.). In this case the server sends the LeaveGroupResponse message to the client containing the reason code.

After the user has left the group, the user cannot receive/send messages from/to the particular group.

10.5.2. Error conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)

- Service not supported. (405)

LeaveGroupRequest error conditions:

- Group was not joined before transaction. (808)

LeaveGroupResponse error conditions:

- Client MAY ignore any error and respond with Successful. (200)
- Group does not exist. (800)
- Own request (824)

10.5.3. Primitives and information elements

Primitive	Direction
LeaveGroupRequest	Client → Server
LeaveGroupResponse	Client ← Server

Table 101. Primitive directions in leave group transaction

Information Element	Req	Type	Description
Message-Type	M	LeaveGroup Request	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Group-ID	M	String	Identifies the requested content.

Table 102. Information elements in LeaveGroupRequest primitive

Information Element	Req	Type	Description
Message-Type	M	LeaveGroup Response	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Result	M	Structure	Result that indicates why did the user leave the group. (Own request, rejected, etc.)
Group-ID	C	String	Identification of the- group that has been left. Not present if the client initiated the transaction.

Table 103. Information elements in LeaveGroupResponse primitive

10.6. Members' list management

10.6.1. Transactions

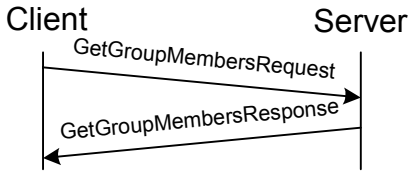

Figure 52. Get group members transaction

A user with sufficient access rights MAY retrieve the member list of a group. The client sends the GetGroupMembersRequest message to the server, which contains the ID of the group. The server responds with the GroupMembersResponse message, which contains the list of all group members. If there is an error, the server responds with a Status message instead of the expected GroupMembersResponse message.

Figure 53. Get joined users transaction

A user MAY retrieve the list of users that have currently joined the group. The client sends the GetJoinedUsersRequest message to the server, which contains the ID of the group. The server responds with the GetJoinedUsersResponse message, which contains the list of users that have currently joined the group. The content of the GetJoinedUsersResponse depends on the privileges of the requesting user. Administrators and Moderators get the list of users divided into privilege categories; ordinary users get a list of all joined users without such division. Administrators and moderators receive the User-ID with each screen-name. Ordinary users MAY receive the User-ID as well, but only for those users that have the ShowID own group property turned on (true). Only joined users MAY retrieve this list.

If there is some error, the server responds with a Status message instead of the expected GetJoinedUsersResponse message.

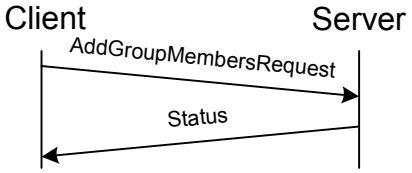

Figure 54. Add group members transaction

A user with sufficient access rights MAY add user(s) to the member list of a group. The client sends the AddGroupMembersRequest message to the server, which contains the ID of the group, and the list(s) of users to be added. The server responds with the Status message.

All of the newly added users have the lowest privilege level: User.

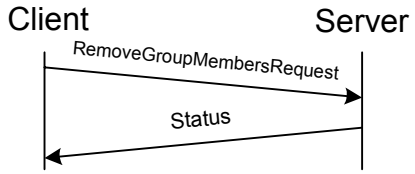

Figure 55. Remove group members transaction

A user with sufficient access rights MAY remove user(s) from the member list of a group. The client sends the RemoveGroupMembersRequest message to the server, which contains the ID of the group, and the list of users to be removed. The server responds with the Status message.

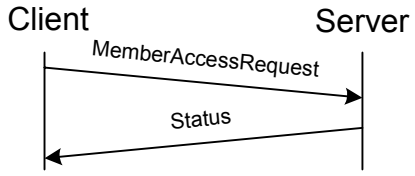

Figure 56. Member access rights transactions

A user with sufficient access rights MAY change the access privileges of user(s). The client sends the MemberAccessRequest message to the server, which contains the ID of the group, and optionally the list of users to be set as administrator, moderator or ordinary user. The server responds with a Status message.

Note for clarification: Being a group member does not have anything to do with being joined. Only members are allowed to join a restricted group.

10.6.2. Error Conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

GetGroupMembersRequest error conditions:

- Group does not exist. (800)
- Group was not joined before transaction. (808)
- Insufficient user rights. (816)

GetJoinedUsersRequest error conditions:

- Group does not exist. (800)
- Group was not joined before transaction. (808)

AddGroupMembersRequest error conditions:

- Group does not exist. (800)
- Insufficient user rights. (816)
- Unknown user. (531)
- The maximum number of group members has been reached (823)

RemoveGroupMembersRequest error conditions:

- Group does not exist. (800)
- Insufficient user rights. (816)

- Unknown user. (531)

MemberAccessRequest error conditions:

- Group does not exist. (800)
- Insufficient user rights. (816)
- Unknown user. (531)
- Not a group member (810)

10.6.3. Primitives and information elements

Primitive	Direction
GetGroupMembersRequest	Client → Server
GetGroupMembersResponse	Client ← Server
GetJoinedUsersRequest	Client → Server
GetJoinedUsersResponse	Client ← Server
AddGroupMembersRequest	Client → Server
Status	Client ← Server
RemoveGroupMembersRequest	Client → Server
Status	Client ← Server
MemberAccessRequest	Client → Server
Status	Client ← Server

Table 104. Primitive directions in add/remove user(s) to/from group transaction

Information Element	Req	Type	Description
Message-Type	M	GetGroupMembers Request	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Group-ID	M	String	Identifies the group.

Table 105. Information elements in GetGroupMembersRequest primitive

Information Element	Req	Type	Description
Message-Type	M	GetGroupMembers Response	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
User-List-Adm	O	Structure	The list of users that are Administrators.
User-List-Mod	O	Structure	The list of users that are Moderators.
User-List	O	Structure	The list of users that are ordinary Users.

Table 106. Information elements in GetGroupMembersResponse primitive

Information Element	Req	Type	Description
Message-Type	M	GetJoinedUsers Request	Message identifier.

Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Group-ID	M	String	Identifies the group.

Table 107. Information elements in GetJoinedUsersRequest primitive

Information Element	Req	Type	Description
Message-Type	M	GetJoinedUsers Response	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Admin-Map-List	C	Structure	Administrators and moderators receive this list.
User-Map-List	C	Structure	Ordinary users receive this list.

Table 108. Information elements in GetJoinedUsersResponse primitive

Information Element	Req	Type	Description
Message-Type	M	AddGroupMembers Request	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Group-ID	M	String	Identification of the group.
User-List	M	Structure	The list of users to be added to the members' list.

Table 109. Information elements in AddGroupMembersRequest primitive

Information Element	Req	Type	Description
Message-Type	M	RemoveGroup MembersRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Group-ID	M	String	Identification of the group.
User-List	M	Structure	The list of members to be removed from the group specified by user-ID.

Table 110. Information elements in RemoveGroupMembersRequest primitive

Information Element	Req	Type	Description
Message-Type	M	MemberAccess Request	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Group-ID	M	Group-ID	Identifies the group.
User-List-Admin	O	Structure	The list of users that are requested to be set as administrators.
User-List-Mod	O	Structure	The list of users that are requested to be set as moderators.
User-List	O	Structure	The list of users that are requested to be set as users.

Table 111. Information elements in MemberAccessRequest primitive

10.7. Modify group properties

10.7.1. Transactions

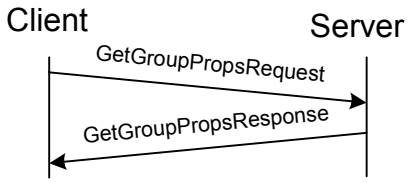

Figure 57. Get group properties transaction

A user with sufficient access rights MAY retrieve the properties of a group, and his/her own properties for that particular group. The client sends the GetGroupPropsRequest to the server, which contains the ID of the group. The server responds with the GetGroupPropsResponse message, which contains the properties of the specified group. If there is an error, the server responds with a Status message instead of the expected GetGroupPropsResponse message.

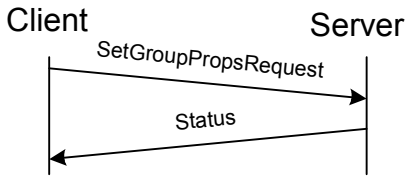

Figure 58. Set group properties transaction

A user with sufficient access rights MAY update the properties of a group, or his/her own properties for that particular group. The client sends the SetGroupPropsRequest message to the server, which contains the ID, the new properties of the group and/or the new user properties. The server responds with a Status message.

10.7.2. Error Conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

GetGroupPropsRequest error conditions:

- Group does not exist. (800)
- Insufficient user rights. (816)

SetGroupPropsRequest error conditions:

- Group does not exist. (800)
- Insufficient user rights. (816)
- Invalid group attribute(s). (806)
- Cannot have searchable group without name or topic. (822)

10.7.3. Primitives and information elements

Primitive	Direction
GetGroupPropsRequest	Client → Server
GetGroupPropsResponse	Client ← Server
SetGroupPropsRequest	Client → Server
Status	Client ← Server

Table 112. Primitive directions in modify group properties transaction

Information Element	Req	Type	Description
Message-Type	M	GetGroupProps Request	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Group-ID	M	String	Identification of the group.

Table 113. Information elements in GetGroupPropsRequest primitive

Information Element	Req	Type	Description
Message-Type	M	GetGroupProps Response	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Group-Props	M	Structure	The list of group properties with the corresponding values.
Own-Props	M	Structure	The list of the users' own group properties with the corresponding values.

Table 114. Information elements in GetGroupPropsResponse primitive

Information Element	Req	Type	Description
Message-Type	M	SetGroupProps Request	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Group-ID	M	String	Identification of the group.
Group-Props	O	Structure	The new list of group properties with the corresponding values.
Own-Props	O	Structure	The list of the users' own group properties with the corresponding values.

Table 115. Information elements in SetGroupPropsRequest primitive

10.8. Rejecting user(s) from group feature

The concept of *rejecting* means kicking the user out of the group (if joined) and disabling certain features in the group.

Rejecting is active for the rejected user(s) until another user with sufficient privileges removes him/her from the rejected users' list.

10.8.1. Transactions

Figure 59. Rejected list transactions

A user with sufficient access rights MAY retrieve/update the reject list of a group. The client sends the RejectListRequest message to the server, which contains the ID of the group, and optionally the users to be added/removed to/from the reject list. The server responds with the RejectListResponse message, which contains the list of users that are rejected. If there is an error, the server responds with a Status message instead of the expected RejectListResponse message.

Users that are active members in a group (e.g. joined) SHOULD be removed (leave) the group when they are rejected from the group.

Users on the reject list cannot join the group.

Users in the reject list are specified by their User-ID.

10.8.2. Error conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

RejectListRequest error conditions:

- User unknown. (531)
- Group does not exist. (800)
- Insufficient user rights. (816)

10.8.3. Primitives and information elements

Primitive	Direction
RejectListRequest	Client → Server
RejectListResponse	Client ← Server

Table 116. Primitive directions in rejected list transaction

Information Element	Req	Type	Description
Message-Type	M	RejectListRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Group-ID	M	String	Identification of the group.
Add-Users-List	O	Structure	The list of users that SHOULD be added to the reject list.

Remove-Users-List	O	Structure	The list of users that SHOULD be removed from the reject list.

Table 117. Information elements in RejectListRequest primitive

Information Element	Req	Type	Description
Message-Type	M	RejectListResponse	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
User-List	O	Structure	The list of users that are in the reject list.

Table 118. Information elements in RejectListResponse primitive

10.9. Subscribe to group change

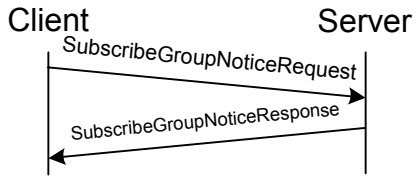

Figure 60. Subscribe group change notification transaction

A user MAY get/set/unset group change subscription status. The client sends the SubscribeGroupNoticeRequest message to the server. The message contains the type of the requested operation. The answer from the server for the get operation is the SubscribeGroupNoticeResponse message, or Status if an error occurs. The answer from the server for the set/unset operation is a Status message. While the subscription is active, the user will receive GroupChangeNotice messages.

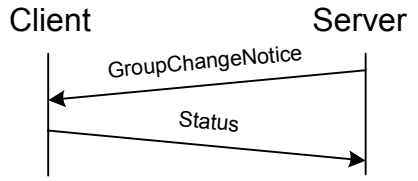

Figure 61. Group change notification

The server MAY send group change notification(s) to the user whenever some other user leaves or joins the group, or the group properties or the user's own properties have been changed. The server sends the GroupChangeNotice message to the users (whose group change subscription is active) containing a list of users, identified by their screen names and user IDs of the recently joined or left users, or the new properties of the group. The user ID is present only if that user has set his ShowID own property to true.

10.9.1. Error Conditions

Generic error conditions:

- Not logged in. (604)
- Service not agreed. (506)
- Service not supported. (405)

SubscribeGroupNoticeRequest error conditions:

- Group does not exist. (800)
- Group was not joined before transaction. (808)

GroupChangeNotice error conditions:

- Client MAY ignore any error and respond with Successful. (200)

10.9.2. Primitives and information elements

Primitive	Direction
SubscribeGroupNoticeRequest	Client → Server
SubscribeGroupNoticeResponse	Client ← Server
GroupChangeNotice	Client ← Server
Status	Client → Server

Table 119. Primitive directions in subscribe group change notification transaction

Information Element	Req	Type	Description
Message-Type	M	SubscribeGroup NoticeRequest	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Group-ID	M	String	Identification of the group.
Subscribe-Type	M	Enumerated character	Indicates the type of subscription request. (“G” for Get, “S” for Set, and “U” for Unset.)

Table 120. Information elements in SubcribeGroupNoticeRequest primitive

Information Element	Req	Type	Description
Message-Type	M	SubscribeGroup NoticeResponse	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Subscription-State	M	Boolean	Indicates the status of subscription.

Table 121. Information elements in SubcribeGroupNoticeResponse primitive

Information Element	Req	Type	Description
Message-Type	M	GroupChange Notice	Message identifier.
Transaction-ID	M	String	Identifies the transaction requested.
Session-ID	M	String	Identifies the session.
Group-ID	M	String	Identification of the group.
Joined-Users-List	O	Structure	A list of the users that recently joined the group (screen name, user ID).
Left-Users-List	O	Structure	A list of screen names of the users that recently left the group.
Group-Props	O	Structure	The new properties of the group.
Own-Props	O	Structure	The new properties of the user in the group.

Table 122. Information elements in GroupChangeNotice primitive

11. Status Codes and Descriptions

CSP uses the concept and paradigm of HTTP/1.1 response to define the status code. However, there is no logical or semantic relationship between the status codes in CSP and the status codes in HTTP.

The following sections define the general categories as well as each status code.

11.1. 1xx – Informational

The client MUST be prepared to accept one or more 1xx status codes prior to a regular response even if the client does not expect a 100 "Continue" status code. A user agent SHALL ignore unexpected 1xx status code. This category of the status codes does not finish a transaction.

11.1.1. 100 – Continue

The client SHOULD continue with its request. The server has accepted the request for processing, but the processing has not been completed. The request might or might not eventually be successfully completed. The server MUST send a final response again upon completing the request. The "100" response is used when time of completion will be too long, possibly causing the server and client connection to break.

11.1.2. 101 – Queued

The client SHOULD continue with its request. The server has accepted the request, but does not have resources to start processing. The request might or might not eventually be successfully completed. The server MUST send a final response again upon completing the request.

11.1.3. 102 – Started

The client SHOULD continue with its request. The server has accepted the request for processing. The "102" response is used when server needs to start additional transactions in order to process the request. The server MUST send a final response again upon completing the request.

11.2. 2xx – Successful

The 2xx class of status codes indicates that the client's request was successfully received, understood and accepted.

11.2.1. 200 – Successful

This is used to indicate that the request succeeded.

11.2.2. 201 – Partially successful

This is used to indicate that the request was successfully completed, but some parts were not completed due to certain errors. The details of the error case(s) are indicated in the response.

11.2.3. 202 – Accepted

This is used to indicate that server accepted the request, but not able to receive acknowledgment about delivery to client device. The request might or might not eventually be acted upon. There is no facility for re-sending a status code from an asynchronous operation such as this.

11.3. 3xx – Redirection

The 3xx class of status codes indicates that further action needs to be taken by the user agent in order to fulfill the request.

11.4. 4xx – Client Error

The 4xx class of status codes is intended for cases in which the client seems to have erred. The server SHOULD include the explanation of the error situation including whether it is a temporary or permanent condition. The user agents SHOULD be able to display the error description to the user.

11.4.1. 400 – Bad Request

The server could not understand the request due to the malformed syntax. The client MAY NOT repeat the request without modification.

11.4.2. 401 – Unauthorized

When an authorization request is expected, the presence server will respond with this status code. Properties will contain details of available authorization schemes.

11.4.3. 402 – Bad Parameter

The server cannot understand one of the parameters in the request. The client MAY NOT repeat the request without modification.

11.4.4. 403 – Forbidden

The server understood the request, but the principal settings denied access to some of the presence, contact information or group. Authorization will not help and the request SHOULD NOT be repeated. This type of response can be returned if user not login in the network yet.

11.4.5. 404 – Not Found

The server cannot find anything matching the request. No indication is given of whether the condition is temporary or permanent.

11.4.6. 405 – Service Not Supported

The server does not support the service method in the request.

11.4.7. 408 – Request Timeout

The client did not produce a request within the time the server was prepared to wait.

11.4.8. 409 – Invalid password

The password provided by the client was incorrect; it does not match with the given user-ID. The client MAY NOT repeat the request without modification.

11.4.9. 410 – Unable to Deliver

The server cannot deliver the request. The requested resource is no longer available at the server and no forwarding address is known.

11.4.10. 415 – Unsupported Media Type

The server cannot deliver the request because the client cannot support the format of the entity that it requested.

11.4.11. 420 – Invalid Transaction ID

The server encountered an invalid transaction ID.

11.4.12. 422 – UserID and ClientID do not match

The UserID and the ClientID do not match in the request.

11.4.13. 423 – Invalid Invitation-ID

The server encountered an invalid invitation ID.

11.4.14. 424 – Invalid Search-ID

The server encountered an invalid search ID.

11.4.15. 425 – Invalid Search-Index

The server encountered an invalid search index.

11.4.16. 426 – Invalid Message-ID

The server encountered an invalid message ID.

11.4.17. 431 – Unauthorized Group Membership

The user agent is not an authorized member of the group.

11.4.18. 432 – Response too large

The response would be larger than the amount agreed in client capability parameter “ParserSize”.

11.5. 5xx – Server Error

The 5xx class of status codes is intended for cases in which the server is aware that it has erred or is incapable of performing the request.

11.5.1. 500 – Internal server or network error

The server encountered an unexpected condition that prevented it from fulfilling the request.

11.5.2. 501 – Not Implemented

The server does not support the functionality REQUIRED to fulfil the request. This is the appropriate response when the server does not recognize the request method, and it is not capable of supporting it for any resources.

11.5.3. 503 – Service Unavailable

The server is currently unable to handle the request due to a temporary overloading or maintenance of the server.

11.5.4. 504 – Timeout

The server could not produce a response within the time that it expected.

11.5.5. 505 – Version Not Supported

The server does not support, or refuse to support, the request version that was used.

11.5.6. 506 – Service not agreed

During service negotiation the server did not agree to provide the transaction that the client requests. The client MAY NOT repeat the request without a new service negotiation.

11.5.7. 507 – Message queue is full

The server cannot fulfil the request, because its message queue is full. The client MAY repeat the request.

11.5.8. 516 – Domain Not Supported

The server does not support forwarding to a different domain space.

11.5.9. 521 – Unresponded Presence Request

The presence information provider does not respond to the presence service specified in the request.

11.5.10. 522 – Unresponded Group Request

The group service provider does not respond to the requested group transaction.

11.5.11. 531 – Unknown user

The specified user is unknown/not valid userID was given.

11.5.12. 532 – Recipient Blocked the Sender

The recipient of the message or invitation blocked the sender. Note that returning this error code reveals to the sender that the recipient has blocked it. It is up to the implementation and service provider to decide whether or not this error code SHOULD be returned. A WV server MAY instead report success, even though the message or invitation was discarded, to conceal this fact.

11.5.13. 533 – Message Recipient Not Logged in

The recipient of the message is not logged in.

11.5.14. 534 – Message Recipient Unauthorized

The recipient of the message is not authorized.

11.5.15. 535 – Search timed out

The server has invalidated the requested search-request.

11.5.16. 536 – Too many hits

The server performed the search successfully, but the server implementation limits the maximum number of hits – the server MAY discard the hits that are over its limits and the discard hits will not be available for the client.

11.5.17. 537 – Too broad search criteria

The server did not perform the search – the server simply by analyzing the search criteria came to the conclusion that according to the requested criteria the search would give higher number of hits than the server is willing to handle. The client MAY repeat the request with narrowed search criteria.

11.5.18. 538 – Message has been rejected

Recipient has rejected message. Note that returning this error code reveals to the sender that the recipient has rejected the message. It is up to the implementation and service provider to decide whether or not this error code SHOULD be returned. A WV server MAY instead report success, even though the message was discarded, to conceal this fact.

11.5.19. 540 – Header encoding not supported

The requested SMS header encoding (UDH or textual) is not supported. The clients MUST NOT repeat the request without modification. The client MAY repeat the request with the opposite header encoding (UDH if it was textual, or vice versa).

11.5.20. 541 – Message has been forwarded

Recipient has forwarded message without retrieving it first. Note that returning this error code reveals to the sender that the recipient has forwarded the message. It is up to the implementation and service provider to decide whether or not this error code SHOULD be returned. A WV server MAY instead report success, even though the message was forwarded, to conceal this fact.

11.5.21. 542 – Message has expired

Message has not been retrieved by the recipient in the specified time period and has been deleted from the server.

11.5.22. 543 – No matching digest scheme supported

The server does not support any of the digest schemas that the client has requested.

11.6. 6xx – Session

The 6xx class status code indicates the session-related status.

11.6.1. 600 – Session Expired

The client was disconnected because time-to-live parameter of user session has expired.

11.6.2. 601 – Forced Logout

The server has disconnected the client.

11.6.3. 603 – Already Logged in

The server will not accept new login request from the client, because the client already logged in. Such behavior of the server is not RECOMMENDED

11.6.4. 604 – Invalid session (not logged in).

There is no such session. (Previously not logged in, disconnected, or logged out.)

11.6.5. 605 – New value not accepted.

The server does not accept the new timeout value requested by the client, the old value MUST be used.

11.7. 7xx – Presence and contact list

The 7xx class indicates the presence and contact list related status codes.

11.7.1. 700 – Contact list does not exist

The contact list specified in the request does not exist.

11.7.2. 701 – Contact list already exists

The contact list specified in the request already exists.

11.7.3. 702 – Invalid or unsupported user properties

The user properties specified in the request are invalid, or not supported.

11.7.4. 750 – Invalid or unsupported presence attribute

The presence attribute(s) specified in the request are invalid, or not supported.

11.7.5. 751 – Invalid or unsupported presence value

The presence value(s) specified in the request are invalid, or not supported. The client SHOULD NOT repeat the request without modification.

11.7.6. 752 – Invalid or unsupported contact list property

One or more contact list properties specified in the request are invalid or not supported. The client SHOULD NOT repeat the request without modification.

11.7.7. 753 – The maximum number of contact lists has been reached for the user

The server limits the maximum number of contact lists per user. The limit has been reached; so additional contact lists cannot be created. The client SHOULD NOT repeat the request until a contact list that belongs to the particular user has been deleted.

11.7.8. 754 – The maximum number of contacts has been reached for the user

The server limits the maximum number of contacts per user. The limit has been reached; so additional contacts cannot be created. The client SHOULD NOT repeat the request until a contact that belongs to the particular user has been deleted.

11.7.9. 755 – The maximum number of attribute lists has been reached for the user

The server limits the maximum number of attribute lists per user. The limit has been reached; so additional attribute lists cannot be created. The client SHOULD NOT repeat the request until an attribute list that belongs to the particular user has been deleted.

11.7.10. 760 – Automatic Subscription / Un-subscription is not supported

The server does not support the automatic subscription when adding a user to the contact list, and does not support the automatic un-subscription when deleting the contact list or removing a user from the contact list.

11.8. 8xx – Groups

The 8xx class indicates the group-related status codes.

11.8.1. 800 – Group does not exist

The group specified in the request does not exist.

11.8.2. 801 – Group already exists

The group specified in the request already exists.

11.8.3. 802 – Group is open

The group specified in the request is an open group.

11.8.4. 803 – Group is restricted

The group specified in the request is a closed group.

11.8.5. 804 – Group is public

The group specified in the request is public.

11.8.6. 805 – Group private

The group specified in the request is private.

11.8.7. 806 – Invalid/unsupported group properties

The group properties specified in the request are invalid or not supported.

11.8.8. 807 – Group is already joined

The group specified in the request is already joined. If the server does not allow the same user to join a group more than once then this error code is used to indicate that the user is already joined the particular group.

11.8.9. 808 – Group is not joined

The request cannot be processed because it requires the user to be joined to the group.

11.8.10. 809 – User has been rejected

The user has been rejected from the particular group. He/she is forced to leave the group and cannot join.

11.8.11. 810 – Not a group member

The request cannot be processed because the user is not a member of the specified restricted group The client SHOULD NOT repeat the request until the user has been added to the group as a member.

11.8.12. 811 – Screen name already in use

The screen name specified in the request is already in use. If the server does not allow the same screen name to be used in a group more than once then this error code is used to indicate that the screen name is already in use. The requesting user MAY try to change his/her screen name and repeat the transaction.

11.8.13. 812 – Private messaging is disabled for group

The client requested private message delivery but the private messaging is disabled in the particular group.

11.8.14. 813 – Private messaging is disabled for user

The client requested private message delivery but the private messaging is disabled for the particular user.

11.8.15. 814 – The maximum number of groups has been reached for the user

The server limits the maximum number of groups per user. The limit has been reached; additional groups cannot be created. The client SHOULD NOT repeat the request until a group that belongs to the particular user has been deleted.

11.8.16. 815 – The maximum number of groups has been reached for the server

The maximum number of groups is limited on the server. The server limit has been reached; additional groups cannot be created. The client MAY repeat the request.

11.8.17. 816 – Insufficient group privileges

The user is a member in the particular group, but does not have sufficient privileges group to perform the requested operation. The client SHOULD NOT repeat the request until the user has been authorized properly.

11.8.18. 817 – The maximum number of joined users has been reached

The maximum number of joined users has been reached in the requested group. The client MAY repeat the request.

11.8.19. 821 – History is not supported.

The server does not support group history. The client MAY NOT repeat the request.

11.8.20. 822 – Cannot have searchable group without name or topic

The server cannot perform group search without group name or group topic. Either group name or group topic or both MUST be non-empty to support group search.

11.8.21. 823 – The maximum number of group members has been reached

The server limits the maximum number of group members per group. The limit has been reached; so additional group members cannot be added. The client SHOULD NOT repeat the request until a group member has been removed from the group.

11.8.22. 824 – Own Request

The reason code for the LeaveGroupResponse. Server sends this error code in the LeaveGroupResponse as a response to the client initiated LeaveGroupRequest.

11.9. 9xx General Errors

The 9xx class indicates status codes too general to fit into other classes.

11.9.1. 900 Multiple Errors

No part of the transaction was successfully processed for several reasons and thus one other status code cannot indicate the errors. The details of the error cases are indicated in the response.

11.9.2. 901 General Address Error

The general address is not supported. No specific error is given due to security or privacy reason.

11.9.3. 902 – Not enough credit to complete requested operation

The server cannot perform the requested operation since the user has not enough credit.

11.9.4. 903 – Operation requires a higher class of service

The server cannot perform the requested operation since it requires a higher class of service. A class of service is a designation assigned by the service provider to describe the service treatment and privileges given to a particular user (e.g., premium, gold).

12. Extension Framework

CSP defines a framework that can be used by different vendors to extend the standard Wireless-Village protocol. This framework includes support for 2 basic operations:

- Extending existing primitives
- Introducing new primitives

It is RECOMMENDED that applications use the version discovery transaction for detecting which extended capabilities the server supports and the service negotiation for negotiating these capabilities. The details of how to achieve these tasks are outside the scope of the Wireless-Village protocol.

Both clients and servers MUST ignore blocks with unrecognised namespaces.

12.1. Extending Existing Primitives

Extended blocks MAY be appended to existing primitives, designated with a namespace.

12.2. Introducing New Primitives

The extension framework defines general primitives that serve as an envelope for proprietary operations. These primitives obey all rules set for the normal primitives with 1 exception:

- The actual content of these primitives are not defined.

13. Static Conformance Requirements for CSP

The static conformance requirements for this specification are specified in [CSP SCR].

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-WV-CSP-V1_1-20021001-A	01 Oct 2002	Version 1.1

A.2 Draft/Candidate Version 1.2 History

Document Identifier	Date	Sections	Description
Candidate Versions OMA-IMPS-WV-CSP-V1_2	02 Feb 2003	n/a	Status changed to Candidate by TP TP ref # OMA-TP-2003-0109-IMPS-V1_2-Candidate-Package
	03 May 2004	All	Typo corrections. Words capitalized according to RFC [2119]. Added CRs: - OMA-IMPS-2003-0107-COMVERSE-BugFix1/2232 - OMA-MWG-IM-2003-0017-Missing_Domain_element - OMA-IMPS-2003-0045-CSP_DTD_CR_2/2225 - OMA-MWG-IM-2003-0023-CR_CSP_GETJU_ServiceTree - OMA-IMPS-2003-0198R02-ParserSizeFix - OMA-IMPS-2003-0207-WrongCaptions - OMA-MWG-IM-2003-0003-GetPresenceRequest - OMA-IM-2004-0036-KeyCap - OMA-IM-2004-0042-UnPreClar - OMA-IM-2004-0079R01-CR-DTD-inconsistency-in-BlockListRequest