WV-046 Client-Server Protocol SMS Binding

Version 1.2

Version 2003-Feb-21

Open Mobile Alliance
OMA-IMPS-WV-CSP_SMS-v1_2-20030221-C

Continues the Technical Activities
Originated in the Wireless Village Initiative

A list of errata and updates to this document is available from the Open Mobile Alliance™ Web site, http://www.openmobilealliance.org/, in the form of SIN documents, which are subject to revision or removal without notice.

This document is available online in PDF format at http://www.openmobilealliance.org/.

Known problems associated with this document are published at http://www.openmobilealliance.org/.

Comments regarding this document can be submitted to the Open Mobile Alliance™ in the manner published at http://www.openmobilealliance.org/documents.asp

Document History	
Accepted all TRACK CHANGES in OMA-IMPS-WV-CSP_SMS-V1_2-20021220-A.doc	2003-01-17
Updated URLs in References	2003-01-17
OMA-IMPS-WV-CSP_SMS-v1_2-20030117-D	Draft
OMA-IMPS-WV-CSP_SMS-v1_2-20030221-C	Current

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 SCR] "WV-048 Client-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-CSP_SCR-V1_2-20030117-D.pdf

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels". S. Bradner. March 1997. URL:http://www.ietf.org/rfc/rfc2119.txt

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

[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

[TS 23.038] "Alphabets and Language-specific Information (Release 5), 3GPP TS 23.038 v5.0.0", 3rd Generation Partnership Project, March 2002. URL:ftp://ftp.3gpp.org/Specs/archive/23_series/23.038/23038-500.zip

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

2.2. Informative References

[Arch] "WV-040 System Architecture Model Version 1.2". Open Mobile Alliance. December 2002. URL:http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-Arch-V1_2-20030117-D.pdf

[FeaFun] "WV-041 Features and Functions Version 1.2". Open Mobile Alliance. December 2002. URL:http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-Features_Functions-V1_2-20030117-D.pdf

[CSP] "WV-042 Client-Server Protocol Session and Transactions Version 1.2". Open Mobile Alliance. December 2002. URL:http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-CSP-V1_2-20030117-D.pdf

[CSP DTD] "WV-043 Client-Server Protocol DTD and Examples Version 1.2". Open Mobile Alliance. December 2002. URL:http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-CSP_DTD-V1_2-20030117-D.pdf

[CSP Trans] "WV-044 Client-Server Protocol Transport Bindings Version 1.2". Open Mobile Alliance. December 2002. URL:http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-CSP_Transport-V1_2-20030117-D.pdf

[CSP DataType] "WV-045 Client-Server Protocol Data Types Version 1.2". Open Mobile Alliance. December 2002. URL:http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-CSP_DataTypes-V1_2-20030117-D.pdf

[CSP SMS] "WV-046 Client-Server Protocol SMS Binding Version 1.2". Open Mobile Alliance. December 2002. URL:http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-CSP_SMS-V1_2-20030117-D.pdf

[CSP WBXML] "WV-047 Client-Server Protocol Binary Definition and Examples Version 1.2". Open Mobile Alliance. December 2002. URL:http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-CSP_WBXML-V1_2-20030117-D.pdf

[CSP SCR] "WV-048 Client-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-CSP_SCR-V1_2-20030117-D.pdf

[PA] "WV-049 Presence Attributes Version 1.2". Open Mobile Alliance. December 2002. URL:http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-PA-V1_2-20030117-D.pdf

[PA DTD] "WV-050 Presence Attribute DTD and Examples Version 1.2". Open Mobile Alliance. December 2002. URL:http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-PA_DTD-V1_2-20030117-D.pdf

[CLP] "WV-051 Command Line Protocol Version 1.2". Open Mobile Alliance. December 2002. URL:http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-CLP-V1_2-20030117-D.pdf

[SSP] "WV-052 SSP - Server-Server Protocol Semantics Document Version 1.2". Open Mobile Alliance. December 2002. URL:http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-SSP-V1_2-20030117-D.pdf

[SSP Syntax] "WV-053 Server-Server Protocol XML Syntax Document Version 1.2". Open Mobile Alliance. December 2002. URL:http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-SSP_Syntax-V1_2-20030117-D.pdf

[SSP Trans] "WV-054 SSP - Transport Binding Version 1.2". Open Mobile Alliance. December 2002. URL:http://www.openmobilealliance.org/member/technicalPlenary/imps/docs/OMA-IMPS-WV-SSP_Transport-V1_2-20030117-D.pdf

[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

[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

WV	Wireless Village

4. Introduction

This document describes how the SMS can be utilized to provide WV functionality.

The message flows are defined in [CSP].

The enumerated values are as specified in [CSP].

5. SMS binding

5.1. Syntax

General SMS message format consists of preamble and parameters as follows:

WVaaBBcccDD <parameters> [&] where:

- **WV** indicates that this is a Wireless Village message. It is case sensitive.
- **aa** is the version number of the IM specification. Major version and minor version numbers without the dot in the middle, both are single digits from 0 to 9. This specification uses version number 1.2 (12 in messages).
- **BB** is the message type, identified by a two-letter code. It is not case sensitive.
- The **ccc** is the Transaction-ID in range 0999 without preceding zero. The server initiated Disconnect primitive contains transaction ID as well, but the client must ignore this ID.
- The **DD** identifies multiple short messages within single transaction; it is a two-letter identifier within the range of a-z ('a' being the first). The first letter indicates the number (position) of the message; the second letter indicates the total number of partS. (This gives the possibility to split a message into 26 SMSes, which limits the size of the message to 26*160=4160 characters. Note that it includes every additional information elements such as tags, commas and so on, so not the whole capacity is available to the user.) It is not case sensitive.
- The ampersand (&) parameter separates multiple WV messages within short message(s). The separating ampersand does not follow the last WV message in the short message.

The general format of parameters with value SHALL be:

<name>=<value>

Example: CI=+1234567890

The general format of parameters without value SHALL be:

<name> Note the missing equal sign and value.

Example: PS

After the *aaBBcccDD* preamble, there is one space character. Each parameter is separated from other parameters by single space character. Every parameter is defined and used with its own unique two-letter code.

If the value of the parameter contains spaces ( ), quotes (”), commas (,),parentheses (()),equal (=) or ampersand (&) characters, it SHALL be wrapped with quotes (“).

If the value contains quotes (“), all quote characters SHALL be doubled. For example if the value is:

John “Johnnie” Smith

then it SHALL be encoded as:

“John “”Johnnie”” Smith”

If the value is only one single quote, the encoding would be four quotes: “””””

Each parameter MAY be present only once in a primitive, so a parameter MAY also contain a list of values. In this case, the syntax SHALL be (example with three values):

<name>=(<value1>,<value2>,<value3>)

Single values MUST NOT be wrapped with parentheses, unless the syntax explicitly specifies that (screen name for example). Please refer to chapter 7.11 Information element-specific encoding on page 25 for exceptions.

A parameter MAY also contain a group of values. These groups MAY be also nested. Examples:

one value having three values (a parent having 3 children):

<name>=(<value1>,(<value2>,<value3>,<value4>))

nested values (root node has a single child, which has two children):

<name>=(<value1>,(<value2>,(<value3>,<value4>)))

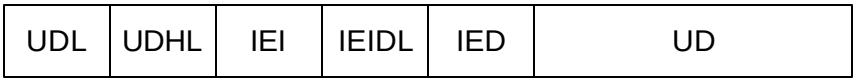
Definitions SHALL be case-sensitive.

5.2. Encoding of Short Messages

The encoding of SMS messages has two alternatives: 7-bit encoding using the SMS default character set [TS 23.038] and 8-bit encoding using UTF-8 character encoding. With 8-bit encoding, the minimum supported set of characters is ISO-8859-1 (Latin 1).

Each WV short message MAY be either textual or contain a User Data Header (UDH) with the TP-UDHI value set to 1. In case of UDH, the detection of the WV-primitives is based on a registered WV application port identified in the headers. In case of textual short message, the detection of WV-primitives is based on detection of the message preamble. The User Data Header contains a 16-bit application port number identifying the source port and destination port. The structure of the short message with UDH is defined as follows [TS 23.040]:

The destination port number is assigned in the CSP transport-binding document [CSP Trans]. When the session is started with a login primitive containing the UDH, the WV server and client must continue the session using the messages encoded with UDH throughout the session. Similarly, when the session is started with a login primitive without UDH, the WV server and client must continue the session without UDH.



UDL	Length of the message
UDHL	Length of User Data Header (7 for WV message)
IEID	Information Element Identifier (05_{hex} = 16-bit application port)
IEDL	Length of IED (4 = four octets for ports)
IED	Port numbers (octet 1,2 = destination port, octet 3,4=originator port)
UD	User Data (WV primitive)

The source and destination port have same port numbers, assigned in the CSP transport-binding document [CSP Trans].

5.3. Handling of Multiple Short Messages within Transaction

A transaction over short message bearer MAY be split to more than one short message when parameters exceed 160 characters. To accomplish this, there are two basic techniques available:

1. Use of concatenated short messages (SM-TP level concatenation).
2. Use concatenation identifier (***DD***) identifier in a short message text.

Alternative 1) is based on SMS technology, which is optionally supported in the SMSC and terminal product. It does not require support in text level.

Alternative 2) needs to be used when SMS concatenation is not available. The rules are:

- If the short message does not need text-level concatenation (single SMS or SM-TP level concatenation), the ***DD*** identifier is not present.
- If the short message needs text-level concatenation, the **DD** identifier is present (starting from 'a') and the first letter identifies the current SMS within concatenation sequence and the second letter identifies the last SMS within the sequence.
- While transferring the multiple short messages concatenated in a text level, the transaction identifier will be the same in all of the short messages.
- The request or response is incomplete until all short messages within the same transaction are received.

The text-level concatenation identifier does not require that the messages will be received in order. An example of concatenation could be the terminal sending a message:

```
WV12NM23ac MC="This is a very
WV12NM23cc very long textual content..."
WV12NM23bc very long message, and it has very
```

And the server responding:

```
WV12ST23 ST=200
```

A single short message MAY also contain messages from multiple transactions. In such cases, messages are separated by an ampersand (&) parameter according to the parameter syntax (e.g. space-ampersand-space). Note that the preamble MUST NOT be fragmented, thus each WV message preamble and an extra space character must fit as a whole within the SMS message.

5.4. Transport Bindings

As defined in the [CSP Trans] document the CSP SMS Binding can be used over a SMS transport. The textual SMS messages (not encoded using a User Data Header) MAY also be used over a general HTTP based transport as defined in [CSP Trans].

6. Availability of transactions

The SMS protocol itself provides some features and lacks others, which simplifies or disables certain transactions.

All instant messages that contain plain text message content and fit into the limits specified in 5.1 Syntax without truncation SHOULD be pushed to the terminal.

The following table describes the availability of each primitive defined in [CSP]:

Transaction	Support	Code
AddGroupMembersRequest	Full	AM
BlockEntityRequest	Full	BE
CancelAuthRequest	Full	CR
CancelInviteRequest	Full	CI
CancelInviteUserRequest	Full	CU
ClientCapabilityRequest	Simplified	CP
ClientCapabilityResponse	Full	PC
CreateAttributeListRequest	Full	CA
CreateGroupRequest	Full	CG
CreateListRequest	Full	CL
DeleteAttributeListRequest	Full	DA
DeleteGroupRequest	Full	DG
DeleteListRequest	Full	DL
DeliveryReportRequest	Full	DR
Disconnect	Full	DI
ExtendedRequest	Full	ER
ExtendedResponse	Full	RE
GetAttributeListRequest	Full	GA
GetAttributeListResponse	Full	AG
GetBlockedListRequest	Full	GB
GetBlockedListResponse	Full	BG
GetGroupMembersRequest	Full	GM
GetGroupMembersResponse	Full	MG
GetGroupPropsRequest	Full	GR
GetGroupPropsResponse	Full	RG
GetJoinedUsersRequest	Full	JU
GetJoinedUsersResponse	Full	UJ
GetListRequest	Full	GL
GetListResponse	Full	LG
GetMessageListRequest	Full	MR
GetMessageListResponse	Simplified	RM
GetMessageRequest	Full	GX
GetMessageResponse	Simplified	MX
GetPresenceRequest	Full	GP
GetPresenceResponse	Full	PG
GetReactiveAuthStatusRequest	Full	AS
GetReactiveAuthStatusResponse	Full	SA
GetSPInfoRequest	Full	GS
GetSPInfoResponse	Simplified	SG
GetWatcherListRequest	Full	GW
GetWatcherListResponse	Full	WG
GroupChangeNotice	Full	GG
ForwardMessageRequest	N/A	
InviteRequest	Full	IR
InviteResponse	Full	RI
InviteUserRequest	Full	IU
InviteUserResponse	Full	UI
JoinGroupRequest	Full	JG
JoinGroupResponse	Full	GJ

KeepAliveRequest	Full	KA
KeepAliveResponse	Full	AK
LeaveGroupRequest	Full	LU
LeaveGroupResponse	Full	UL
ListManageRequest	Full	LM
ListManageResponse	Full	ML
LoginRequest	Full	LR
LoginResponse	Simplified	RL
LogoutRequest	Full	OR
MemberAccessRequest	Full	ME
MessageDelivered	Full	MD
MessageNotification	N/A	
NewMessage	Simplified	NM
PollingRequest	Full	PO
PresenceAuthRequest	Full	PR
PresenceAuthUser	Full	RP
PresenceNotificationRequest	Full	PN
RemoveGroupMembersRequest	Full	RM
RejectListRequest	Full	RE
RejectListResponse	Full	ER
RejectMessageRequest	N/A	
SearchRequest	Full	SR
SearchResponse	Full	RS
SendMessageRequest	Simplified	SM
SendMessageResponse	Full	MS
ServiceRequest	Simplified	SQ
ServiceResponse	Simplified	QS
SetDeliveryMethodRequest	N/A	
SetGroupPropsRequest	Full	SP
Status	Full	ST
StopSearchRequest	Full	SS
SubscribeGroupNoticeRequest	Full	SU
SubscribeGroupNoticeResponse	Full	US
SubscribePresenceRequest	Full	SB
UnsubscribePresenceRequest	Full	PS
UpdatePresence	Full	UP
VersionDiscoveryRequest	Simplified	VD
VersionDiscoveryResponse	Simplified	DV

The codes are not case sensitive.

7. SMS-specific encoding

7.1. Information elements

All information elements have been encoded into shorter codes. Note that in order to be able to differentiate different types of data, some of the information elements have multiple codes based on the type of the data carried within the information element. Examples for this kind of codes are: AdminMapList, Detailed-Result, Recipient, Sender, Block/Unblock Lists, Grant/Ungrant lists, etc.

Element	Code
Acceptance	AC
Add-Users-List	AU
Add-Nick-List	AN
Administrator	AD
AdminMapList – AdminMapping	AA
AdminMapList – ModMapping	AM
AdminMapList – UserMapping	AE
Agreed-CapabilityList	AP
Attribute-Association-Contact-List	AG
Attribute-Association-User-List	AL
All-Functions	AF
All-Functions-Request	AR
Auto-Subscribe	AS
Blocked – Group-List	BG
Blocked – ScreenName-List	BS
Blocked – User-List	BL
Blocked-List-Inuse	BU
Client-ID	CI
Code	RC
CapabilityList	CA
CapabilityRequest	CR
CompletionFlag	CF
Contact-List-ID	CL
Contact-List-Props	CP
DateTime	DT
Default-Association-List	DA
Default-CList-ID	DC
Default-List	DL
Delivery-Report-Request	DE
Description	RT
Detailed-Result – Contact-List-ID	DI
Detailed-Result – Domain	DD
Detailed-Result – Group	DG
Detailed-Result – Message-ID	DM
Detailed-Result – Screenname	DS
Detailed-Result – User	DU
Delivery-Time	DX
Extension-Block	EB
Digest-Bytes	DB
Digest-Schema	DI
Granted – Group-List	GG
Granted – ScreenName-List	GS
Granted – User-List	GL
Granted-List-Inuse	GU
Group-ID	GI
Group-Props	GP
HistoryPeriod	HP
Invite-ID	II

Invite-Reason	IR
Invite-Response	IX
Invite-Type	IT
JoinGroup	JG
Joined-Request	JR
Joined-Users-List	JU
Keep-Alive-Time	KA
Left-Users-List	LU
Max-Watcher-List	MW
Message-Content	MC
Message-Count	MN
Message-ID	MI
Moderator	MO
Name	NA
Nonce	NO
Not-Available-Functions	NF
Other-Server-MSISDN	OS
Other-Server-URL	OU
Own-Props	OP
Own-Screen-Name	ON
Password-String	PW
Presence	PR
PresenceSubList	PS
ReactiveAuthStatus-List	RA
Recalled-Content	RC
Recall-Reason	RR
Receive-List	RL
Recipient – Contact-ListID	RI
Recipient – GroupID	RG
Recipient – ScreenName	RM
Recipient – UserID	RE
Remove-Nick-List	RN
Remove-Users-List	RU
Requested-Functions	RF
Result (Status code and description)	ST
Screen-Name	SN
Search-Findings	SF
Search-ID	SD
Search-Index	SX
Search-Limit	SL
Search-Pair-List	SP
Search-Results	SR
Sender – GroupID	SG
Sender – ScreenName	SM
Sender – UserID	SE
Session-Cookie	SC
Session-ID	SI
SubscribeNotification	SA
Subscribe-Type	SU
Subscription-State	SS
Supported-Digest-Schema	SH
Text	TX
Time-To-Live	TL
Unblock – Group-List	UB
Unblock – ScreenName-List	UC
Unblock – User-List	UU
Ungrant – Group-List	UG
Ungrant – ScreenName-List	UD

Ungrant – User-List	UL
Update-Value-List	UV
URL	UR
User (ordinary access)	US
User-ID	UI
User-Nick-List	UN
User-Prop-List	UP
UserMapList	UM
Validity	VA
Version-List	VL
Watcher	WA
Welcome-Text	WT

The codes are not case sensitive.

7.2. Service tree elements

To shorten messages, the elements of the service tree have been encoded into shorter codes. The tree is different as from what is specified in [CSP]. For further clarification please refer to chapter 7.11.1 Version-List on page 25. Note that the elements marked unsupported in the table below cannot be used in the SMS-specific service list.

Name	Support	Code
ADDGM	Yes	AG
AttListFunc	Yes	AF
BLENT	Yes	BL
CAAUT	Yes	CA
CAINV	Yes	CI
CALI	Yes	CL
CCLI	Yes	CC
ContListFunc	Yes	FC
CREAG	Yes	CG
DALI	Yes	DA
DCLI	Yes	DC
DELGR	Yes	DG
FundamentalFeat	Yes	FF
FWMSG	No	
GALS	Yes	GA
GCLI	Yes	GC
GETAUT	Yes	AS
GETGM	Yes	GG
GETGP	Yes	GR
GETJU	Yes	GJ
GETLM	Yes	GL
GETM	Yes	GM
GETPR	Yes	GP
GETSPI	Yes	GS
GETWL	Yes	GW
GLBLU	Yes	GB
GRCHN	Yes	GN
GroupAuthFunc	Yes	GF
GroupFeat	Yes	GE
GroupMgmtFunc	Yes	GT
GroupUseFunc	Yes	GU
IMAuthFunc	Yes	IA
IMFeat	Yes	IF
IMReceiveFunc	Yes	IR
IMSendFunc	Yes	IS
MF	Yes	MF

MG	Yes	MG
MM	Yes	MM
MP	Yes	MP
INVIT	Yes	IV
InviteFunc	Yes	IN
MBRAC	Yes	MA
MCLS	Yes	MC
MDELIV	Yes	MD
NEWM	Yes	NM
NOTIF	No	
PresenceAuthFunc	Yes	PA
PresenceDeliverFunc	Yes	PD
PresenceFeat	Yes	PF
REACT	Yes	RA
REJCM	No	
REJEC	Yes	RE
RMVGM	Yes	RG
SearchFunc	Yes	SF
ServiceFunc	Yes	SE
SETD	No	
SETGP	Yes	SG
SRCH	Yes	SR
STSRC	Yes	ST
SUBGCN	Yes	SU
UPDPR	Yes	UP
WVCSPFeat	Yes	WV

The codes are not case sensitive.

7.3. Client capabilities

In order to shorten messages, the client capabilities have been encoded into a shorter code.

Name	Code
ClientType	CT
CIRHTTPAddress	CI
DefaultLanguage	DL
MultiTrans	MT
ServerPollMin	PM
SSMS	SS
STCP	ST
SUDP	SU
SupportedCIRMethod	SC
TCPAddress	TA
TCPPort	TP
UDPPort	UP
WAPSMS	WS
WAPUDP	WU

7.4. The codes are not case sensitive. Presence attributes

Extension attributes are not supported, only presence attributes described in the [PA].

Only plain text -based presence attributes are supported.

The qualifier is not present.

To shorten messages, the presence attributes have been encoded in shorter codes.

Name	Support	Code
Accuracy (GeoLocation)	Full	AL
Accuracy (Address)	Full	AA
Address	Full	AD
AddrPref	Full	AP
Alias	Full	AI
Altitude	Full	AT
Building	Full	BU
Caddr	Full	CD
Cap	Full	CA
City	Full	CI
ClientInfo	Full	CF
ClientProducer	Full	CP
ClientType	Full	CT
ClientVersion	Full	CV
CommC	Full	CM
CommCap	Full	CC
Contact	Full	CB
ContactInfo	Simplified	CE
ContainedvCard	N/A	
ContentType	Full	CY
Country	Full	CO
Crossing1	Full	C1
Crossing2	Full	C2
Cname	Full	CN
Cpriority	Full	CR
Cstatus	Full	CS
DevManufacturer	Full	DM
DirectContent	N/A	
Extended Presence Info	N/A	
FreeTextLocation	Full	FT
GeoLocation	Full	GL
Inf_Link	Full	IK
InfoLink	Full	IL
Language	Full	LN
Latitude	Full	LA
TimeZone	Full	LT
Link	Full	LI
Longitude	Full	LO
Model	Full	MO
NamedArea	Full	NA
Note	Full	NT
OnlineStatus	Full	OS
PLMN	Full	PM
PrefC	Full	PF
PreferredContacts	Full	PC
PreferredLanguage	Full	PL
ReferredContent	Full	RC
ReferredvCard	Full	RV
Registration	Full	RG
Status	Full	SA
StatusContent	Simplified	SC
StatusMood	Full	SM
StatusText	Full	ST
Street	Full	SR

Text	Full	TE
TimeZone	Full	TZ
UserAvailability	Full	UA

The codes are not case sensitive.

7.5. Presence values

To shorten messages, the enumerated presence values have been encoded in short codes.

Name	Code
ANGRY	AG
ANXIOUS	AX
ASHAMED	AS
AUDIO_CALL	AU
AVAILABLE	AV
BORED	BO
CALL	CA
CLI	CL
CLOSED	CS
COMPUTER	CO
DISCREET	DI
EMAIL	EM
EXCITED	EX
HAPPY	HA
IM	IM
IM_OFFLINE	OF
IM_ONLINE	ON
IN_LOVE	IL
INVINCIBLE	IN
JEALOUS	JE
MMS	MS
MOBILE_PHONE	MP
NOT_AVAILABLE	NA
OPEN	OP
OTHER	OT
PDA	PD
SAD	SA
SLEEPY	SL
SMS	SM
VIDEO_CALL	VC
VIDEO_STREAM	VS

The codes are not case sensitive.

7.6. Group properties

To shorten messages, the group properties have been encoded in shorter codes.

Name	Code
Accesstype	AT
ActiveUsers	AU
AutoDelete	AD
AutoJoin	AJ
History	HT
IsMember	IM
MaxActiveUsers	MU

Name	NM
PrivateMessaging	PM
PrivilegeLevel	PL
Searchable	SE
ShowID	SI
Topic	TO
Type	TY
Validity	VL
WelcomeNote	WN
AutoJoin	AJ

The codes are not case sensitive.

7.7. Contact list properties

To shorten messages, the contact list properties have been encoded in shorter codes.

Name	Code
DisplayName	DN
Default	DE

The codes are not case sensitive.

7.8. Search elements

To shorten messages, the enumerated search elements have been encoded in shorter codes.

Name	Code
GROUP_ID	GI
GROUP_NAME	GN
GROUP_TOPIC	GT
GROUP_USER_ID_JOINED	GJ
GROUP_USER_ID_OWNER	GO
USER_ALIAS	UA
USER_EMAIL_ADDRESS	UE
USER_FIRST_NAME	UF
USER_ID	UI
USER_LAST_NAME	UL
USER_MOBILE_NUMBER	UM
USER_ONLINE_STATUS	UO
GROUP_USER_ID_OWNER	GO
GROUP_USER_ID_AUTOJOIN	UJ

The codes are not case sensitive.

7.9. Watcher state values

To shorten messages, the enumerated watcher state values have been encoded in shorter codes.

Name	Code
CURRENT_SUBSCRIBER	CS
FORMER_SUBSCRIBER	FS
PRESENCE_ACCESS	PA

The codes are not case sensitive.

7.10. Reactive authorization state values

To shorten messages, the enumerated reactive authorization state values have been encoded in shorter codes.

Name	Code
DENIED	DN
GRANTED	GN
PENDING	PN

The codes are not case sensitive.

7.11. Information element-specific encoding

There are a number of information elements where the syntax is not self-explanatory, and requires additional clarification. These information elements are collected and explained in this chapter.

7.11.1. Version-List

The version list – unlike in XML – is not expressed using namespaces, as the SMS protocol itself does not use namespaces. Version numbers are expressed as it is defined in section 4.1 – Syntax on page 4 (see "aa"), thus the list of different version numbers is expressed as a list of comma-separated numbers. For example the following list is valid for a client/server that supports 1.0, 1.1, and 1.2 versions of WV specifications:

```
VL=(10,11,12)
```

The version numbers MAY be in any order.

7.11.2. Service tree

The service tree – unlike in XML – is not transferred as a tree. It is simply a list of short codes, which indicate the requested or not agreed services. The list MUST NOT contain one specific element more than once. If a child element is specified then the parent is not present in the list. The short codes MAY be in any order.

7.11.2.1. Explanation using an example

The following step-by-step build-up demonstrates the following case: the client requests all available features and functions, but the server does not agree to provide FundamentalFeat (FF), ContListFunc (FC), PresenceAuthFunc (PA), and IMAuthFunc (IA). Note that the examples below are indicating the service list only, all other parameters are missing.

The client requests every features and functions so it sends the root element only:

```
RF=WV
```

The server sends the negated list to the client. It indicates only those elements that the client did request, but the server does not agree to provide. In our example FF, FC, PA, IA SHALL be indicated. Let us build up this list (top-down).

WV is the root, that's surely needed, and we put the parentheses there, since there will be more than one element.

```
NF=(WV)
```

Let us go one level down. The server does not provide the whole FF sub-tree. FF is a sub-tree; it is a child of WV, so the parent (WV) is removed and FF is added to the list.

```
NF=(FF)
```

The server does not provide FC and PA. These are child element of the PF, so PF needs to be added to the list as well.

```
NF=(FF,PF)
```

The server does not provide IA. These are child elements of the IF, so IF needs to be added to the list as well.

```
NF=(FF,PF,IF)
```

Let us go one level down. FC, and PA SHALL be indicated in PF, so we add these two and remove the parent PF.

```
NF=(FF,FC,PA,IF)
```

Finally, IA SHALL be indicated in IF, so we add IA, and remove the parent IF.

```
NF=(FF,FC,PA,IA)
```

The servers sends this tree in the response:

```
NF=(FF,FC,PA,IA)
```

The list MAY be constructed many different ways, depends on the actual implementation.

7.11.3. PresenceSubList parameter

Unlike in XML, SMS does not support any extended presence information.

The PresenceSubList element is very similar to the service tree. Each presence information element and enumerated value has its own short code, which must be used at all times.

For clarification the basic syntax is:

Attributes that do not have child attributes:
<attribute>[,<qualifier$_m$>][,<value$_n$>]
Attributes that do have child attributes:
<attribute>[,<qualifier$_o$>,<sub-attribute $_o$>]

<sub-attribute> has the same structure as <attribute>, <qualifier> and <value> carries the actual qualifier and presence value. The qualifier is always present in attributes that do have qualifier; the value MAY be missing (if value is meant to be empty). If there is more than one presence element present in under the same parent, then these must be in parentheses.

The order of the <attribute> element within the PresenceSubList does not matter, but every single <attribute> element MAY be present only once.

7.11.3.1. Explanation using an example

The following step-by-step build-up demonstrates how to build up an empty Presence SubList parameter. (Empty PresenceSubList is used for referencing: attribute lists, authorization, etc.) It will contain all presence attributes that are available for referencing (except ContainedvCard, DirectContent, extended presence information: these are not supported presence attributes).

Let us begin with the easy ones that do not have child attributes: OnlineStatus (OS), Registration (RG), FreeTextLocation (FT), PLMN (PM), UserAvailability (UA), PreferredLanguage (PL), StatusText (ST), StatusMood (SM), Alias (AI), TimeZone (TZ):

```
PS=(OS,RG,FT,UA,PL,ST,SM,AI)
```

Let us add now the ones that do have child attributes (let us not add child attributes yet): ClientInfo (CF), GeoLocation (GL), Address (AD), CommCap (CC), PreferredContacts (PC), StatusContent (SC), ContactInfo (CE):

```
PS=(OS,RG,FT,UA,PL,ST,SM,AI,TZ,CF,GL,AD,CC,PC,SC,CE,IL)
```

If used in the proper primitive, the above PresenceSubList element would authorize all presence information to the specified users. Note that some of these attributes do have child attribute(s), but these child attributes are not usable in empty attribute lists, as those cannot be authorized separately.

Now let us see how the PresenceSubList is filled up with qualifiers and values (and the possible child attributes). The following example demonstrates how to build up a PresenceSubList parameter that carries all supported presence attributes. In order to demonstrate empty values; attributes without qualifier and empty values - the TimeZone (TZ) and the DevManufacturer attributes will have empty values. In order to demonstrate how to include more than one child attribute (where allowed of course), two CommC (CM) and two AddrPref (AP) attributes will be included.

Let us create the whole structure first without any qualifiers, or values (note that this example will be invalid, because a reference (empty) list MAY not contain child attributes). The child attributes to be added to the empty PresenceSubList above are:

ClientInfo (CF)	ClientType (CT), DevManufacturer (DM), ClientProducer (CP), Model (MO), ClientVersion (CV), Language (LN)
GeoLocation (GL)	Longitude (LO), Latitude (LA), Altitude (AT), Accuracy (AL)
Address (AD)	Country (CO), City (CI), Street (SR), Crossing1 (C1), Crossing2 (C2), Building (BU), NamedArea (NA), Accuracy (AA)
CommCap (CC)	CommC (CM)
PreferredContacts (PC)	AddrPref (AP)
StatusContent (SC)	ReferredContent (RC), ContentType (CY) - DirectContent is not supported.
ContactInfo (CE)	ReferredvCard (RV) - ContainedvCard is not supported.
InfoLink (IL)	Inf_Link (IK)

```
PS=(OS,RG,FT,UA,PL,ST,SM,AI,TZ,(CF,(CT,DM,CP,MO,CV,LN)),(GL,(LO,LA,AT,AL)),
(AD,(CO,CI,SR,C1,C2,BU,NA,AA)),(CC,CM),(PC,AP),(SC,RC),(CE,RV),(IL,IK))
```

Let us create the lowest level of the structure now (note that this example will be invalid as well). The child attributes to be added to the empty PresenceSubList above are:

CommC (CM)	Cap (CA), Status (SA), Contact (CB), Note (NT)
AddrPref (AP)	PrefC (PF), Caddr (CD), Cstatus (CS), Cname (CN), Cpriority (CR)
Inf_Link (IK)	Link (LI), Text (TE), ContentType (CY)

```
PS=(OS,RG,FT,UA,PL,ST,SM,AI,TZ,(CF,(CT,DM,CP,MO,CV,LN)),(GL,(LO,LA,AT,AL)),
(AD,(CO,CI,SR,C1,C2,BU,NA,AA)),(CC,(CM,((CA,SA,CB,NT),(CA,SA,CB,NT)))),(PC,
(AP,((PF,CD,CS,CN,CR),(PF,CD,CS,CN,CR)))),(SC,(RC,CY)),(CE,RV),(IL,(IK,(LI,
TE,CY))))
```

The structure is complete, let us fill it up now with qualifiers (those that have), and values (those that have).

```
PS=((OS,T,T),(RG,T,T),(FT,T,"At home"),(UA,T,AV),(PL,T,fin),(ST,T,"Busy
editing a
document"),(SM,T,SL),(AI,T,ASa),(TZ,T,+02),(CF,T,((CT,MP),(DM,"ABC
company"),(CP,"DEF
Company"),(MO,xyz200),(CV,1.1b),(LN,fin))),(GL,T,((LO,"35 24
15.652W"),(LA,"12 36
22.5N"),(AT,250),(AL,50))),(AD,T,((CO,FR),(CI,Paris),(SR,"Big
street"),(C1,"A street"),(C2,"B street"),(BU,"Eiffel tower"),(NA,"Eiffel
tower"),(AA,300))),(CC,T,(CM,(((CA,CA),(SA,CS),(CB,+35899123123),(NT," I am
using this phone outside office
hours")),((CA,IM),(SA,OP),(CB,wv:user@im.com),(NT,"My IM-application is now
online"))))),(PC,T,(AP,(((PF,CA),(CD,+35899123123),(CS,OP),(CN,"Home
Phone"),(CR,10)),((PF,SM),(CD,+35899123123),(CS,CS),(CN,"Home
SMS"),(CR,20))))),(SC,T,((RC,http://www.foo.com/MMS/Pictures/MyLogo),(CY,im
age/gif))),(CE,T,(RV,http://www.foo.com/Contacts/vCards/MyCard)),(IL,T,(IK,
((LI,http://www.myserviceprovider.com/myHomePage),(TE,"This is my
homepage"),(CY,text/html)))))
```

The PresenceSubList above is complete. Note that it includes all available attributes, and there are two attributes, which are included twice. The size of the above parameter is about 2000 characters shorter than the equivalent XML representation.

7.11.4. AdminMapList, UserMapList parameters

These information elements are defined as structures in XML, and cannot be included in SMS without specific syntax. In order to be able to distinguish between recipient Admin, Mod, and User mappings within the AdminMapList element are separated to AA, AM, AS.

The syntax is:

```
AA=<MappingList>
AM=<MappingList>
AE=<MappingList>
UM=<MappingList>
```

Where <MappingList> is comma-separated <Mapping> elements: <Mapping$_1$>,<Mapping$_n$>

One single <Mapping> element consists of a Screen-Name (note that it is only the Name part without Group-ID), and an optional User-ID: <SName>[,<UserID>]. If the <Mapping> element contains the optional User-ID, it is wrapped with parentheses: (<SName>,<UserID>) .

Examples:

```
AA=He
AM=((He,wv:he@there.com))
AE=(He,She)
UM=((He,wv:he@there.com),(She,wv:she@there.com))
```

Note that the parentheses must be doubled when there is only one screen name with user-id. Please refer to chapter 7.12.5 Single nickname on page 30, as the problem described there is identical.

7.11.5. ExtBlock parameter

The syntax for the ExtBlock parameter is as follows:

```
EB=(<namespace indication>,([param values]))
```

The contents and syntax of “Param values” is outside the scope of Wireless-Village.

The following is a valid example:

```
EB=(foo.com/1.2,(value1,value2))
```

7.11.6. ReactiveAuthStatus-List parameter

The basic syntax for the ReactiveAuthStatus-List parameter is as follows:

```
RA=(<owner>,<reactive auth state>[,(<attribute 1>,<attribute n>)])
```

where

<owner> is the user-ID to whom the authorization status belongs.
<reactive auth state> state code of the authorization.
<attribute> is the code of the presence attribute.

The following is a valid example:

```
RA=((wv:john@smith.com,GN,(OS,FT)),(wv:matthias@smith.com,PN))
```

7.12. Single parameters

7.12.1. Single presence attribute with qualifier or value

Case: the PresenceSubList parameter is a single presence attribute with qualifier and/or value.

A single presence attribute consists an attribute, a qualifier, and/or a value:

Parm=<attribute>[,<qualifier>,<sub-attribute>]

Parm=<attribute>[,<qualifier>][,<value>]

The problem is that this looks like a list for the parser, so it is not possible to include a single presence attribute with qualifier and/or value in any parameter like this:

```
PS=OS,T,T
```

Nor like this (this would be recognized as a reference (empty) attribute list):

```
PS=(OS,T,T)
```

In order to be able to include a single presence attribute with qualifier and/or value in a parameter the parentheses must be doubled:

```
PS=((OS,T,T))
```

Note that this problem does not occur when there is a list of presence attributes (with qualifier and/or value) assigned to a parameter, since then the double parentheses are already there:

```
PS=((OS,T,T),(RG,T,T),(FT,T,"At home"))
```

7.12.2. Single user-ID with client-ID

Case: any user-ID parameter is a single user-ID with client-ID.

A single user-ID with client-ID consists of a user-ID, and a client-ID:

```
Parm=<User-ID>,<Client-ID>
```

The problem is that this looks like a list for the parser, so it is not possible to include a single user-ID with client-ID in any parameter like this:

```
RE=wv:john@smith.com,+1234567890
```

Nor like this (this would be recognized as two user-IDs):

```
RE=(wv:john@smith.com,+1234567890)
```

In order to be able to include a single user-ID with client-ID in a parameter the parentheses must be doubled:

```
RE=((wv:john@smith.com,+1234567890))
```

Note that this problem does not occur when there is a list of user-IDs (with client-IDs) assigned to a parameter, since then the double parentheses are already there:

```
RE=((wv:john@smith.com,+1234567890),(wv:me@home.com,+1122334455))
```

7.12.3. Single screen name

Case: any screen name parameter is a single screen name.

A single screen name consists of a name, and a group-ID part:

```
Parm=<SName>,<Group-ID>
```

The problem is that this looks like a list for the parser, so it is not possible to include a single screen name in any parameter like this:

```
SN=matthias,wv:/chatroup@wv.com
```

Nor like this:

```
SN=(matthias,wv:/chatroup@wv.com)
```

In order to be able to include a single screen name in a parameter the parentheses must be doubled:

```
SN=((matthias,wv:/chatroup@wv.com))
```

Note that this problem does not occur when there is a list of screen names assigned to a parameter, since then the double parentheses are already there:

```
SN=(("The
boss",wv:/othergroup@somewhere.com),(matthias,wv:/chatroup@wv.com))
```

7.12.4. Single search-pair

Case: a search-pair parameter is a single search-pair.

A single search-pair consists of a type, and a substring part:

```
Parm=<Type>,<Substring>
```

The problem is that this looks like a list for the parser, so it is not possible to include a single search-pair in a search-pair parameter like this:

```
SP=UL,Smith
```

Nor like this:

```
SP=(UL,Smith)
```

In order to be able to include a single search-pair in a parameter the parentheses must be doubled:

```
SP=((UL,Smith))
```

Note that this problem does not occur when there is a list of search-pairs assigned to a parameter, since then the double parentheses are already there:

```
SP=((UL,Smith),(UO,T))
```

7.12.5. Single nickname

Case: any nickname parameter is a single nickname.

A single nickname consists of a name, and a user-ID part:

```
Parm=<Name>,<User-ID>
```

The problem is that this looks like a list for the parser, so it is not possible to include a single nickname in any parameter like this:

```
AN="Randall the Vandal",wv:randall@fairlane.com
```

Nor like this:

```
AN=("Randall the Vandal",wv:randall@fairlane.com)
```

In order to be able to include a single nickname in a parameter the parentheses must be doubled:

```
AN=(("Randall the Vandal",wv:randall@fairlane.com))
```

Note that this problem does not occur when there is a list of nicknames assigned to a parameter, since then the double parentheses are already there:

```
AN=(("Randall the
Vandal",wv:randall@fairlane.com),wv:no.nick@name.com,(Brainstrom,wv:bright@
dark.com))
```

7.12.6. Single contact-list property

Case: a contact-list property parameter is a single property.

A single contact-list property consists of a property name, and a property value part:

```
Parm=<Property>,<Value>
```

The problem is that this looks like a list for the parser, so it is not possible to include a single contact-list property in a contact-list property parameter like this:

```
CP=DN,"My enemies"
```

Nor like this:

```
CP=(DN,"My enemies")
```

In order to be able to include a single contact-list property in a parameter the parentheses must be doubled:

```
CP=((DN,"My enemies"))
```

Note that this problem does not occur when there is a list of contact-list properties assigned to a parameter, since then the double parentheses are already there:

```
CP=((DN,"My enemies"),(DE,T))
```

7.12.7. Single watcher

Case: any watcher parameter includes only a single watcher.

A single watcher (client-ID is optional) consists of a user-ID, a client-ID, and the watcher status:

```
Parm=<User-ID>,<WatcherStatus>
Parm=(<User-ID>,<Client-ID>),<WatcherStatus>
```

The problem is that this looks like a list for the parser, so it is not possible to include a single watcher in any parameter like this:

```
WA=wv:he@there.com,FS
WA=(wv:she@there.com,http://123.123.123.123:80/IMPSAPP),FS
```

Nor like this (this would be recognized as two separate watchers):

```
WA=(wv:he@there.com,FS)
WA=((wv:she@there.com,http://123.123.123.123:80/IMPSAPP),FS)
```

In order to be able to include a single watcher (with or without client-ID) in a parameter the parentheses must be doubled:

```
WA=((wv:he@there.com,FS))
WA=(((wv:she@there.com,http://123.123.123.123:80/IMPSAPP),FS))
```

Note that this problem does not occur when there is a list of user-IDs (with client-IDs) assigned to a parameter, since then the double parentheses are already there:

```
WA=((wv:he@there.com,CS),((wv:she@there.com,http://123.123.123.123:80/IMPSA
PP),FS)),((wv:me@there.com,http://123.123.123.123:80/IMPSAPP),PA))
```

8. SMS Binding Examples

8.1. Status primitive

With a single status code:

ST=<StatusCode>

```
WV12ST761 SI=im.user.com#48815@server.com ST=200
```

With a status code and description:

ST=(<StatusCode>,<Description>)

```
WV12ST761 SI=im.user.com#48815@server.com ST=(200,”Successfully
completed.”)
```

With multiple status codes and descriptions:

ST=(<StatusCode>,<Description>)

```
DI=(<StatusCode>,<Description>,<Contact-List ID n>)
DD=(<StatusCode>,<Description>,<Domain n>)
DG=(<StatusCode>,<Description>,<GroupID n>)
DM=(<StatusCode>,<Description>,<Message-ID n>)
DS=(<StatusCode>,<Description>,(<ScreenName n>,GroupID n))
DU=(<StatusCode>,<Description>,<UserID n>)


WV12ST761 SI=im.user.com#48815@server.com ST=(201,”Partially completed.”)
DU=((531,”Unknown user.”,wv:bad_user1@im.com,wv:bad_user2@im.com)
,(532,”Blocked.”,wv:bad_user3@im.com,wv:bad_user4@im.com))
```

8.2. PollingRequest primitive

```
WV12PO761 SI=im.user.com#48815@server.com
```

8.3. Version discovery transaction

8.3.1. VersionDiscoveryRequest primitive

```
WV12VD761
```

8.3.2. VersionDiscoveryResponse primitive

```
WV12DV761 VL=(11,12) OS=(+123456789,+987654321)
```

8.4. 2-Way Login transaction

8.4.1. LoginRequest primitive

```
WV12LR761 UI=wv:john@smith.com CI=+1234567890 PW=this1is2my3pass
SC=im.user.com#20011224#328746293 TL=600
```

8.4.2. LoginResponse primitive

```
WV12RL761 CI=+1234567890 ST=(200,”Successfully completed.”)
SI=im.user.com#48815@server.com
KA=300 CR=T
```

8.5. 4-way Login transaction

The Capability-Request element is omitted from the LoginResponse primitive, as there is no such transaction in SMS bindings.

8.5.1. LoginRequest primitive

```
WV12LR761 UI=wv:john@smith.com CI=+1234567890 SH=(PWD,SHA,MD4,MD5,MD6)
SC=im.user.com#20011224#328746293
```

8.5.2. LoginResponse primitive

```
WV12RL761 CI=+1234567890 ST=(401,”Further authorization required”)
NO=92387rhf934fho3fh9fkn309fn3pfun304ufn3 DH=MD6 CR=F
```

8.5.3. LoginRequest primitive

```
WV12LR762 UI=wv:john@smith.com CI=+1234567890
DB=alkkuayfdsAKDSJfsdfjhksadhlkasdlkfgsal TL=600
SC=im.user.com#20011224#328746293
```

8.5.4. LoginResponse primitive

```
WV12RL762 CI=+1234567890 ST=(200,”Successfully logged in.”)
SI=im.user.com#48815@server.com KA=300 CR=T
```

8.6. Client capability negotiation transaction

Note that the transaction is simplified:

- The request does not contain all of the values defined in [CSP] - as some of those are not applicable.
- If the underlying transport is not SMS, the response does not contain the AgreedCapabilities element - as none of the values are applicable to SMS transport.

8.6.1. ClientCapabilityRequest primitive

```
WV12CP761 SI=im.user.com#48815@server.com CA=((CT,MP),(DL,fin),(MT,5))
```

8.6.2. ClientCapabilityResponse primitive

```
WV12PC761 SI=im.user.com#48815@server.com
```

8.7. Logout transaction

8.7.1. LogoutRequest primitive

```
WV12OR761 SI=im.user.com#48815@server.com
```

8.7.2. Status primitive

```
WV12DI761 SI=im.user.com#48815@server.com ST=200
```

8.8. Server initiated logout transaction

8.8.1. Disconnect primitive

```
WV12DI761 SI=im.user.com#48815@server.com ST=(601,”Updating server
software. All services offline for 3 hours.”)
```

8.9. Keep Alive transaction

8.9.1. KeepAliveRequest primitive

WV12KA761 SI=im.user.com#48815@server.com TL=600

8.9.2. KeepAliveResponse primitive

```
WV12AK761 SI=im.user.com#48815@server.com ST=(200,”Successfully
completed.”)KA=600
```

8.10. Get service provider info transaction

8.10.1. GetSPInfoRequest primitive

```
WV12GS761 SI=im.user.com#48815@server.com
```

8.10.2. GetSPInfoResponse primitive

MMS messages are not allowed, thus the Logo element is not present in the primitive.

```
WV12SG761 SI=im.user.com#48815@server.com NA=”Wireless Village” TX=”This is
OMA’s IMPS test service” UR=”http://www.openmobilealliance.org”
```

8.11. Service Negotiation transaction

The following example illustrates service negotiation with the following parameters:

Client requests:

- All Fundamental Features,
- All IM Features,
- All Presence Features,
- None of the Group Features.

Server does not agree to provide:

- Fundamental features,
- GetWatcherList transaction,
- IM Authorization Functions.

8.11.1. ServiceRequest primitive

```
WV12SQ761 SI=im.user.com#48815@server.com RF=(FF,IF,PF) AR=F
```

8.11.2. ServiceResponse primitive

```
WV12QS761 SI=im.user.com#48815@server.com  NF=(FF,GW,IA)
```

8.12. Search transaction

8.12.1. [CSP]SearchRequest primitive (1st)

```
WV12SR761 SI=im.user.com#48815@server.com SP=((UL,Smith),(UT,T)) SL=5
```

8.12.2. SearchResponse primitive (1st)

```
WV12RS761 SI=im.user.com#48815@server.com SF=7 CF=F SX=6 SD=112233
SR=(wv:john,wv:smithy@village.com,wv:smith@wezzon.com,wv:locksmith@site.org
,wv:smithereens@car.org)
```

8.12.3. SearchRequest primitive (continued)

```
WV12SR762 SI=im.user.com#48815@server.com SD=112233 SX=6
```

8.12.4. SearchResponse primitive (continued)

```
WV12RS762 SI=im.user.com#48815@server.com SF=7 CF=T SX=7
SR=(wv:tinsmith@home.se,wv:coppersmith@bigfish.com)
```

8.13. Stop search transactions

8.13.1. StopSearchRequest primitive

```
WV12SS763 SI=im.user.com#48815@server.com SD=112233
```

8.13.2. Status primitive

```
WV12ST763 SI=im.user.com#48815@server.com ST=200
```

8.14. Invitation transactions

8.14.1. InviteRequest primitive

```
WV12IR761 SI=im.user.com#48815@server.com II=11 IT=PR PS=(OS,TZ,FT)
RE=(wv:lara.naval@secret.gov,wv:francisco) IR=”Feel free to use my presence
infos!”
```

8.14.2. Status primitive

```
WV12ST761 ST=200
```

8.14.3. InviteUserRequest primitive

```
WV12IU762 SI=54321 II=11 IT=PR SE=wv:john@smith.com PS=(OS,TZ,FT) IR=”Feel
free to use my presence infos!”
```

8.14.4. Status primitive

```
WV12ST762 SI=54321 ST=200
```

8.14.5. InviteUserResponse primitive

```
WV12UI763 SI=54321 II=11 AC=T IX=”Thanks a lot!”
SN=((john,wv:%2Fchatroup@wv.com))
```

8.14.6. Status primitive

```
WV12ST763 SI=54321 ST=200
```

8.14.7. InviteResponse primitive

```
WV12RI764 SI=im.user.com#48815@server.com II=11 SE=wv:francisco AC=T
IX=”Thanks a lot!”
```

8.14.8. Status primitive

```
WV12ST764 ST=200
```

8.15. Canceling invitation transactions

8.15.1. CancelInviteRequest primitive

```
WV12CI761 SI=im.user.com#48815@server.com II=11
RE=(wv:lara.naval@secret.gov,wv:francisco) RR=”I will be on vacation for 1
week.”
```

8.15.2. Status primitive

```
WV12ST761 SI=im.user.com#48815@server.com ST=200
```

8.15.3. CancelInviteUserRequest primitive

```
WV12CU762 SI=54321 II=11 SE=wv:john@smith.com RR=”I will be on vacation for
a week.”
```

8.15.4. Status primitive

```
WV12ST762 SI=54321 ST=200
```

8.16. Get list of contact list IDs transaction

8.16.1. GetListRequest primitive

```
WV12GL761 SI=im.user.com#48815@server.com
```

8.16.2. GetListResponse primitive

```
WV12LG761 SI=im.user.com#48815@server.com
CL=(wv:john/colleagues,wv:john/friends) DL=wv:john/family
```

8.17. Create contact list transaction

8.17.1. CreateListRequest primitive

```
WV12CL761 SI=im.user.com#48815@server.com CL=wv:john/friends UN=((”New
friend”,wv:new@friend.org),wv:no.nick@name.com) CP=((DN,”My
friends”),(DE,T))
```

8.17.2. Status primitive

```
WV12ST761 SI=im.user.com#48815@server.com ST=(200,”Successfully
completed.”)
```

8.18. Delete contact list transaction

8.18.1. DeleteListRequest primitive

```
WV12DL761 SI=im.user.com#48815@server.com CL=wv:john/friends
```

8.18.2. Status primitive

```
WV12ST761 SI=im.user.com#48815@server.com ST=(200,”Successfully
completed.”)
```

8.19. Retrieve a contact list transaction

8.19.1. ListManageRequest primitive

```
WV12LM761 SI=im.user.com#48815@server.com CL=wv:john/friends RL=T
```

8.19.2. ListManageResponse primitive

```
WV12ML761 SI=im.user.com#48815@server.com ST=200 CP=((DN,”My
friends”),(DE,T)) UN=((”New friend”,wv:new@friend.org),wv:no.nick@name.com)
```

8.20. Add users to a contact list transaction

8.20.1. ListManageRequest primitive

```
WV12LM761 SI=im.user.com#48815@server.com CL=wv:john/friends AN=((“Randall
the
Vandal”,wv:randall@fairlane.com),wv:no.nick@name.com,(Brainstrom,wv:bright@
dark.com))
```

8.20.2. ListManageResponse primitive

```
WV12ML761 SI=im.user.com#48815@server.com ST=200 UN=((“Randall the
Vandal”,wv:randall@fairlane.com),wv:no.nick@name.com,(Brainstrom,wv:bright@
dark.com),(”New friend”,wv:new@friend.org))
```

8.21. Remove users from a contact list

8.21.1. ListManageRequest primitive

```
WV12LM761 SI=im.user.com#48815@server.com CL=wv:john/friends RN=((”New
friend”,wv:new@friend.org))
```

8.21.2. ListManageResponse primitive

```
WV12ML761 SI=im.user.com#48815@server.com ST=200 UN=((“Randall the
Vandal”,wv:randall@fairlane.com),wv:no.nick@name.com,(Brainstrom,wv:bright@
dark.com))
```

8.22. Modify properties of contact list transaction

8.22.1. ListManageRequest primitive

```
WV12LM761 SI=im.user.com#48815@server.com CL=wv:john/friends CP=((DN,”My
enemies”),(DE,T))
```

8.22.2. ListManageResponse primitive

```
WV12ML761 SI=im.user.com#48815@server.com ST=200 CP=((DN,”My
enemies”),(DE,T))
```

8.23. Create attribute list transaction

8.23.1. CreateAttributeListRequest primitive

```
WV12CA761 SI=im.user.com#48815@server.com PS=(OS,TZ,FT)
UI=(wv:matthias@salamander.com,wv:francisco) DL=T
```

8.23.2. Status primitive

```
WV12ST761 SI=im.user.com#48815@server.com ST=(200,”Successfully
completed.”)
```

8.24. Delete attribute list transaction

8.24.1. DeleteAttributeListRequest primitive

```
WV12DA761 SI=im.user.com#48815@server.com CL=wv:john/friends DL=F
```

8.24.2. Status primitive

```
WV12ST761 SI=im.user.com#48815@server.com ST=200
```

8.25. Get attribute list(s) transaction

8.25.1. GetAttributeListRequest primitive

```
WV12GA761 SI=im.user.com#48815@server.com DL=T
```

8.25.2. GetAttributeListResponse primitive

In order to be able to distinguish between User-IDs, Contact-List-IDs and the default attribute list, the parameters are separated into AG, AL, DA.

For clarification the basic syntax of Attribute-Association-List is:

(<owner(s)>,<PresenceSubList(s)>) ,where

<owner(s)> is the contact-list-ID(s) or the user-ID(s) to whom the attributes are assigned.

<PresenceSubList(s)> is the related presence structure. See chapter 7.11.3 PresenceSubList on page 26.

The <owner(s)> is omitted from the DA parameter (default attribute list).

<attribute(s)> is the name of the presence attribute(s) that are associated.

```
WV12AG761 SI=im.user.com#48815@server.com ST=200
AG=((wv:john/colleagues,OS),(wv:john/family,(OS,FT)))
AL=(((wv:matthias@salamander.com,wv:francisco@don.com),(OS,FT)),(wv:mary@si
te.com,FT)) DA=OS
```

8.26. Subscribe/unsubscribe presence transaction

8.26.1. SubscribePresenceRequest primitive

```
WV12SB761 SI=im.user.com#48815@server.com
UI=(wv:matthias@salamander.com,wv:francisco) CL=wv:john/family PS=OS AS=T
```

8.26.2. Status primitive

```
WV12ST761 SI=im.user.com#48815@server.com ST=(200,"Successfully
completed.")
```

8.26.3. PresenceNotificationRequest primitive

For clarification the basic syntax is:

(<owner(s)>,<PresenceSubList(s)>)

,where

<owner(s)> is the user-ID(s) to whom the presence attributes and values belong.

<PresenceSubList(s)> is the related presence structure. See chapter 7.11.3 PresenceSubList on page 26.

<attribute> is the name of the presence attribute to which the value belongs.

```
WV12PN761 SI=im.user.com#48815@server.com
PR=((wv:matthias@salamander.com,((OS,T,T),(FT,T,”In the
office”))),(wv:francisco,((OS,T,T))))
```

8.26.4. Status primitive

```
WV12ST761 SI=im.user.com#48815@server.com ST=200
```

8.26.5. UnsubscribePresenceRequest primitive

```
WV12PS761 SI=im.user.com#48815@server.com
UI=(wv:matthias@salamander.com,wv:francisco)
```

8.26.6. Status primitive

```
WV12ST761 SI=im.user.com#48815@server.com ST=(200,”Successfully
completed.”)
```

8.27. Get watcher list transaction

8.27.1. GetWatcherListRequest primitive

```
WV12GW761 SI=im.user.com#48815@server.com HP=0 MW=200
```

8.27.2. GetWatcherListResponse primitive

```
WV12WG761 SI=im.user.com#48815@server.com HP=172800
WA=((wv:he@there.com,CS),((wv:she@there.com,http://123.123.123.123:80/IMPSA
PP),FS)),((wv:me@there.com,http://123.123.123.123:80/IMPSAPP),PA))
```

8.28. Get presence transaction

8.28.1. GetPresenceRequest primitive

```
WV12GP761 SI=im.user.com#48815@server.com
UI=(wv:matthias,wv:francisco@don.com) PS=OS
```

8.28.2. GetPresenceResponse primitive

For clarification the basic syntax is:

(<owner(s)>,<PresenceSubList(s)>) ,where

<owner(s)> is the user-ID(s) to whom the presence attributes and values belong.

<PresenceSubList(s)> is the related presence structure. See chapter 7.11.3 PresenceSubList on page 26.

```
WV12PG761 SI=im.user.com#48815@server.com ST=200
PR=((wv:matthias@salamander.com,((OS,T,T))),(wv:francisco,((OS,T,T))))
```

8.29. Reactive presence authorization transactions

8.29.1. PresenceAuthRequest primitive

```
WV12PR761 SI=im.user.com#48815@server.com UI=wv:matthias@salamander.com
PS=(OS,TZ,FT)
```

8.29.2. PresenceAuthResponse primitive

```
WV12RP761 SI=im.user.com#48815@server.com UI=wv:matthias@salamander.com
AC=T PS=(OS,TZ,FT)
```

8.29.3. CancelAuthRequest primitive

```
WV12CR762 SI=54321 UI=wv:matthias@salamander.com
```

8.29.4. Status primitive

```
WV12ST762 SI=im.user.com#48815@server.com ST=200
```

8.30. Update presence transaction

8.30.1. UpdatePresenceRequest primitive

```
WV12UP761 SI=im.user.com#48815@server.com UV=((OS,T,T),(FT,T,”In the
office”))
```

8.30.2. Status primitive

```
WV12ST761 SI=im.user.com#48815@server.com ST=200
```

8.31. 7.28 Get reactive authorization status transaction

8.31.1. GetReactiveAuthStatusRequest primitive

```
WV12AS761 SI=im.user.com#48815@server.com
UI=(wv:matthias@salamander.com,wv:francisco)
```

8.31.2. GetReactiveAuthStatusResponse primitive

```
WV12SA761 SI=im.user.com#48815@server.com
RA=((wv:matthias@salamander.com,PN),(wv:francisco,GN,(OS,FT)))
```

8.32. Send message transaction

Only plain text messages are allowed, thus the MIME-type encoding, size and URI are not carried in the primitive.

In order to be able to distinguish between recipient User-IDs, Contact-List-IDs, Group-IDs and screen names, the parameters are separated to RE, RI, RG, RM.

8.32.1. SendMessageRequest primitive

```
WV12SM761 SI=im.user.com#48815@server.com SE=wv:me@home.com DE=T
RE=(wv:matthias@salamander.com,wv:francisco) RI=wv:john/colleagues
RG=wv:/chatroup@wv.com RM=((“The boss”,wv:/chatroup@wv.com)) MC=”Hello
everybody! How You guys doing?”
```

8.32.2. SendMessageResponse primitive

```
WV12MS761 SI=im.user.com#48815@server.com ST=(200,”Successfully
completed.”) MI=11235
```

8.33. Pushing a message from the server transaction

8.33.1. NewMessage primitive

Only plain text messages are allowed, thus the MIME-type encoding, size and URI are not carried in the primitive.

In order to be able to distinguish between sender User-IDs, Group-IDs and screen names, the parameters are separated to SE, SG, SM.

```
WV12NM761 SI=im.user.com#48815@server.com MI=11235 SE=wv:john@smith.com
DT=20011118T1203Z MC=”Hello everybody! How You guys doing?”
```

8.33.2. MessageDelivered primitive

```
WV12MD761 SI=im.user.com#48815@server.com MI=11235
```

8.34. Get message list transaction

8.34.1. GetMessageListRequest primitive

```
WV12MR761 SI=im.user.com#48815@server.com GI=wv:/chatroup@wv.com MN=5
```

8.34.2. GetMessageListResponse primitive

Only message-IDs are carried in the primitive.

```
WV12RM761 SI=im.user.com#48815@server.com MI=(1212,1123,897,624,372)
```

8.35. Retrieving a message from the server transaction

8.35.1. GetMessageRequest primitive

```
WV12GM761 SI=im.user.com#48815@server.com MI=1212
```

8.35.2. GetMessageResponse primitive

Only plain text messages are allowed, thus the MIME-type encoding, size and URI are not carried in the primitive.

In order to be able to distinguish between sender User-IDs, Group-IDs and screen names, the parameters are separated to SE, SG, SM.

```
WV12MG761 SI=im.user.com#48815@server.com MI=1212
SM=((Some1,wv:/chatroup@wv.com)) DT=20011118T1203Z MC="Hello everybody! How
You guys doing'?"
```

8.35.3. MessageDelivered primitive

```
WV12NM761 SI=im.user.com#48815@server.com MI=11235
```

8.35.4. Status primitive

```
WV12ST761 SI=im.user.com#48815@server.com ST=200
```

8.36. Delivery status report transaction

8.36.1. DeliveryReportRequest primitive

In order to be able to distinguish between recipient User-IDs, Contact-List-IDs, Group-IDs and screen names, the parameters are separated to RE, RC, RG, RM.

In order to be able to distinguish between sender User-IDs, Group-IDs and screen names, the parameters are separated to SE, SG, SM.

```
WV12DR761 SI=im.user.com#48815@server.com ST=200 DX=20011118T1204Z
SE=wv:me@home.com RE=wv:matthias@salamander.com DT=20011118T1203Z MI=11235
```

8.36.2. Status primitive

```
WV12ST761 SI=im.user.com#48815@server.com ST=200
```

8.37. Get blocked user list transaction

8.37.1. GetBlockedListRequest primitive

```
WV12GB761 SI=im.user.com#48815@server.com
```

8.37.2. GetBlockedListResponse primitive

In order to be able to distinguish between User-IDs, Group-IDs and ScreenNames, these are separated into BL, BG, BS (blocked) and GL, GG, GS (granted) and UU, UB, UC(unblock) and UL, UG, UD(ungrant).

```
WV12BG761ab SI=im.user.com#48815@server.com
BL=(wv:he@there.com,wv:she@there.com) BG=wv:/chatgroup@nowhere.com
BS=(("The boss",wv:/othergroup@somewhere.com)) BU=T
GL=(wv:matthias@salamander.com,wv:francisco) G

WV12BG761bb G=wv:/rock@roll.com GS=((Talkative,wv:/nowhere@there.org)) GU=F
```

8.38. Block entity transaction

8.38.1. BlockEntityRequest primitive

```
WV12BE761 SI=im.user.com#48815@server.com
GL=(wv:mary@site.com,wv:wife@home.org) BU=F GU=T
```

8.38.2. Status primitive

```
WV12ST761 SI=im.user.com#48815@server.com ST=(200,”Successfully
completed.”)
```

8.39. Create group transaction

8.39.1. CreateGroupRequest primitive

```
WV12CG761ab SI=im.user.com#48815@server.com GI=wv:john/private@there.com
GP=((NM,”Chit chat group”),(AT,Restricted),(PM,T),(SE,F),(TO,”Family,
relation ships”),(MU

WV12CG761bb ,30),(WN,”Welcome to my group. Feel free to discuss about our
current topic.”),(AD,T),(VL,60))
```

8.39.2. Status primitive

```
WV12ST761 SI=im.user.com#48815@server.com ST=200
```

8.40. Delete group transaction

8.40.1. DeleteGroupRequest primitive

```
WV12DG761 SI=im.user.com#48815@server.com GI=wv:john/private@there.com
```

8.40.2. Status primitive

```
WV12ST761 SI=im.user.com#48815@server.com ST=200
```

8.41. Join Group transaction

8.41.1. JoinGroupRequest primitive

```
WV12JG761 SI=im.user.com#48815@server.com GI=wv:/chatgroup@there.com
SN=((”-=Bart Simpson=-”,wv:/chatgroup@there.com)) JR=T SA=F
```

8.41.2. JoinGroupResponse primitive

```
WV12GJ761 SI=im.user.com#48815@server.com
JU=((Matthias,wv:mat@ny.net)),”Francisco (of the Dons)”,(”Anonymous12”
,wv:anon@foo.com)) WT=”Welcome to WV!”
```

8.42. User initiated leave group transaction

8.42.1. LeaveGroupRequest primitive

```
WV12LU761 SI=im.user.com#48815@server.com GI=wv:/chatgroup@there.com
```

8.42.2. LeaveGroupResponse primitive

```
WV12UL761 SI=im.user.com#48815@server.com ST=200 GI=wv:/chatgroup@there.com
```

8.43. Server initiated leave group transaction

8.43.1. LeaveGroupResponse primitive

```
WV12UL761 SI=im.user.com#48815@server.com ST=(809,"You have been rejected
from the group." GI=wv:/chatgroup@there.com
```

8.43.2. Status primitive

```
WV12ST761 SI=im.user.com#48815@server.com ST=200
```

8.44. Get group members' list transaction

8.44.1. GetGroupMembersRequest primitive

```
WV12GM761 SI=im.user.com#48815@server.com GI=wv:john/private@there.com
```

8.44.2. GetGroupMembersResponse primitive

```
WV12MG761 SI=im.user.com#48815@server.com AD=wv:john@smith.com
MO=(wv:matthias@salamander.com,wv:francisco)
US=(wv:he@there.com,wv:she@there.com)
```

8.45. Get Joined User's list transaction

8.45.1. GetJoinedUsersRequest primitive

```
WV09JU761 SI=im.user.com#48815@server.com GI=wv:/chatgroup@there.com
```

8.45.2. GetJoinedUsersResponse primitive

```
WV09UJ761 SI=im.user.com#48815@server.com AA=((John,wv:john@smith.com))
AE=((He,wv:he@there.com),(She,wv:she@there.com))
```

8.46. Add group member(s) transaction

8.46.1. AddGroupMembersRequest primitive

```
WV12AM761 SI=im.user.com#48815@server.com GI=wv:john/private@there.com
UI=(wv:me@home.com,wv:you@there.com)
```

8.46.2. Status primitive

```
WV12ST761 SI=im.user.com#48815@server.com ST=(200,”Successfully
completed.”)
```

8.47. Remove group member(s) transaction

8.47.1. RemoveGroupMembersRequest primitive

```
WV12RM761 SI=im.user.com#48815@server.com GI=wv:john/private@there.com
UI=(wv:me@home.com,wv:you@there.com)
```

8.47.2. Status primitive

```
WV12ST761 SI=im.user.com#48815@server.com ST=200
```

8.48. Member access rights transaction

8.48.1. MemberAccessRequest primitive

```
WV12ME761 SI=im.user.com#48815@server.com GI=wv:john/private@there.com
AD=(wv:matthias@salamander.com,wv:francisco)
MO=(wv:he@there.com,wv:she@there.com) US=wv:john@smith.com
```

8.48.2. Status primitive

```
WV12ST761 SI=im.user.com#48815@server.com ST=(200,”Successfully
completed.”)
```

8.49. Modify group properties transactions

8.49.1. GetGroupPropsRequest primitive

```
WV12GR761 SI=im.user.com#48815@server.com GI=wv:john/private@there.com
```

8.49.2. GetGroupPropsResponse primitive

```
WV12RG761ab SI=im.user.com#48815@server.com GP=((NM,”Chit chat
group”),(AT,Restricted),(PM,T),(SE,F),(TO,”Family,
relationships”),(MU,30),(WN,”Welcome to my group. Feel free to discuss
about our current topi

WV12RG761bb c.”),(AD,T),(VL,60))
OP=((PM,T),(PL,Admin),(IM,T),(AJ,F),(SI,F))
```

8.49.3. SetGroupPropsRequest primitive

```
WV12SP762 SI=im.user.com#48815@server.com GI=wv:john/private@there.com
OP=(PM,T)
```

8.49.4. Status primitive

```
WV12ST762 SI=im.user.com#48815@server.com ST=(200,”Successfully
completed.”)
```

8.50. Rejected list transactions

8.50.1. RejectListRequest primitive

```
WV12RE761 SI=im.user.com#48815@server.com GI=wv:john/private@there.com
AU=(wv:he@there.com,wv:she@there.com)
RU=(wv:matthias@salamander.com,wv:francisco)
```

8.50.2. RejectListResponse primitive

```
WV12ER761 SI=im.user.com#48815@server.com
UI=(wv:he@there.com,wv:she@there.com)
```

8.51. Subscribe group change notification transaction

8.51.1. SubscribeGroupNoticeRequest primitive (get)

```
WV12SU761 SI=im.user.com#48815@server.com GI=wv:/chatgroup@there.com SU=G
```

8.51.2. SubscribeGroupNoticeResponse primitive

```
WV12US761 SI=im.user.com#48815@server.com SS=F
```

8.51.3. SubscribeGroupNoticeRequest primitive (set)

```
WV12SG762 SI=im.user.com#48815@server.com GI=wv:/chatgroup@there.com SU=S
```

8.51.4. Status primitive

```
WV12ST762 SI=im.user.com#48815@server.com ST=(200,”Successfully
completed.”)
```

8.51.5. Group change notification primitive

```
WV12GG761 SI=im.user.com#48815@server.com GI=wv:/chatgroup@there.com
JU=(Matthias,Anonymous22) LU=((Matthias,wv:mat@ny.net),”Francisco (of the
Dons)”) GP=((AU,8)) OP=((PL,Mod))
```

8.51.6. Status primitive

```
WV12ST761 SI=im.user.com#48815@server.com ST=(200,”Successfully
completed.”)
```

8.52. Example for multiple transactions

In the following example the client encapsulates 3 transactions into two SMSes. The client:

Responds to a request (Transaction -ID 700),

Requests to join a group (Transaction-ID 701).

Responds to another request (Transaction-ID 702),

1st SMS:

```
WV12ST700 SI=im.user.com#48815@server.com ST=(200,”Successfully
completed.”) & WV12JG701ab SI=im.user.com#48815@server.com
GI=wv:/chatgroup@there.com SN=((”-=
```

2nd SMS:

```
WV12JG701bb Bart Simpson=-”,wv:/chatgroup@there.com)) JR=T SA=F & WV12ST702
SI=im.user.com#48815@server.com ST=200
```

9. Extension Framework

9.1. Extending Existing Primitives

Extended blocks MAY be added to SMS binding using the "EB" parameter. This parameter MUST be followed by a namespace. In SMS binding the order of the parameters is not significant, however we suggest using this particular parameter as the very last parameter within a primitive.

9.2. Introducing New Primitives

New primitives use the "ER" and "RE" primitives as follows:

```
WV12ER761 NS=foo.com/1.2 AA=Some_Data BB=More_Data
```

Note that the "NS" parameter MUST be the very first parameter within the primitive.

10. Static Conformance Requirement for CSP SMS Binding

Req#	Description	C-Req	S-Req	Reference
CSPSMS-1	Support for SMS encoded with UDH	O	O	
CSPSMS-2	When session is started with UDH and the server supports it (CSPSMS-1), all primitives are encoded with UDH during the session	M	M	
CSPSMS-3	Support for SMS encoded without UDH (textual)	O	O	
CSPSMS-4	When session is started without UDH and the server supports it (CSPSMS-3), all primitives are encoded without UDH during the session	M	M	
CSPSMS-5	Support for one SMS message to contain multiple WV messages	O	O	
CSPSMS-6	All primitive names, parameter names, service tree elements, presence attributes, presence values, group properties, contact list properties and search elements in a primitive are encoded with the short code(s).	M	M	
CSPSMS-7	Support for SMS Bindings over a SMS transport	O	O	
CSPSMS-8	Support for SMS encoded without UDH (textual) over a non-SMS transport	O	O	

Appendix A. Static Conformance Requirements (Normative)

The rest of the static conformance requirements for this specification are specified in [CSP SCR] and [SSP SCR].

Appendix B. Change History (Informative)

Type of Change	Date	Section	Description
Class 0	2003-Feb-21		Version 1.2