Command Line Protocol

Version 1.1

WV Internal Tracking Number: WV-031

Notice

Implementation of all or part of any Specification may require licenses under third party intellectual property rights, including without limitation, patent rights (such a third party may or may not be a Supporter). The Sponsors of the Specification are not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

THIS DOCUMENT AND THE INFORMATION CONTAINED HEREIN ARE PROVIDED ON AN "AS IS" BASIS WITHOUT WARRANTY OF ANY KIND AND ERICSSON, MOTOROLA and NOKIA DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL ERICSSON, MOTOROLA or NOKIA BE LIABLE TO ANY PARTY FOR ANY LOSS OF PROFITS, LOSS OF BUSINESS, LOSS OF USE OF DATA, INTERRUPTION OF BUSINESS, OR FOR DIRECT, INDIRECT, SPECIAL OR EXEMPLARY, INCIDENTAL, PUNITIVE OR CONSEQUENTIAL DAMAGES OF ANY KIND IN CONNECTION WITH THIS DOCUMENT OR THE INFORMATION CONTAINED HEREIN, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE. The above notice and this paragraph must be included on all copies of this document that are made.

Intellectual Property Rights have been asserted or conveyed in some manner toward these Wireless Village specifications. The Wireless Village initiatives' intellectual property rights guidelines are defined in Section 5.1 of the Wireless Village Specification Supporter Agreement. The Wireless Village initiative takes no position regarding the validity or scope of any intellectual property right or other rights that might be claimed to pertain to the implementation or use of the technology, or the extent to which any license under such rights might or might not be available. A public listing of all claims against the Wireless Village specifications, as well as an excerpt of Section 5.1 of the Wireless Village Specification Supporter Agreement, can be found at:

http://www.wireless-village.org/ipr.html

Table of Contents

1. REVISION HISTORY

Date	Issue	Description	Author
February 13th	TBD	Initial release	WV TechComm
July 31, 2002	V1.1	Version 1.1	WV TechComm

2. DOCUMENT INFORMATION

2.1. DEFINITIONS

The following definitions are intended to define terms specific to this specification, or general terms that may have some special context within the documentation. These definitions are provided to enhance your use of this documentation

The terms MAY, SHOULD, MUST are consistent with the definition in RFC 2119.

2.2. ABBREVIATIONS

For the purposes of this specification the following abbreviations apply.

ARPA	Advanced Research Projects Agency An agency of the United States Department of Defense, ARPA underwrote the development of the Internet beginning in 1969. A precursor to IETF.
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Number Authority
IETF	Internet Engineering Task Force A society of engineers and developers dedicated to designing and advancing standards for internet use.
WAP	Wireless Application Protocol A specification for a set of communication protocols to standardize the way that wireless devices, such as cellular telephones and radio transceivers, can be used for Internet access

2.3. REFERENCES

[RFC822]	“Standard for the Format of ARPA Internet Text Messages”, August 1982.
[RFC2119]	“Keywords for using RFCs to Indicate Requirements levels”, March 1997.
[RFC2778]	“A Model for Presence and Instant Messaging”, February 2000
[RFC2779]	“Instant Messaging / Presence Protocol Requirements”, February 2000
[IMPP-CPIM]	“A Common Profile for Instant Messaging (CPIM)”, Internet Draft, Nov 2000

3. OVERVIEW

The Command Line Interface (CLI) client is a legacy wireless device such as GSM phone or other device that has two-way Short Message Service (SMS) capabilities. The CLI client uses text messages to communicate with the Wireless Village (WV) server. The IMPS service provided in a CLI client is a subset of the complete functionality provided in a WV embedded client. The WV Command Line Protocol (CLP) is designed to provide the WV server and the CLI client with the communication and interaction means between each other to support the IMPS services in a CLI client. The CLP makes it possible for the CLI client to interact with the WV server via text messages. The CLP consists of commands that a user types on the device's keyboard / keypad and sends as SMS messages using WV server as an addressee. The WV server responds with the status messages that user reads on the device's display.

This document describes the semantics and the syntax of the CLP.

4. PROTOCOL INTRODUCTION

CLP is based on the architecture and model described in the *"System Architecture Model"* document and focuses on the communication and interaction between the Command Line Interface (CLI) client and the WV server via two-way text messages. The semantics of CLP is consistent with the functional description of the Command Line Protocol in the architecture model. The semantics of CLP supports a subset of the semantics of Client to Server Protocol (CSP).

4.1. TRANSACTIONS, MESSAGES AND PRIMITIVES

The CLP semantics are accomplished by "**transactions**". Transactions include one-way transactions and two-way transactions. A one-way transaction consists of a service request. A two-way transaction consists of a service request and a service response. Each service request includes a CLP **primitive** with appropriate parameters. Each service response includes a CLP primitive with the processing result, or a status primitive with error messages. Both service requests and service responses are called CLP "messages". Each type of client service request is implemented by sending the required primitives to a specific SMS address/Command Alias. Response to these messages is delivered from the same SMS address.

A CLP primitive is implemented with either a command or a set of status messages.

4.1.1 Status Primitive

The `status` primitive defines the error messages to a CLP request.

Examples:

Response Type	Messages
Successful	"IMPS: OK."
General Errors	"IMPS: Authorization failed. You are not logged in."
	"IMPS: Server is busy. Try again later."
	"IMPS: Bad request – command error"
	"IMPS: Bad request – incorrect or insufficient parameter"
	"IMPS: Service unavailable. Try again later"
	"IMPS: Service not supported"

Table 1. General Error Messages

4.2. TRANSPORT BINDING – SMS SETUP

The CLP messages are carried and transmitted via two-way text messages. Two-way Short Message Service (SMS) is the transport binding for CLP by default.

To simplify the addressing of SMS messages from the CLI client to the server, it is recommended that a short alias (4 digits or less) be used as a phonebook entry for each of the client service requests. In addition, it is possible for the device user to setup short aliases for each of his mobile contacts. This mechanism eliminates the need for the user to remember otherwise cryptic language specific acronyms for each command and eases remembering how to send specific commands. Further, with aliases for each user in a contact list, it allows the user to directly reply to IM messages from people in his contact list. The table below shows a list of the defined alias names and their purpose. It is

beyond the scope of this document to define the exact 4 digit aliases for each command, as these may be installation specific.

To support networks where the usage of multiple SMS aliases (addresses) is not allowed or desirable, user can send all commands to the same SMS address (defined by operator and supported by WV server) but put the command acronym in the beginning of SMS message. See the following table for the command acronyms.

Alias	Acronym	Purpose
WV-Login	LI	Used to send login requests and receive responses to login commands
WV-Logout	LO	Used to send logout requests and receive logout responses and asynchronous disconnections
WV-Contacts	L	Used to request and receive list of contacts that are online
WV-Add	A	Used to add contacts to the default contact list and receive confirmation of those additions
WV-Remove	R	Used to remove contacts from the default contact list and receive confirmation of those deletions
WV-Subscribe	S	Used to request subscriptions to other users' presence and is used to receive requests from other users for presence
WV-Unsubscribe	U	Used to unsubscribe to others' presence
WV-Accept	AC	Used to accept a presence request
WV-Deny	DN	Used to deny a presence request
WV-GetPresence	GP	Get presence information about a single user
WV-Presence	P	Used to update one's own presence and to receive presence updates from other users
WV-Message	M	Used to send and receive messages from user's who are not in the default contact list
WV-System		Used to send and receive help messages and to receive other system related error messages
WV-<UserID 1> … <WV-<UserID 2>	<None>	Used to send and receive messages who are in a user's default contact list.
WV-JoinGroup	JN	Used to join a group
WV-LeaveGroup	LV	Used to leave a group that client joined before
WV-MessageGroup	MG	Used to send and receive messages from group

4.3. ADDRESSING

CLP addressing schema uses the uniform Wireless Village addressing model in a unique Wireless Village address space. CLP addressing schema is consistent with that in CSP.

4.3.1 UserID

The definition of User-ID in CLP is consistent with that in CSP. Examples of the User-IDs are:

Local-User-ID: wv:Jon.Smith

wv:+123456789

wv:09876543

Global-User-ID: wv:Jon.Smith@imps.com

wv:+123456789@imps.com

wv:09876543@imps.com

4.4. CLP SERVICE SCOPE

CLP supports the session management between the CLI client and the server. The authentication and authorization are performed between the CLI client and the WV server.

CLP supports a subset of Presence Service.

CLP supports a subset of Instant Message Service.

CLP supports a subset of Group Service.

5. SESSION MANAGEMENT

5.1. PRIMITIVES

5.1.1 The "LoginRequest" Primitive – WV-Login Command

The `LoginRequest` primitive is issued from the CLI client to the server to initiate a new IM session. The `LoginRequest` primitive is implemented by sending a message to the WV-Login alias. Depending on whether the server has preprovisioned recognition of an MSISDN address, there are two optional parameters in the WV-Login command: UserID and Password.

Syntax: <UserID> <Password>

Send To: WV-Login

Example: john@im.mot.com inst1

User does not need to enter the domain part of UserID if the server is configured to support one domain only. If the server is configured to automatically recognize the MSISDN of the device, no userid or password would be required.

5.1.2 The "LoginResponse" Primitive

The `LoginResponse` primitive is issued from the server to the CLI client as a response to the WV-Login command and is received from the WV-Login alias. This primitive MUST provide confirmation or the reason for failure of the login and SHOULD provide a list of the user's contacts who are currently online. The actual text responses are implementation and language specific.

Examples:

Response Type	Messages
Successful	"IMPS: User <UserID> is logged in. Contacts Online: <User1>, <User2>, <User3>, <User4>
Errors	"IMPS: User <UserID> is unknown"
	"IMPS: Domain <IMPS-Domain-Name> is not supported"

Table 2. LoginResponse Messages

5.1.3 The "LogoutRequest" Primitive – WV-Logout Command

The `LogoutRequest` primitive is issued from the CLI client to the server in order to terminate the session with the server. The `LogoutRequest` primitive is implemented by sending a message to the WV-Logout alias. No data is required in this message – although some devices may require at least a space in the message

Syntax: ~~b~~

Sent To: WV-Logout

5.1.4 The "LogoutResponse" Primitive

The `LogoutResponse` primitive is issued from the server to the CLI client as a response to the WV-Logout command and is received from the WV-Logout alias. This primitive MUST provide confirmation or the reason for failure of the logout. The actual text responses are implementation and language specific.

Examples:

Response Type	Messages
Successful	"IMPS: User <UserID> is logged out."
Errors	"IMPS: Your are not registered."
	"IMPS: Domain <IMPS-Domain-Name> is not supported"

Table 3. LogoutResponse Messages

5.1.5 The "Disconnect" Primitive

The `Disconnect` primitive is issued from the server to the CLI client in order to terminate the session. When the server initiates a disconnection, the server SHOULD send a message to the device indicating the reason for disconnection. This message will be received from the WV-Logout Alias. The actual text response is implementation and language specific

Example: User <UserID> has been logged off because user logged on from another station.

5.2. TRANSACTIONS

5.2.1 The "Login" Transaction

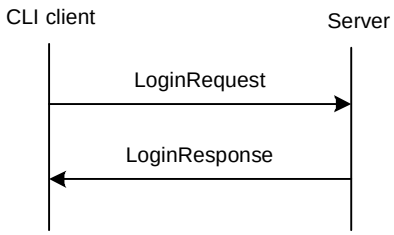

Figure 1. The "Login" Transaction

The two-way **Login** transaction is completed with `LoginRequest` primitive and `LoginResponse` primitive.

Primitive	Direction
LoginRequest	CLI client → Server
LoginResponse	CLI client ← Server

Table 4. Primitive Directions for Login Transaction

5.2.2 The “Logout” Transaction

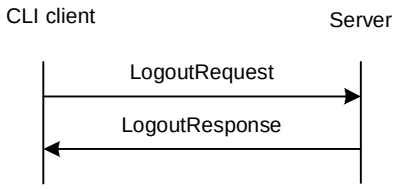

Figure 2. The “Logout” Transaction

The two-way **`Logout`** transaction is completed with `LogoutRequest` primitive and `LogoutResponse` primitive.

Primitive	Direction
LogoutRequest	CLI client → Server
LogoutResponse	CLI client ← Server

Table 5. Primitive Directions for Logout Transaction

5.2.3 The “Disconnect” Transaction

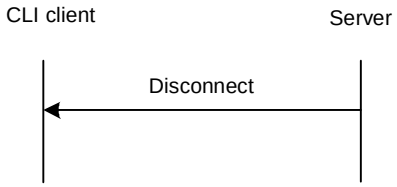

Figure 3. The “Disconnect” Transaction

The one-way **`Disconnect`** transaction is completed with `Disconnect` primitive.

Primitive	Direction
Disconnect	CLI client ← Server

Table 6. Primitive Directions for Disconnect Transaction

6. PRESENCE FEATURES

6.1. OVERVIEW

CLP supports a subset of Presence features including contact list management, proactive authorization, reactive authorization, presence subscription, update presence, get presence and presence notification.

With regard to Contact List features, while CSP supports multiple contact lists and attribute lists, CLP only supports a default contact list and a default attribute list. The CLI client may retrieve the members of, or add a member to, or remove a member from the default contact list. However, the CLP does not support create, or delete the contact list. The CLP does not support attribute list-related functions in contact list management either. The proactive authorization is achieved through adding a member to or removing a member from the contact list.

With regard to Presence features, CLP supports reactive authorization in presence subscription.

6.2. PRIMITIVES

6.2.1 The "GetContactListRequest" Primitive – WV-Contacts Command

The `GetContactListRequest` primitive is issued from the CLI client to the server to retrieve a list of all members in his contact list who are available or retrieve the presence of a single user. The server should maintain a default contact list with a default attribute list. The `GetContactListRequest` primitive is implemented by sending a message to the WV-Contacts alias. No data is required in this message, although some devices may require at least a space in the message

Syntax: [<UserID>]

Sent To: WV-Contacts

6.2.2 The "GetContactListResponse" Primitive

The `GetContactListResponse` primitive is issued from the server to the CLI client as a response to the WV-Contacts command and is received from the WV-Contacts alias. In addition to general error messages, this response should contain a list of the user's online contacts or the status of the requested user. The actual text response is implementation and language specific.

Note that due to the limit on the SMS message size, the `GetContactListResponse` message may be delivered in several SMS messages, or may be truncated.

Examples:

Response Type	Messages
Successful	"IMPS: your online contacts are <UserID 1>, … <UserID n>"
	"IMPS: <UserID> is offline
Errors	"IMPS: you do not have a default contact list"
	"IMPS: you contact list is empty"

Table 7. GetContactListResponse Messages

6.2.3 The "AddListMemberRequest" Primitive – WV-Add Command

The `AddtListMemberRequest` primitive is issued from the CLI client to the server to add a member in his contact list. The `AddListMemberRequest` primitive is implemented by sending a message to the WV-Add alias

Syntax: <UserID>

Send To: WV-Add

Example: john

6.2.4 The "AddListMemberResponse" Primitive

The `AddListMemberResponse` primitive is issued from the server to the CLI client as a response to the WV-Add command and is received from the WW-Add alias. In addition to the general error messages, the `AddListMemberResponse` MUST provide confirmation of the addition or a reason for the failure. In the confirmation, the server SHOULD provide information on the Alias assigned to this contact so that the user can enter this user into the phone's contact list. The actual text response is implementation and language specific.

Examples:

Response Type	Messages
Successful	"IMPS: <UserID> is added to your contact list"
Errors	"IMPS: User <UserID> is unknown"
	"IMPS: Domain <IM-Domain-Name> is not supported"
	"IMPS: you do not have a default contact list"
	"IMPS: you contact list is full"

Table 8. AddListMemberResponse Messages

6.2.5 The "RemoveListMemberRequest" Primitive – WV-Remove Command

The `RemoveListMemberRequest` primitive is issued from the CLI client to the server in order to remove a member from his contact list. The `RemoveListMemberRequest` primitive is implemented by sending a message to the WV-Remove alias

Syntax: <UserID>

Sent To: WV-Remove

Example: john

6.2.6 The "RemoveListMemberResponse" Primitive

The `RemoveListMemberResponse` primitive is issued from the server to the CLI client as a response to the WV-Remove command and is received from the WV-Remove alias. In addition to the general error messages, the `RemoveListMemberResponse` primitive MUST provide confirmation of the deletion or a reason for the failure. The response should also contain a reminder to have the user remove the user from his contact list. The actual text response is implementation and language specific.

Examples:

Response Type	Messages
Successful	“IMPS: <UserID> is removed from your contact list”
Errors	“IMPS: User <UserID> is unknown”
	“IMPS: Domain <IM-Domain-Name> is not supported”
	“IMPS: you do not have a default contact list”
	“IMPS: you contact list is empty”

Table 9. RemoveListMemberResponse Messages

6.2.7 The “SubscribeRequest” Primitive – WV-Subscribe Command

The `SubscribeRequest` primitive is issued from the CLI client to the server in order to establish the subscription to another user’s presence information. The server shall determine whether or not the reactive authorization is needed based on whether or not the subscriber is in the publisher’s contact list. The `SubscribeRequest` primitive is implemented by sending a message to the WV-Subscribe alias

Syntax: <UserID>

Send To: WV-Subscribe

Example: john

6.2.8 The “SubscribeResponse” Primitive

The `SubscribeResponse` primitive is issued from the server to the CLI client as a response to the WV-Subscribe command and is received from the WV-Subscribe alias. In addition to the general error messages, the `SubscribeResponse` MUST provide a confirmation that the subscription request was received by the server or of any errors that occurred in processing of this message by the local server. The actual text response is implementation and language specific.

Examples:

Response Type	Messages
Successful	“IMPS: Subscription to <UserID> is complete
Errors	“IMPS: User <UserID> is unknown”
	“IMPS: Domain <IM-Domain-Name> is not supported”

6.2.9 The “UnsubscribeRequest” Primitive – WV-Unsubscribe Command

The `UnsubscribeRequest` primitive is issued from the CLI client to the server in order to terminate the subscription to a publisher’s presence information. The `UnsubscribeRequest` primitive is implemented by sending a message to the WV-Unsubscribe Alias:

Syntax: <UserID>

Send To: WV-Unsubscribe

Example: john

6.2.10 The "UnsubscribeResponse" Primitive

The `UnsubscribeResponse` primitive is issued from the server to the CLI client as a response to the WV-Unsubscribe command and is received from the WV-Unsubscribe alias. In addition to the general error messages, the `UnsubscribeResponse` MUST contain a confirmation or reason for failure of the command. The actual text response is implementation and language specific.

Example:

Response Type	Messages
Successful	"IMPS: Unsubscribed from <UserID>
Errors	"IMPS: User <UserID> is unknown"
	"IMPS: Domain <IM-Domain-Name> is not supported"

Table 10. UnsubscribeResponse Messages

6.2.11 The "AuthorizationRequest" Primitive

The `AuthorizationRequest` primitive is issued from the server to the CLI client to ask for a reactive authorization of a subscription request. The `AuthorizationRequest` primitive is implemented with a message that indicates that the user should initiate a command to accept or deny the request. The message is received via the WV-Subscribe alias. The actual text response is implementation and language specific. The implementation should allow the forwarding of this message to the WV-Accept and WV-Deny commands for easy replies.

Example: "<UserID> is subscribing to your presence information. Please reply: accept (AC) or deny (DE)?"

6.2.12 The "AcceptSubscription" Primitive – WV-Accept Command

The `AcceptSubscription` primitive is issued from the CLI client to the server as a response to the reactive authorization `AuthorizationRequest` request in order to accept the subscription. The "*AcceptSubscription*" primitive is implemented by sending a message to the WV-Accept alias:

Syntax: <UserID>

Sent To: WV-Accept

Example: john

6.2.13 The "DenySubscription" Primitive – WV-Deny Command

The `DenySubscription` primitive is issued from the CLI client to the server as a response to the reactive authorization `AuthorizationRequest` request in order to deny the subscription. This command can also be used by a client to withdraw authorization of presence information at any time. The `DenySubscription` primitive is implemented by sending a message to the WV-Deny alias:

Syntax: <UserID>

Sent To: WV-Deny

Example: john

6.2.14 The "DenySubscriptionResponse" Primitive

The `DenySubscriptionResponse` primitive is issued from the server to the CLI client as a response to the WV-Deny command and is received from the WV-Deny alias. In addition to the general error messages, the `DenySubscriptionResponse` primitive is implemented with the following response messages. The actual text response is implementation and language specific.

Response Type	Messages
Successful	"IMPS: Authorization for <User-ID> is denied."
Errors	"IMPS: User <UserID> is unknown"
	"IMPS: Domain <IM-Domain-Name> is not supported"

Table 11. DenySubscriptionResponse Mess ages

6.2.15 The "Presence" Primitive

The `Presence` primitive is issued from the server to the CLI client to provide a contact's presence information. It could be generated in response to a GetPresence request, or from an asynchronous notification when a contact's status changes. The presence primitive is received from the WV-Presence alias. The `Presence` primitive should indicate the presence status of the contact(s) (i.e. Online, Available, Not Available), the User-ID(s) of the contact(s) and any other status information which is desired for the implementation. The actual text response is implementation and language specific.

Example:

Message Type	Messages
Successful	"IMPS: 1-<Short-Status>-<UserID>-[(<Custom-Status>)] …"
Notification	"IMPS: User <UserID> is <Status> [(<Custom-Status>)]"
Error	"IMPS: <UserID>'s presence is unchanged"
	"IMPS: User <UserID> is unknown"
	"IMPS: Domain <IM-Domain-Name> is not supported"

Table 12. Presence Messages

In this table <Short-Status> is a placeholder for one of one-character status values:

"O" – user is offline

"A" – user is available

"N" – user is not available, but online

Example: IMPS: 1-N-john-(Will be back soon) 2-A-mike 3-O-kate

See Appendix A for the matching of CSP presence attributes and one-character status values.

Note: due to limitations on the size of SMS message, the result of presence message may be delivered in several SMS messages or may be truncated.

6.2.16 The "UpdatePresence" Primitive – WV-Presence Command

The `UpdatePresence` primitive is issued from the CLI client to the server in order to update the presence information. The `UpdatePresence` primitive is implemented by sending a message to the WV-Update alias

Syntax: <Short-Status> [<Custom-Status>]

Send To: WV-Presence

where, the`ShortStatus` is a one-character string representing one of the following three types of status – O(ffline), A(vailable), or N(ot available). The optional`CustomStatus` is a string describing user-defined status. See "Presence" description.

Example: N Will be back soon

6.2.17 The "GetPresence" Primitive

With the **Get Presence** transaction the current presence values for a contact is fetched without or in parallel with a presence subscription. The return value is a `Presence` Primitive as described in section 5.2.15

Syntax: <UserID>

Sent To: WV-GetPresence

Example: john

6.3. TRANSACTIONS

6.3.1 The "GetContactList" Transaction

Figure 4. The "GetContactList" Transaction

Primitive	Direction
GetContactListRequest	CLI client → Server
GetContactListResponse	CLI client ← Server

Table 13. Primitive Directions for GetContactList Transaction

6.3.2 The "AddListMember" Transaction

Figure 5. The "AddListMember" Transaction

Primitive	Direction
AddListMemberRequest	CLI client → Server
AddListMemberResponse	CLI client ← Server

Table 14. Primitive Directions for AddListMember Transaction

6.3.3 The "RemoveListMember" Transaction

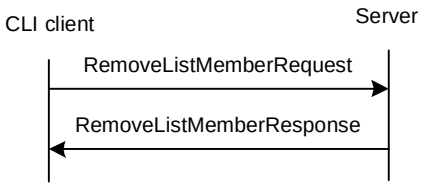

Figure 6. The "RemoveListMember" Transaction

Primitive	Direction
RemoveListMemberRequest	CLI client → Server
RemoveListMemberResponse	CLI client ← Server

Table 15. Primitive Directions for RemoveListMember Transaction

6.3.4 The "Subscribe" Transaction

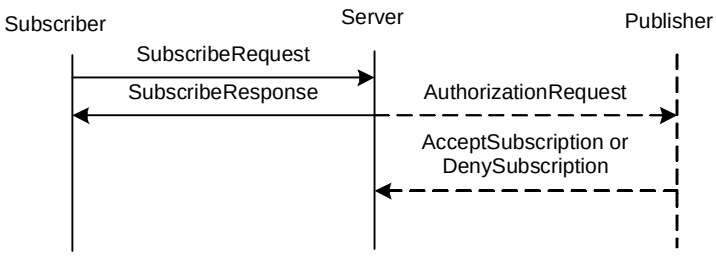

Figure 7. The "Subscribe" Transaction

The two-way **Subscribe** transaction is completed with `SubscribeRequest` primitive and `SubscribeResponse` primitive. During the transaction, the server shall determine

whether or not the reactive authorization is needed based on whether or not the subscriber is in the publisher's contact list. A separate two-way transaction "ReactiveAuthorization" may be needed for the server to communicate with the publisher to accomplish the entire processing.

Primitive	Direction
SubscribeRequest	CLI client → Server
SubscribeResponse	CLI client ← Server

Table 16. Primitive Directions for Subscribe Transaction

6.3.5 The "ReactiveAuthorization" Transaction

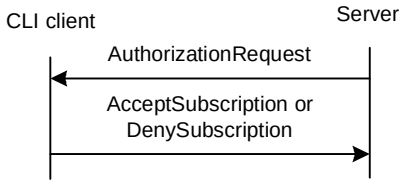

Figure 8. The "ReactiveAuthorization" Transaction

Primitive	Direction
AuthorizationRequest	CLI client ← Server
AcceptSubscription or DenySubscription	CLI client → Server

Table 17. Primitive Directions for ReactiveAuthorization Transaction

6.3.6 The "Unsubscribe" Transaction

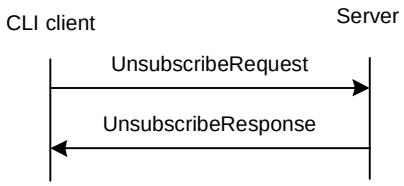

Figure 9. The "Unsubscribe" Transaction

The two-way **Unsubscribe** transaction is completed with the `UnsubscribeRequest` primitive and `UnsubscribeResponse` primitive.

Primitive	Direction
UnsubscribeRequest	CLI client → Server
UnsubscribeResponse	CLI client ← Server

Table 18. Primitive Directions for Unsubscribe Transaction

6.3.7 The "WithdrawAuthorization" Transaction

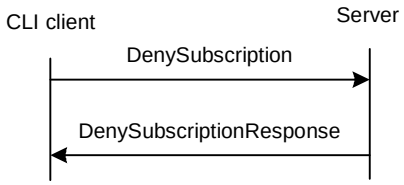

Figure 10. The "WithdrawAuthorization" Transaction

Primitive	Direction
DenySubscription	CLI client → Server
DenySubscriptionResponse	CLI client ← Server

Table 19. Primitive Directions for WithdrawAuthorization Transaction

The **WithdrawAuthorization** transaction is usually used to withdraw the reactive authorization. The proactively authorization is withdrawn by removing the user from the contact list.

6.3.8 The "NotifyPresence" Transaction

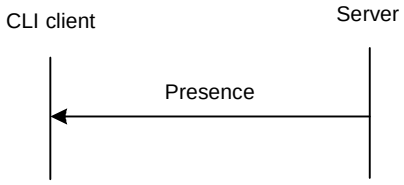

Figure 11. The "NotifyPresence" Transaction

The one-way **NotifyPresence** transaction is completed with `Presence` primitive notification message.

Primitive	Direction
Presence - Notification	CLI client ← Server

Table 20. Primitive Directions for NotifyPresence Transaction

6.3.9 The "UpdatePresence" Transaction

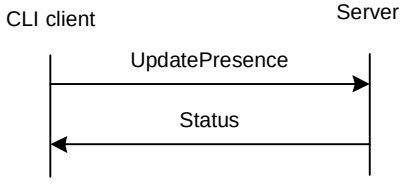

Figure 12. The "UpdatePresence" Transaction

The two-way **UpdatePresence** transaction is completed with `UpdatePresence` primitive and `Status` primitive.

Primitive	Direction
UpdatePresence	CLI client → Server
Status	CLI client ← Server

Table 21. Primitive Directions for UpdatePresence Transaction

6.3.10 The “GetPresence” Transaction

Figure 13. The “GetPresence” Transaction

Primitive	Direction
GetPresence	CLI client → Server
Presence	CLI client ← Server

Table 22. Primitive Directions for GetPresence Transaction

7. INSTANT MESSAGING FEATURES

7.1. OVERVIEW

CLP supports a subset of IM features including sending and delivery of message. Messages can be sent to and received by users within a predefined contact list via specific aliases for each contact or can be sent or received from users outside the user's contact list

7.2. PRIMITIVES

7.2.1 The "SendMessageRequestContact" Primitive – WV-<UserID> Command

The `SendMessageRequest` primitive is issued from the CLI client to the server in order to send an instant message to another IMPS user who resides in your default contact list and who has been setup with a predefined alias. The `SendMessageRequesContact` primitive is implemented by sending a message to this WV<UserID> alias:

Syntax: <Message-Text>

Sent-To: WV-<UserID>

Example: Hi John, how are you?

Sent-To: WV-John

This command sends instant message 'Hi John, how are you?' to the user with UserID 'john'.

7.2.2 The "SendMessageRequestUnlisted" Primitive – WV-Message Command

The `SendMessageRequestUnlistedt` primitive is issued from the CLI client to the server in order to send an instant message to another IMPS user who does not reside in your contact list. The `SendMessageRequestContact` primitive is implemented by sending a message to this WV-Message alias:

Syntax: <UserID> <Message-Text>

Sent-To: WV-Message

Example: john Hi John, how are you?

Sent-To: WV-Message

This command sends instant message 'Hi John, how are you?' to the user with UserID 'john'.

7.2.3 The "NewMessageContact" Primitive

The `NewMessageContact` primitive is issued from the server to the CLI client to deliver an instant message from another user who is in the user's default contact list. The `NewMessageContact` primitive is received from the WV-<UserID> alias. This allows the user to simply reply to the message to deliver the message back through the sender's alias into the IM system.

Example:

Message Type	Messages
Notification	“IMPS: From <UserID>: <Message-text>”

Table 23. NewMessageContact Messages

Note: due to limitations on the size of SMS message, the result of the message text may be delivered in several SMS messages or may be truncated.

7.2.4 The “NewMessageUnlisted” Primitive

The `NewMessageUnlisted` primitive is issued from the server to the CLI client to deliver an instant message from a user who is not in the user’s default contact list. The `NewMessageUnlisted` primitive is received from the WV-Unlisted alias. Direct replies to these messages are not permitted; instead a new message needs to be created and sent using the WV-Message command. The `NewMessageUnlistedt` primitive will also be used to return error messages that may occur in sending messages using the WV-Message or WV-<UserID> commands.

Example:

Message Type	Messages
Notification	“IMPS: UNLISTED From <UserID>: <Message-text>”
Errors	“IMPS: Error User <UserID> is unknown”
	“IMPS: Error Domain <IM-Domain-Name> is not supported”

Table 24. NewMessageUnlisted Messages

Note: due to limitations on the size of SMS message, the result of the message text may be delivered in several SMS messages or may be truncated.

7.2.5 The “MessageDelivery” Primitive

The `MessageDelivery` primitive is issued from the server to the CLI client when a request to send an instant message has failed. The `MessageDelivery` primitive is received from the WV-Message alias. It should contain a description of why the message failed. The actual text is implementation and language specific.

Example:

Message Type	Messages
Errors	“IMPS: Error: User <UserID> is unknown”
	“IMPS: Error: Domain <IM-Domain-Name> is not supported”

Table 25. MessageDelivery Messages

7.3. TRANSACTIONS

7.3.1 The “SendMessageContact” Transaction

Figure 13. The “SendMessageContact” Transaction

The “SendMessageContact” transaction is completed with `SendMessageRequesContact` primitive.

Primitive	Direction
SendMessageRequest	CLI client → Server
MessageDelivery	Server →CLI Client (on failure)-

Table 26. Primitive Directions for SendMessageContact Transaction

7.3.2 The “SendMessageUnlisted” Transaction

Figure 14. The “SendMessageUnlisted” Transaction

The “SendMessageUnlisted” transaction is completed with `SendMessageRequestUnlisted`

Primitive	Direction
SendMessageRequestUnlisted	CLI client → Server
MessageDelivery	Server → CLI Client (on failure)

Table 27. Primitive Directions for SendMessageUnlisted Transaction

7.3.3 The “NewMessageContact” Transaction

Figure 15. The “NewMessageContact” Transaction

The one-way `NewMessageContact` transaction is completed with `NewMessageContact` primitive.

Primitive	Direction
NewMessageContact	CLI client ← Server

Table 28. Primitive Directions for NewMessageContact Transaction

7.3.4 The “NewMessageUnlisted” Transaction

Figure 16. The “NewMessageUnlisted” Transaction

The one-way **NewMessageUnlisted** transaction is completed with `NewMessageUnlisted` primitive.

Primitive	Direction
NewMessageUnlisted	CLI client ← Server

Table 29. Primitive Directions for NewMessageUnlisted Transaction

8. GROUP FEATURE

8.1. OVERVIEW

CLP supports a subset of Group features including joining and leaving a group, sending and receiving messages from a group. Only partial functionality of these features is supported.

User can join a group, send messages to this group, receive messages from this group and leave it. User may not sent or receive messages from the group he has not joined. User can join only one group at the same time, and he/she has to leave group to join another group.

8.2. PRIMITIVES

8.2.1 The "JoinGroupRequest" Primitive – WV-JoinGroup Command

The `JoinGroupRequest` primitive is issued from the CLI client to the server in order to indicate to the server that client wants to join the group. The `JoinGroupRequest` primitive is implemented by sending a message to this WV-JoinGroup alias:

Syntax: <Group-Name>

Sent-To: WV-JoinGroup

Example: wireless-village

Sent-To: WV-JoinGroup

This command indicates that user wants to join the group named "wireless-village".

8.2.2 The "JoinGroupResponse" Primitive

The `JoinGroupResponse` primitive is issued from the server to the CLI client as a response to the `JoinGroupRequest` primitive and is received from the WW-JoinGroup alias. In addition to the general error messages, the `JoinGroupResponse` MUST provide confirmation or a reason for the failure. The actual text response is implementation and language specific.

Message Type	Messages
Errors	"IMPS: Group <Group-Name> is unknown"
	"IMPS: Insufficient user rights"
	"IMPS: Already joined group <Group-Name>"
	"IMPS: Maximum number of users has been reached"

Table 30. JoinGroupResponse Messages

8.2.3 The "LeaveGroupRequest" Primitive – WV-LeaveGroup Command

The `LeaveGroupRequest` primitive is issued from the CLI client to the server in order to indicate to the server that client wants to leave the group. The `LeaveGroupRequest` primitive is implemented by sending a message to this WV-LeaveGroup alias. Client

doesn't have to specify a group name because it may be joined to only one group at the same time.

Syntax: <empty>

Sent-To: WV-JoinGroup

Message Type	Messages
Errors	"IMPS: <Group-Name>: Not joined"

Table 31. JoinGroupRequest Messages

8.2.4 The "SendMessageGroupRequest" Primitive – WV-MessageGroup Command

The `SendMessageGroupRequest` primitive is issued from the CLI client to the server in order to send an instant message to the group. The `SendMessageGroupRequest` primitive is implemented by sending a message to WV-MessageGroup alias:

Syntax: <Message-Text>

Sent-To: WV-MessageGroup

Example: Hi John, how are you?

Sent-To: WV-MessageGroup

This command sends instant message 'Hi John, how are you?' to the group.

8.2.5 The "NewMessageGroup" Primitive

The `NewMessageGroup` primitive is issued from the server to the CLI client to deliver an instant message sent to the group joined by the user. The `NewMessageGroup` primitive is received from the WV-MessageGroup alias. This allows the user to simply reply to the message to deliver the message back through the WV-MessageGroup alias to the group.

Example:

Message Type	Messages
Notification	"IMPS: From <Group-Name>: <Message-text>"

Table 32. NewMessageGroup Messages

Note: due to limitations on the size of SMS message, the resulting message text may be delivered in several SMS messages or may be truncated.

8.2.6 The "MessageGroupDelivery" Primitive

The `MessageGroupDelivery` primitive is issued from the server to the CLI client when a request to send an instant message to a group has failed. The `MessageGroupDelivery` primitive is received from the WV-MessageGroup alias. It should contain a description of why the message failed. The actual text is implementation and language specific.

Example:

Message Type	Messages
Errors	“IMPS: Error: Not joined any group”

Table 33. MessageGroupDelivery Messages

8.3. TRANSACTIONS

8.3.1 The “JoinGroup” Transaction

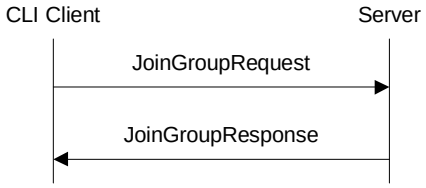

Figure 17. The “JoinGroup” Transaction

The **JoinGroup** transaction is completed with `JoinGroupRequest` and `JoinGroupResponse` primitives.

Primitive	Direction
JoinGroupRequest	CLI client → Server
JoinGroupResponse	Server → CLI Client (on failure)

Table 34. Primitive Directions for JoinGroup Transaction

8.3.2 The “LeaveGroup” Transaction

Figure 18. The “LeaveGroup” Transaction

The one-way **LeaveGroup** transaction is completed with `LeaveGroupRequest` primitive notification message.

Primitive	Direction
LeaveGroupRequest – Notification	CLI client → Server

Table 35. Primitive Directions for LeaveGroup Transaction

8.3.3 The “SendMessageGroup” Transaction

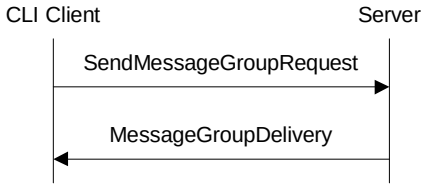

Figure 19. The “SendMessageGroup” Transaction

The **SendMessageGroup** transaction is completed with `SendMessageGroupRequest`

Primitive	Direction
SendMessageGroupRequest	CLI client → Server
MessageGroupDelivery	Server → CLI Client (on failure)

Table 36. Primitive Directions for SendMessageGroup Transaction

8.3.4 The “NewMessageGroup” Transaction

Figure 20. The “NewMessageGroup” Transaction

The one-way **NewMessageGroup** transaction is completed with `NewMessageGroup` primitive.

Primitive	Direction
NewMessageGroup	CLI client ← Server

Table 37. Primitive Directions for NewMessageGroup Transaction

9. MISCELLANEOUS FEATURES

9.1. PRIMITIVES

9.1.1 The "GetHelp" Primitive – WV-System Command

The `GetHelp` primitive is issued from the CLI client to the server in order to get the help with the syntax and aliases for the CLP commands and user's contacts. The `GetHelp` primitive is implemented by sending a message to the WV-System Alias:

Syntax: [<Command-Name> or <UserID>]

Send To: WV-System

Where *Command-Name* is one of the CLP commands or <UserID> is one of the user's in the user's default contact list. If the command is used without parameter the server returns the list of all commands and contacts along with their aliases.

Example: WV-Login

9.1.2 The "Help" Primitive

The `Help` primitive is issued from the server to the CLI client as a response to the WV-System command and is received over the WV-System alias. In addition to the general error messages, the `Help` primitive **must** either list each of the supported commands and their aliases, or list a specific command along with its alias and syntax. The actual text is implementation and language specific.

Example:

Response Type	Messages
Successful	"IMPS Help: Send <UserID> <Password> to WV-Login (Alias 9901)"
Errors	"IMPS: Unknown command <Command-Name>"

Table 38. Help Messages

Note: due to limitations on the size of SMS message, the result of the help text may be delivered in several SMS messages or may be truncated.

9.2. TRANSACTIONS

9.2.1 The "Help" Transaction

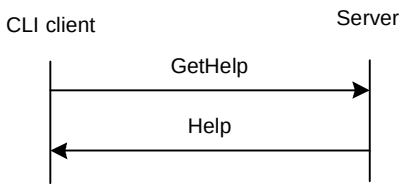

Figure 21. The "Help" Transaction

The two-way “Help” transaction is completed with `GetHelp` primitive and `Help` primitive.

Primitive	Direction
GetHelp	CLI client → Server
Help	CLI client ← Server

Table 39. Primitive Directions for Help Transaction

10. EXAMPLE OF AN IMPS SESSION

10.1. CREATE A SESSION

SEND TO WV-LOGIN: “john 1234”

RECV FROM WV-LOGIN: “IMPS: User john is logged in to imps.wv.com domain”

10.2. CONTACT LIST MANAGEMENT

SEND TO WV-CONTACTS: “”

RECV FROM WV-CONTACTS: “IMPS: your contact List is empty”

SEND TO WV-ADD: “mike”

RECV FROM WV-ADD: “IMPS: mike is added to your contact list as alias 9801”

SEND TO WV-ADD: “mark”

RECV FROM WV-ADD : “IMPS: mark is added to your contact list as alias 9802”

10.3. GET PRESENCE

SEND TO WV-CONTACTS: “mark, mike”

RECV FROM WV-CONTACTS: “1-O-mike 2-A-mark”

SEND TO WV-SUBSCRIBE: “mike”

RECV FROM WV-SUBSCRIBE: “IMPS: Subscription to mike is complete”

RECV FROM WV-PRESENCE: “IMPS: User mike is Offline”

……

RECV FROM WV-PRESENCE: “IMPS: User mike is Available”

SEND TO WV-UNSUBSCRIBE: “mike”

RECV FROM WV-UNSUBSCRIBE: “IMPS: Unsubscribed from mike”

10.4. INSTANT MESSAGING

SEND TO WV-mike (e.g. alias 9801): “Hi Mike, this is John, how are you”

RECV FROM WV-mike: “IMPS: From mike: I’m fine, John, how are you?”

Respond to Message: “Fine, thank you. I need to talk to you”

RECV FROM WV-mike: “IMPS: From mike: I’m busy now. Let’s talk tomorrow”

10.5. TERMINATE THE SESSION

SEND TO WV-LOGOUT: “”

RECV FROM WV-LOGOUT: “IMPS: User john is logged out”

11. APPENDIX A. CSP PRESENCE ATTRIBUTES AND ONE-CHARACTER STATUS VALUES

To support the simplified CLP presence information, server has to perform the conversion between CSP presence attributes and CLP one-character status values according to the following table.

CSP Presence Attribute		CLP status value
OnlineStatus	**UserAvail**	
F	any	O
T	AVAILABLE	A
T	NOT_AVAILABLE	N
T	DISCREET	N

All CSP presence attributes not listed in the table should be ignored.