LifeSocial Commands

Report



TECHNISCHE UNIVERSITÄT DARMSTADT

Department of Electrical Engineering
and Information Technology
Department of Computer Science
(Adjunct Professor)

Multimedia Communications Lab - KOM
Prof. Dr.-Ing. Ralf Steinmetz

Submitted by
Luciana Alvite
on 08.07.2009

Advisor
Prof. Dr.-Ing. Ralf Steinmetz

Tutor
Kalman Graffi

Contents

1 List of LifeSocial commands 1

2 Starting the platform 9
2.1 Test mode 9
2.2 Live mode 9

A Commands Parameters 11
A.1 Profile Plugin 11

B Status Messages 13
B.1 Network Plugin 13

List of Tables

1.1 Network commands 1
1.2 Login Plugin commands 2
1.3 Profile Plugin commands 2
1.4 Friends Plugin commands 3
1.5 Groups Plugin commands 4
1.6 Live Chat Plugin commands 5
1.7 Messaging Plugin commands 5
1.8 Search Plugin commands 6
1.9 Photos Plugin commands 7
1.10 Whiteboard Plugin commands 8
A.1 Profile Fields 11
B.1 Network Status Messages 14
B.2 Login Plugin Status Messages 14

1 List of LifeSocial commands

This is a list of the commands currently[1] available in the LifeSocial.KOM platform. The status messages returned for each case can be found in the Appendix B.

Network commands			
Command	Arguments	Description	Example
Connect	(no args)	connects using localhost and standard port 9001	Connect
Connect	<localport>	connects using localhost and given port	Connect 9013
Connect	<localport> <bindaddress>	connects using given host and port	Connect 9013 130.149.105.12
Connect	<localport> <bindport> <bindaddress>	connects to given host and bind port and using given local port	Connect 9013 9001 130.149.105.12
ConnectP	<localport> <bindport>	connects using localhost and given local and bind ports	ConnectP 9013 9001
Disconnect	(no args)	disconnects from the LifeSocial network	Disconnect
IsConnected	(no args)	check if user is connected to the LifeSocial network	IsConnected

Table 1.1: Network commands

Login Plugin commands			
Command	Arguments	Description	Example
Login	<username> <password>	logs in. Must be connected first.	Login alice tudlifesocial
Logout	(no args)	logs the current user out (must be logged in)	Logout
IsLoggedIn	(no args)	returns true if the user is logged in	IsLoggedIn

Continued on next page

[1] Status: 28th September 2009

Table 1.2 – continued from previous page

Login Plugin commands			
Command	Arguments	Description	Example
LoginName	(no args)	shows the username of the currently user	LoginName
Recover	<username> (or/and) <email>	recover user password	Recover alice or Recover alice@yahoo.com
ChangePassword	<old pass><new pass>	changes the user password	ChangePassword tudlifesocial alicepass

Table 1.2: Login Plugin commands

Profile Plugin commands			
Command	Arguments	Description	Example
Register	<username> <password> <email> <country> <city>	creates a new user account with given register information	Register alice tudlifesocial alice@yahoo.com germany darmstadt
EditProfileField[1][2]	<field name> <new value>	changes the value for the given field	EditProfileField ProfileCity Berlin
EditProfileFields[1][2]	n* (<field name> <new value>)	changes the values for the given fields	EditProfileFields Profile.Email a@yahoo.com\nProfile.Interests \nProfileCity Berlin
RequestProfile	<username>	loads the profile of the user	RequestProfile john
ProfileCompletelyLoaded	(no args)	indicates whether previously requested profile is already loaded	ProfileCompletelyLoaded
ShowCurrentProfile	(no args)	returns the previously requested profile	ShowCurrentProfile
GetMyProfile	(no args)	returns own profile	GetMyProfile
MyProfileCompletelyLoaded	(no args)	indicates whether own profile is already loaded	MyProfileCompletelyLoaded
StoreMyProfile	(no args)	stores own profile	StoreMyProfile

Table 1.3: Profile Plugin commands

[1] See Appendix A for more information about currently available field names
[2] This Command does NOT store the profile on the network, need to call StoreMyProfile after changes

Friends Plugin commands			
Command	Arguments	Description	Example
GetMyFriendsList	(no args)	returns user's own friends list	GetMyFriendsList
MyFriendsListLoaded	(no args)	indicates whether my friends list is loaded	MyFriendsListLoaded
GetMyFriendsListAndWait	(no args)	requests own list and waits for its arrival	GetMyFriendsListAndWait
GetFriendsListHost	(no args)	returns the username of the current list owner	GetFriendsListHost
LoadFriendsList	<username>	loads the list of friends of the given user	LoadFriendsList Tina
FriendsListLoaded	(no args)	indicates whether the requested list is loaded	FriendsListLoaded
GetFriends	(no args)	returns the list of user names of the current friends' list	GetFriends (returns list of Tina's friends)
RequestFriend	<username><reason>	sends an friendship request to the defined user	RequestFriend Tina Hey!
AcceptFriend	<request _key><initiator>	accepts the request and adds the user to own list	AcceptFriend ef6243 Alice
DeclineFriend	<key> <initiator>	declines the given friendship request	DeclineFriend ef6243 Alice
DeleteFriend	<user name>	removes a friend from the friends' list	DeleteFriend Tina
GetFriendsMsgs	(no args)	returns the friends msgs/requests	GetFriendsMsgs
CheckForUndeliveredMsgs	(no args)	checks for undelivered messages for the Friends plugin	CheckForUndeliveredMsgs
MessagesAvailable	(no args)	indicates whether new messages for the Friends Plugin are available	MessagesAvailable

Table 1.4: Friends Plugin commands

Groups Plugin commands			
Command	Arguments	Description	Example
LoadGroups	<user name>	loads a list of groups for the given user	LoadGroups John
UserGroupsLoaded	(no args)	indicates whether the list of user groups is completely loaded	UserGroupsLoaded
GetGroupsList	(no args)	returns the list of groups currently loaded	GetGroupsList (returns list of John's groups)
LoadMyGroups	(no args)	loads my list of groups	LoadMyGroups
MyUserGroupsLoaded	(no args)	indicates whether the list of user groups is completely loaded	MyUserGroupsLoaded

Continued on next page

Table 1.5 – continued from previous page

Groups Plugin commands			
Command	Arguments	Description	Example
GetMyGroupsList	(no args)	returns my list of groups currently loaded	GetMyGroupsList (displays list of groups the user belongs to)
JoinGroup	<group name>	adds the current user to the members list of the given group	JoinGroup Berlin
LeaveGroup	<group name>	deletes the current user from the members list of the given group	LeaveGroup Darmstadt
CreateGroup	<group name>	creates a new group if the group name is free	CreateGroup TUDinBerlin
GroupCreated	(no args)	informs if the previous group creation was a success	GroupCreated
LoadSingleGroup	<group name>	loads the list of members for the defined group	LoadSingleGroup Berlin
SingleGroupLoaded	(no args)	indicates whether the requested single group is already loaded	SingleGroupLoaded (in this case, Berlin)
ShowCurrentGroupMembers	(no args)	returns a list of group members of the currently available group	ShowCurrentGroupMembers (returns list of users in "Berlin")
MessagesAvailable	<no args>	indicates whether new messages for the Groups Plugin are available	MessagesAvailable
GroupsGetMsgBox	(no args)	returns a list of keys of group msgs	GroupsGetMsgBox
GetGroupsMsg	<key of a msg>	returns a single group msg for the given key	GetGroupsMsg ae6492ab
GetGroupsHost	(no args)	returns the user name whose group list is currently available	GetGroupsHost
SendGroupMsg	<groupname><msg>	sends msg to all participants of the given group	SendGroupMsg Berlin Hello all, I am new in Berlin!
CheckForUndeliveredMsgs	(no args)	checks for undelivered messages for the Groups plugin	CheckForUndeliveredMsgs

Table 1.5: Groups Plugin commands

Live Chat Plugin commands			
Command	Arguments	Description	Example
HistoryUsers	(no args)	returns list of usernames that have a chat history with the user(e.g. John, Tina)	HistoryUsers
HistoryForUser	<username>	returns the history for chat with given username	HistoryForUser Tina
MessagesAvailable	(no args)	indicates whether new messages for the Chat Plugin are available	MessagesAvailable
CheckForUndeliveredMsgs	(no args)	checks if there are undelivered messages for the Chat Plugin	CheckForUndeliveredMsgs
SendMsg	<recipient> <message>	sends a chat message to the given user	SendMsg Tina Hey Tina, thank you for your message.
GetChatMsgs	(no args)	returns chat messages which are available	GetChatMsgs

Table 1.6: Live Chat Plugin commands

Messaging Plugin commands			
Command	Arguments	Description	Example
GetInbox	(no args)	returns message ids available in inbox	GetInbox
ShowInboxMsg	<msg_key>	returns the content of the 'inbox' (not group) message for the given message key	ShowInboxMsg abc
GetOutbox	(no args)	returns message ids available in outbox	GetOutbox
ShowOutboxMsg	<msg key>	returns 'sent' message for the given message key	ShowOutboxMsg efd
CheckMsgs	(no args)	loads undelivered messages for the Messaging plugin	CheckMsgs
MessagesAvailable	(no args)	returns true if new messages are available	MessagesAvailable
DeleteInboxMsg	<msg key>	deletes an inbox message with the given message key	DeleteInboxMsg abc
DeleteOutboxMsg	<msg key>	deletes a sent message with the given message key	DeleteOutboxMsg efd
SendMsg	<username> <msg>	sends a message (Messaging plugin)	SendMsg John Hey, John, would you like to join the Berlin group?

Table 1.7: Messaging Plugin commands

Search Plugin commands			
Command	Arguments	Description	Example
Search	<criteria> (search criteria. e.g. city)	searches for usernames based on given search criteria	Search Berlin
SearchIsComplete	<ticket>	indicates whether previously executed search is complete	SearchIsComplete 3
GetSearchResults	<ticket>	returns a list of users retieved with previous 'Search'. ticket is the unique identifier returned when executing 'Search'	GetSearchResults 3

Table 1.8: Search Plugin commands

Photos Plugin commands			
Command	Arguments	Description	Example
CreateNewAlbum	<title> <comments>	<title> <comments> - creates a new album with given title and comments. Terminate title with \n. Space clears the field.	CreateNewAlbum Darmstadt\n these are some pictures of Darmstadt
GetAlbums	(no args)	returns the collection of albums of the current host	GetAlbums
AddPhoto	<album key> <title> <path> <comments>	adds a photo to the defined album. path is absolute. Terminate title and path with \n	AddPhoto 64aeb TUD\n C:\MyPictures\tud.jpg\nThis is the University
EditAlbum	<album key><title><comments>	changes album information. Terminate title with \n Space clears the field	EditAlbum 64aeb TUD\nPictures of the TUD
EditPhoto	<title> <comments>	changes the info for the current photo. Terminate title with \n. Space clears the field.	EditPhoto ElectricalEngineering\n This is where I study
DeletePhoto	<album key> <photo key>	deletes a photo with the given keys	DeletePhoto 64aeb 830123
DeleteAlbum	<album key>	deletes an album with the given key	DeleteAlbum 64aeb
LoadUserAlbums	<username>	loads the albums for the given user	LoadUserAlbums John
LoadMyAlbums	(no args)	loads own albums	LoadMyAlbums
AlbumsLoaded	(no args)	indicates whether the requested list of albums is already loaded	AlbumsLoaded
GetAlbumHost	(no args)	returns the host of the currently loaded album	GetAlbumHost (John in this sequence)
LoadPhoto	<photo_key>	loads a photo with the defined key	LoadPhoto 12547
PhotoLoaded	(no args)	indicates whether the requested photo is already loaded	PhotoLoaded

Continued on next page

Table 1.9 – continued from previous page

Photos Plugin commands			
Command	Arguments	Description	Example
GetPhoto	(no args)	returns the currently loaded photo	GetPhoto

Table 1.9: Photos Plugin commands

Whiteboard Plugin commands			
Command	Arguments	Description	Example
CheckForUndeliveredMsgs	(no args)	checks for undelievered messages for the whiteboard plugin	CheckForUndeliveredMsgs
MessagesAvailable	(no args)	indicates if new msgs are available for the whiteboard plugin	MessagesAvailable
JoinWhiteboard	(whiteboard name)	joins whiteboard with given name	JoinWhiteboard JohnAliceWhiteboard
ListMyWhiteboards	(no args)	returns the whiteboards the user is currently a member of (as a Set)	ListMyWhiteboards
SendLine	(whiteboard name) (color) (thickness) (start x) (start y) (end x) (end y)	sends a line with specified parameters to given whiteboard. Color must be in rgb (0-255)	SendLine JohnAliceWhiteboard 2 1 10 10 50 25
SendPoint	(whiteboard name) (color) (thickness) (x) (y)	sends a point with specified parameters to given whiteboard. All parameters but group name must be numbers (int) as String. Color must be in rgb (0-255).	SendPoint JohnAliceWhiteboard 1 2 30 56
SendPolyline	(whiteboard name) n* ((x) (y))	sends a list of points to the given whiteboard. X and Y must be numbers (int) as string. Terminate every x,y pair with \n	SendPolyline JohnAliceWhiteboard 10 10\n50 25\n20 30
ListShapesForWhiteboard	(whiteboard name)	returns the shapes drawn on given whiteboard	ListShapesForWhiteboard JohnAliceWhiteboard
ListWhiteboardMembers	(whiteboard name)	returns the users currently in the given whiteboard	ListWhiteboardMembers JohnAliceWhiteboard
LeaveWhiteboard	(whiteboard name)	leaves whiteboard with given name	LeaveWhiteboard JohnAliceWhiteboard

Continued on next page

Table 1.10 – continued from previous page

Whiteboard Plugin commands			
Command	Arguments	Description	Example

Table 1.10: Whiteboard Plugin commands

2 Starting the platform

To use the LifeSocial Platform, the user must have an OSGi implementation installed, like [Fou03]. After installing OSGi, the user has to make sure the directory where LifeSocial.KOM is started contains a folder called "default" with the default image in it before starting the LifeSocial platform.

2.1 Test mode

Once the platform has been started, the first step is to connect to the LifeSocial.KOM network using one of the connect methods described in 1. When running the platform in a LAN (usually the case while testing), the user should first use the connect method without arguments. This will create a new ring using the localhost and the standard port (currently 9001, which can be changed on the PropertiesContainer class available in the Utilities module). The second step is to register a new user, since in test mode the ring disappears after all the clients are shut down. Repeat this procedure for a few clients to be able to retrieve profiles, add friends and user other functionalities. Notice that after registering, a user is automatically logged in and is ready to use platform services.

We can summarize the steps for starting the LifeSocial.KOM platform in test mode as:

- Check if default folder present
- Start the platform
- Connect to the network. connect (no arguments) for first client and connect <local port> for the others
- Register a new user
- Repeat the steps above for a few users
- Start using the platform

2.2 Live mode

When the platform is running over the Internet, it is assumed that the ring will persist between accesses, so there is obviously no need of registering every time. In a live environment, the user will need to change the current FreePastry settings (which is configured in test mode to be run in a LAN). In the network module, the parameter nat_search_policy=never set when connecting to the network has to be changed to nat_search_policy=prefix or simply deleted if the user has the freepastry.params file present. More information about FreePastry's parameters can be found on [Frea] and specifically about NAT settings on [Freb]. When connecting to the LifeSocial.KOM network over the Internet, the connect <localport> <bindaddress> can be used (assuming other users are using the standard port), or connect <localport> <bindport> <bindaddress> if they are

not. The bind address is the IP of a node which is already connected to the ring. After connecting to the network, the user can then login and starting using its services.

A Commands Parameters

This appendix provides more information about parameters that can be used with certain commands for the different Plug-ins. Please refer to chapter 1 for an explanation of the different commands and examples of use.

A.1 Profile Plugin

The fields currently available in the User Profile of the LifeSocial platform are listed in Table A.1. Mandatory fields when registering are: username, email, city and country. The other fields can be changed using the editprofilefield command. Currently no plausibility test is done on the values entered (for example, checking if the country actually exits or allowing only 2 options for the gender).

Profile Fields			
Field name	Type	Description	Example
ProfileUsrname	String	The username. This field cannot be changed.	X (can only be chosen when registering)
ProfileEmail	String	The email of the user. Can only be changed through the Login Plugin	X (can only be changed through the Login Plugin)
ProfileCountry	String	The country of the user	editprofilefield ProfileCountry Canada
ProfileCity	String	The city the user lives in	editprofilefield ProfileCity Montreal
ProfileUsrGender	String	The gender of the user	editprofilefield ProfileUsrGender female
ProfileFirstLastName	String	First and last name of the user	editprofilefield ProfileFirstLastName Alice Smith
ProfileDateOfBirth	String	The date of birth	editprofilefield ProfileDateOfBirth 15.04.1982
ProfileRelationshipStatus	String	The current marital status of the user	editprofilefield ProfileRelationshipStatus in a relationship
ProfileTelephonNumber	String	User's phone number	editprofilefield ProfileTelephonNumber +49 151 869 52367
ProfileInterests	String	interests of the user. Allows more than one value. (to be retrieved using the search plugin)	editprofilefield ProfileInterests Soccer, music, sailing

Table A.1: Profile Fields

B Status Messages

This appendix provides information about the status messages returned when certain commands are executed in the LifeSocial Platform. Please refer to chapter 1 for an explanation of the different commands and examples of use.

B.1 Network Plugin

When connecting and disconnecting to and from the LifeSocial Platform, there are several status messages which can be returned to the user according to the result of the operation. The related status messages can be seen on Table B.1.

Network Status Messages				
Command	Status Name	Message Text	Description	Explanation
connect	ConnectionAlreadyConnectedMsg	"You are already connected!"	user is already connected	no need to reconnect. User can start using services.
	ConnectionSuccessMsg	"Welcome! You were successfully connected to the network."	Connection to the LifeSocial Platform succeeded.	user is connected to the platform and can use its services
	ConnectionFailedMsg	"Join to the LifeSocial network failed. Please try again."	There was a problem connecting to the LifeSocial Network	Check Network settings\Error connecting to the network. Try again.
connectp	ConnectionAlreadyConnectedMsg	"You are already connected!"	user is already connected	no need to reconnect. User can start using services.
	ConnectionSuccessMsg	"Welcome! You were successfully connected to the network."	Connection to the LifeSocial Platform succeeded.	user is connected to the platform and can use its services
	ConnectionFailedMsg	"Join to the LifeSocial network failed. Please try again."	There was a problem connecting to the LifeSocial Network	Check Network settings, bind port number and address\ Error connecting to the network. Try again.
disconnect	DisconnectionSuccessMsg	"You were successfully disconnected from the network. See you again soon."	user was successfully disconnected	Disconnection success

Continued on next page

Table B.1 – continued from previous page

Network Status Messages				
Command	Status Name	Message Text	Description	Explanation
	DisconnectionFailedMsg	"Failure disconnecting from the LifeSocial network failed. Please try again."	Could not disconnect from the platform.	Check connection settings

Table B.1: Network Status Messages

Login Plugin Status Messages				
Command	Status Name	Message Text	Description	Explanation
LOGIN	ConnectionNotConnectedMsg	"You are not connected!"	user must be connected to the LifeSocial Framweork to use any other services	Connect to the LifeSocial Network first
COMING SOON!				

Table B.2: Login Plugin Status Messages

Bibliography

[Fou03] Eclipse Foundation. Equinox website, 2003. Available from: http://www.eclipse.org/equinox/ [cited 2008.10.18]. One citation in page 9.

[Frea] FreePastry. Adjusting freepastry parameters. Available from: http://freepastry.org/FreePastry/tutorial/params.html [cited 2008.10.20]. One citation in page 9.

[Freb] FreePastry. Freepastry nat/firewall configuration. Available from: http://freepastry.org/FreePastry/tutorial/nat.html [cited 2008.10.20]. One citation in page 9.