An API for Developing Mobile Ad-hoc Networking Applications Using a Public and Private Screen-based Programming Model
Technical Manual

A Technical Manual
Presented to
the Faculty of the College of Computer Studies
De La Salle University

In Partial Fulfillment
of the Requirements for the Degree of
Bachelor of Science in Computer Science

by
LARON, Andrew V.
LIM, Sharmaine Amanda S.
SYSON, Oliver Brian C.
XU, Peigen

SEE, Solomon
Faculty Adviser

March 31, 2015

Table of Contents

1.0. System Information

The P/PS library facilitates developing networked applications on Android devices, with functions that enable developers to create applications that allow for public and private screen assignment, session handling, and the use of a distributed event model which abstracts inter-device messaging from messages being sent into events being triggered.

1.1. System Dependencies

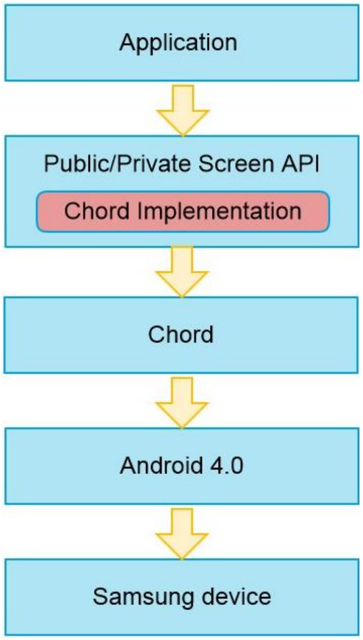
The current implementation of the P/PS library makes use of the Samsung Chord API 2.0. Therefore, all applications developed using this library must be run on a WiFi-capable Samsung smartphone or tablet running Android Ice Cream Sandwich (4.0, API Level 14) or higher, in order to function properly.

The documentation and other relevant information for Chord is available from Samsung at http://developer.samsung.com/resources/chord.

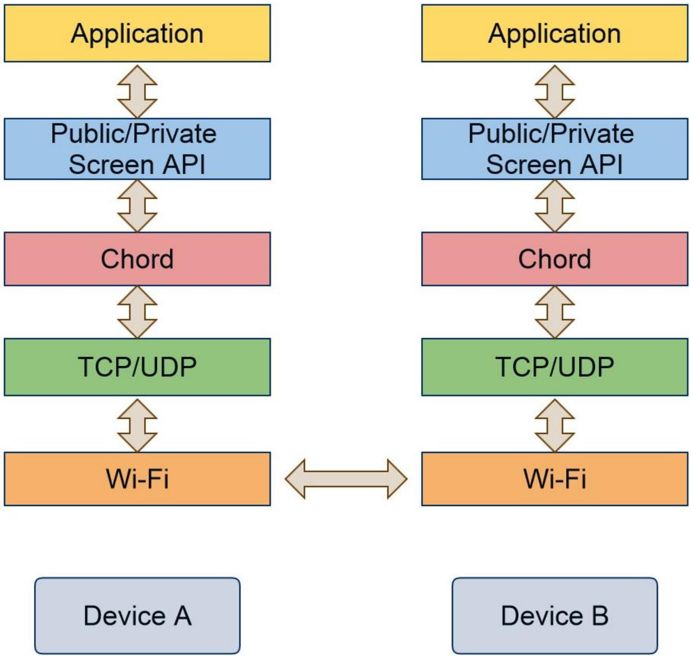
2.0. System Architecture

The Public/Private Screen library has two main components: a module consisting of the Public/Private Screen based API classes, and a module which handles interactions with the underlying Chord API.

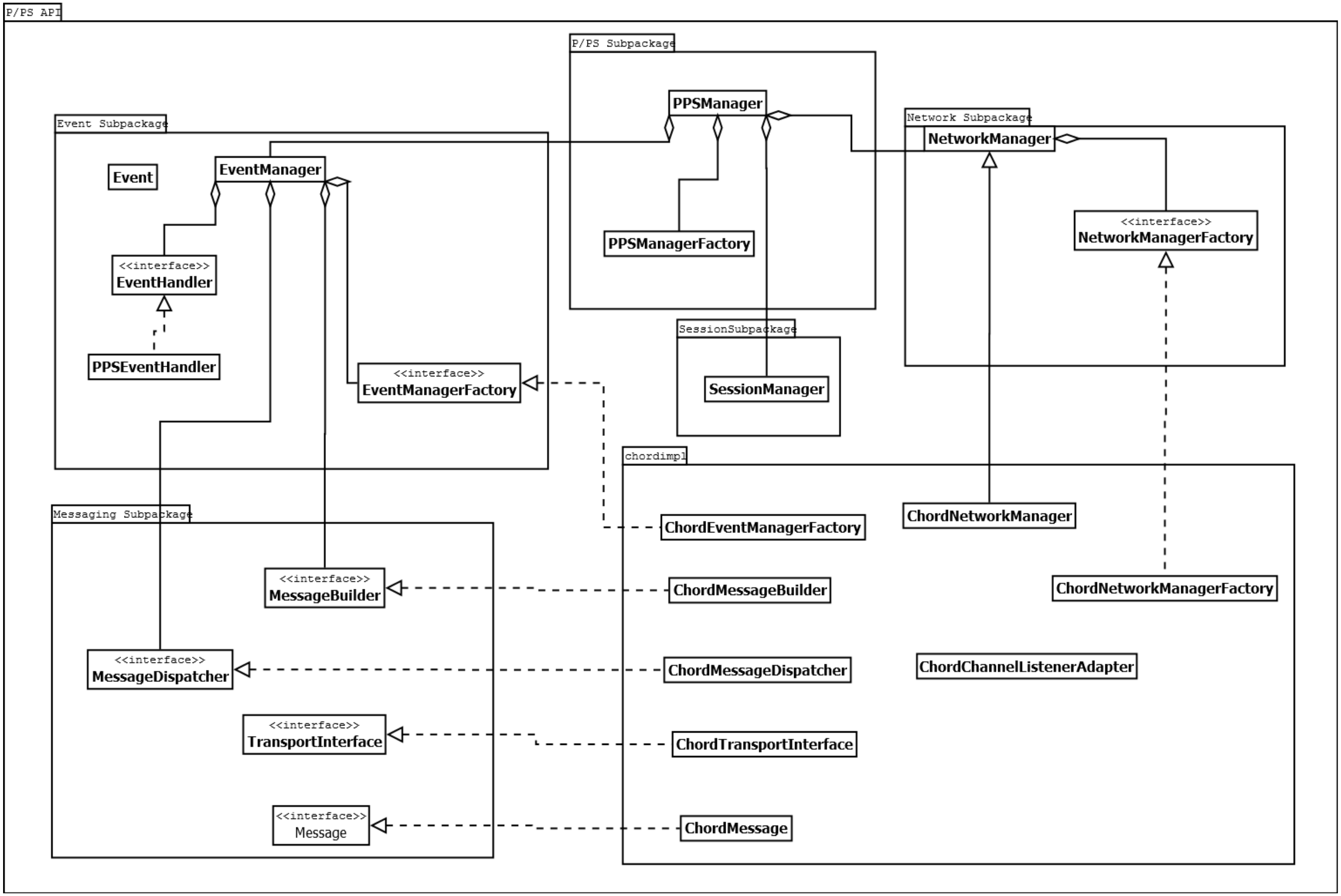
Applications built using the P/PS API only directly call the classes in the Public/Private Screen module, but the Chord interaction module handles converting the methods called by the application into something understandable by Chord.



The basic architecture of the P/PS library.



The basic communication architecture of the P/PS library.



The class diagram of the P/PS library. Details of fields and methods are listed in the next section.

The succeeding subsections contain the details of the classes and interfaces in each module and package the P/PS library. Interfaces are denoted by italics.

2.1. P/PS Module

This subsection contain the packages and interfaces of the P/PS module.

2.1.1. P/PS Package (com.llsx.pps)

Class Name	Superclass	Implemented Interface	Description
PpsManager	none	none	Manages public and private screens.
PpsManagerFactory	none	none	Used to create a PpsManager.

2.1.1.1. PpsManager

Variable Name	Data Type	Description
APP_MODE	boolean	Constant representing the mode of PpsManager in which events will be routed through the custom event handler before being processed by the PpsEventHandler.
PRIVATE	boolean	Constant indicating that a device is acting as a private screen.
PUBLIC	boolean	Constant indicating that a device is acting as a public screen.
SESSION_MODE	boolean	Constant representing the mode of PpsManager in which events will be handled by the PpsEventHandler.

Method Name	Parameters	Return Type	Description
addPrivateScreen	deviceId (String), deviceName (String)	void	Adds the device into the list of private screens as a map entry containing the device’s preset device ID and a developer- or user-specified device name.
addPublicScreen	deviceId (String), deviceName (String)	void	Adds the device into the list of public screens as a map entry containing the device’s preset device ID and a developer- or user-specified device name.
clearSessionList	none	void	Clears/Empties the list of sessions.
getContext	none	Context	Returns the current Context of PpsManager. (A Context gives 'context' or more information about the current state of the application or object.)

getDeviceName	none	String	Returns the name of the device.
getEventManager	none	EventManager	Returns the EventManager which takes care of which event handlers will be in effect to catch events for that specific context.
getInstance	none	PpsManager	Returns the current instance of PpsManager.
getNetworkManager	none	NetworkManage r	Returns the current NetworkManager, which takes care of the link between the app and the available network connections.
getPrivateScreenList	none	List<String>	Returns the list of private screen device IDs visible to the PpsManager.
getPrivateScreenNa meList	none	List<String>	Returns the list of private screen device names visible to the PpsManager.
getPublicScreenList	none	List<String>	Returns the list of public screen device IDs visible to the PpsManager.
getPublicScreenNam eList	none	List<String>	Returns the list of public screen device names visible to the PpsManager.
isPrivate	none	boolean	Returns true if the local device is acting as a private screen, or false if it is acting as a public screen.
logScreenType	screenType (boolean)	void	Logs the current screen type of the given device in the Android log.
removeFromPrivateS creen	deviceId (String)	void	Remove the given device from the list of private screens.
removeFromPublicSc reen	deviceId (String)	void	Remove the given device from the list of public screens.
setContext	context (Context)	void	Sets the application Context of PpsManager as the given Context.
setScreenMode	sessionMode (boolean)	void	Sets the screen mode as Session Mode or App Mode.
setScreenType	screenType (boolean)	void	Sets the current device as either public or private.
start	none	void	Starts the network manager and connects the app to the network.

stop	none	void	Stops the network manager and disconnects the app from the network.

2.1.1.2. PpsManagerFactory

Method Name	Parameters	Return Type	Description
createPpsManager	none	PpsManager	Returns a new PpsManager object.

2.1.2. Event Package (com.llsx.pps.event)

Class Name	Superclass	Implemented Interface	Description
Event	none	none	Represents a change in application state, triggered on certain screens in a session
EventManager	none	none	Manages processing and distribution of Events
PpsEventHandler	none	EventHandler	Used to handle common (non-application-specific) public/private screen events
EventHandler	none	none	Performs actions based on a received Event and its type
EventManagerFactory	none	none	Used to create an EventManager with certain MessageBuilder and MessageDispatcher implementations.

2.1.2.1. Event

Variable Name	Data Type	Description
APP_EVENT	boolean	Constant used to indicate that an event is application-specific (fired from the application)
PPS_EVENT	boolean	Constant used to indicate that an event is a common event (automatically fired by the PPS API)
R_ALL_SCREENS	String	Recipient constant indicating that an event will be triggered on all screens
R_LOCAL_SCREEN	String	Recipient constant indicating that an event will be triggered only on the local device
R_PERSONAL_SCREENS	String	Recipient constant indicating that an event will be triggered on all personal screens

R_PUBLIC_SCREENS	String	Recipient constant indicating that an event will be triggered on all public screens
T_ADD_NEW_SESSION	int	PPS Event Type constant indicating that a device has created a new session
T_JOIN_NETWORK	int	PPS Event Type constant indicating that a device has connected to the local network
T_JOIN_SESSION	int	PPS Event Type constant indicating that a device has joined a session
T_LEAVE_SESSION	int	PPS Event Type constant indicating that a device has left a session
T_LOCK_SESSION	int	PPS Event Type constant indicating that a device has locked a session
T_REQUEST_SESSIONS	int	PPS Event Type constant indicating that a device has requested an updated list of sessions
T_RESPOND_REQUEST_SESSIONS	int	PPS Event Type constant indicating that a device has responded to a request for sessions with a list of available sessions
T_SCREEN_TYPE_CHANGED	int	PPS Event Type constant indicating that a device has changed its screen type
T_SEND_CURRENT_STATE	int	PPS Event Type constant indicating that a device has sent its current state
T_UNLOCK_SESSION	int	PPS Event Type constant indicating that a device has unlocked a session
T_USER_JOIN_PRIVATE	int	PPS Event Type constant representing a private screen joining a session
T_USER_JOIN_PUBLIC	int	PPS Event Type constant representing a public screen joining a session
T_USER_LEFT_PRIVATE	int	PPS Event Type constant representing a private screen leaving a session
T_USER_LEFT_PUBLIC	int	PPS Event Type constant representing a public screen leaving a session

Method Name	Parameters	Return Type	Description
getType	none	int	Returns this Event's type, which determines how it will be handled

getPayload	none	java.io.Serializable	Returns the payload (contents) of this Event
getRecipient	none	String	Returns this Event's recipient type
getSession	none	String	Get the session name of the session where this Event was triggered
isPpsEvent	none	boolean	Returns true if this event was triggered by the PPS API itself, false if this event was triggered from the application level
setPpsEvent	isPpsEvent (boolean)	void	Sets whether this Event will be handled as an application-triggered Event (isPpsEvent==false) or a PPS API-triggered Event (isPpsEvent==true)

2.1.2.2. EventManager

Variable Name	Data Type	Description
messageBuilder	MessageBuilder	MessageBuilder used by the EventManager
messageDispatcher	MessageDispatcher	MessageDispatcher used by the EventManager

Method Name	Parameters	Return Type	Description
getInstance	none	EventManager	Used to get the current (singleton) instance of EventManager
setDefaultFactory	factory (EventManager Factory)	void	Sets the type of EventManagerFactory to be used, and thus the type of EventManager that will be created
sendEvent	event (Event)	void	Converts an Event into a Message and sends it to its recipients using the current session/channel
sendEventOnDefault Channel	event (Event)	void	Converts an Event into a Message and sends it to its recipients using the default session/channel
unpackEvent	message (Message)	void	Converts a received Message into an Event and applies it
setEventHandler	handler (EventHandler)	void	Sets the application event handler of the EventManager
triggerEvent	event (Event)	void	Sends an event to all recipients and/or applies it on the local device

applyEvent	event Event	void	Applies an Event to the local device using the appropriate EventHandler

2.1.2.3. PpsEventHandler

Method Name	Parameters	Return Type	Description
handleEvent	event (Event)	void	Handles a PPS event.

2.1.2.4. *EventHandler*

Method Name	Parameters	Return Type	Description
handleEvent	event (Event)	void	Handles an event as specified by the implementing class.

2.1.2.5. *EventManagerFactory*

Method Name	Parameters	Return Type	Description
createEventManager	none	EventManager	Returns an instance of EventManager with network layer-specific MessageBuilder and MessageDispatcher

2.1.3. Messaging Package (com.llsx.pps.messaging)

Class Name	Superclass	Implemented Interface	Description
Message	none	java.io.Serializable	Represents the actual message being sent between devices.
MessageBuilder	none	none	Handles the building of Messages from Events, and vice versa.
MessageDispatcher	none	none	Handles sending and receiving messages.
TransportInterface	none	none	Interfaces with the Chord API to send data over the network.

2.1.3.1. Message

Method Name	Parameters	Return Type	Description
getType	none	int	Returns the type of message corresponding to the type of event.

getContents	none	java.io.Serializable	Returns a Serializable containing the actual contents of the message.
getRecipient	none	String	Returns the intended recipient of the message.
isPpsEvent	none	boolean	Returns true if the message is for PPS API Events; false if the message is for Application Events

2.1.3.2. *MessageBuilder*

Method Name	Parameters	Return Type	Description
buildMessage	event (Event)	Message	Builds and returns the message based from the given event.
unpackEvent	message (Message)	Event	Builds and return the event based from the given message.

2.1.3.3. *MessageDispatcher*

Method Name	Parameters	Return Type	Description
receiveMessage	message (Message)	void	Triggers when the device receives a message.
sendMessage	message (Message), isCustomChannel (boolean)	void	Sends out the message through the transport interface. If isCustomChannel is true, it sends the message over the current custom session. Otherwise the message is sent over the default channel/session.
setTransportInterface	transportInterface (TransportInterface)	void	Sets the transport interface that will be used for transferring messages.

2.1.3.4. *TransportInterface*

Method Name	Parameters	Return Type	Description
send	recipient (String), message (Message), isCustomChannel (boolean)	void	Sends the given Message to the device specified by “recipient”. .If isCustomChannel is true, it sends the message over the current custom session. Otherwise the message is sent over the default channel/session.

sendToAll	message (Message), isCustomChannel (boolean)	void	Sends the given Message to all devices in the current session.If isCustomChannel is true, it sends the message over the current custom session. Otherwise the message is sent over the default channel/session.
setMessageDispatcher	dispatcher (MessageDispatcher)	void	Sets the message dispatcher to be used by the TransportInterface.
joinNetworkSession	none	void	Join the device to the network session. In this case, is the SchordChannel.

2.1.4. Network Package (com.llsx.pps.network)

Class Name	Superclass	Implemented Interface	Description
NetworkManager	none	none	Manages the link between the app and the available network connections.
NetworkManagerFactory	none	none	Used to create a NetworkManager.

2.1.4.1. NetworkManager

Method Name	Parameters	Return Type	Description
getInstance	none	NetworkManager	Returns the current singleton instance of NetworkManager, and initializes it if it has not been created yet.
setDefaultFactory	factory (NetworkManager Factory)	void	Sets the given factory as the new default factory for creating the singleton instance of NetworkManager.

2.1.4.2. *NetworkManagerFactory*

Method Name	Parameters	Return Type	Description
createNetworkManager	none	NetworkManager	Creates, initializes and returns a new NetworkManager.

2.1.5. Session Package (com.llsx.pps.session)

Class Name	Superclass	Implemented Interface	Description
SessionManager	none	none	Manages session handling, including tracking other devices in the current session.

2.1.5.1. SessionManager

Variable Name	Data Type	Description
DEFAULT_SESSION	String	Name of the default session (constant).
LOCK	boolean	Constant representing a session state of “locked”.
UNLOCK	boolean	Constant representing a session state of “unlocked”.

Method Name	Parameters	Return Type	Description
addAvailableSession	sessionID (String), isLock (boolean)	void	Adds the newly created or newly received session to the list of sessions.
addDevice	deviceId (String), deviceName (String)	void	Register a new device given its unique identifier (device ID) and its name representation (e.g. "Player 1").
clearAvailable SessionsMap	none	void	Clears/Empties the list of available sessions
clearDeviceMap	none	void	Clears/Empties the list of accessible devices
createSession	sessionName (String)	String	Creates a new session given a unique name for the session.
joinSession	sessionName (String)	void	Joins the device to the specified session name
getAvailableSessions	none	java.util.Set <String>	Returns a set of all names of sessions available on the network.
getAvailable SessionsMap	none	java.util.Map <String, Boolean>	Returns a map of all names of sessions available on the network and their corresponding lock status (true if locked or false if unlocked).
getSessionToJoin	none	String	Returns the currently selected session,

			which may not be the currently active session.
getDeviceId	deviceName (String)	String	Gets the identifier (ID) of the target device given the device's name representation.
getDeviceName	deviceId (String)	String	Gets the name representation of the device given the device ID.
getInstance	none	SessionManager	Returns the current singleton instance of SessionManager.
getOwnDeviceName	none	String	Returns the name of the local device.
isSessionLocked	sessionId (String)	boolean	Returns true if the given session is locked; false otherwise.
loadSavedSessionId	none	void	Loads the last session ID and its status (locked/unlocked) from the device's memory.
lockSession	sessionId (String)	void	Locks the given session so that no one new can join.
removeAvailable Session	sessionId (String)	void	Removes the session with the specified session ID from the session list.
removeDevice	deviceId (String)	void	Removes the device with the specified device identifier from the list of devices.
requestSessions	none	void	Sends an event to all devices requesting for all the known sessions.
saveDefaultSessionId	none	void	Saves the ID of the default session into the device's memory as the most recently joined session.
saveSessionId	none	void	Saves the current session ID and its status (locked/unlocked) into device's memory.
setSessionToJoin	sessionId (String)	void	Sets the given session as the current session.
setDefaultSession	none	void	Sets the default session as the current session.
setDeviceName	deviceName (String)	void	Sets the current device's name representation (e.g. "Player 1") as the given string.

setSessionMode	sessionMode (boolean)	void	Sets the given session mode (default or custom) as the current session mode.
unlockSession	sessionId (String)	void	Unlocks the given session so that other people can join.
getCurrentSessionName	none	String	Returns the name of the current session.

2.2. Internal Chord Implementation Module

This subsection contains the packages and classes of the internal Chord Implementation module, which is an implementation of the communication module of the P/PS library which makes use of the Samsung Chord API. Inherited methods are not listed below unless they have been overridden.

2.2.1. Chord Implementation Package (com.llsx.pps.internal.chord)

Class Name	Superclass	Implemented Interface	Description
ChordChannelListenerAdapter	none	com.samsung.android.sdk.chord.SchordChannel.StatusListener	An implementation of SchordChannel.StatusListener. Purpose is simply to eliminate the need to override all of these methods in the ChordTransportInterface.
ChordEventManagerFactory	none	com.llsx.pps.event.EventManagerFactory	Concrete implementation of EventManagerFactory for Samsung Chord API
ChordMessage	com.llsx.pps.messaging.Message	java.io.Serializable	Concrete implementation of Message using the Samsung Chord API. Implements Serializable to allow the message to contain Objects.
ChordMessageBuilder	none	com.llsx.pps.messaging.MessageBuilder	Concrete implementation of MessageBuilder using the Samsung Chord API
ChordMessageDispatcher	none	com.llsx.pps.messaging.MessageDispatcher	Concrete implementation of MessageDispatcher using the Samsung Chord API
ChordNetworkManager	com.llsx.pps.network.NetworkManager	none	Concrete implementation of NetworkManager using Samsung Chord API
ChordNetworkManagerFactory	none	com.llsx.pps.network.NetworkManager	Concrete implementation of NetworkManagerFactory

		Factory	using Samsung Chord API
ChordTransport Interface	none	com.llsx.pps. messaging.Transport Interface	Concrete implementation of TransportInterface using the Samsung Chord API.
ChordTransport Interface.SPSChord ChannelListener Adapter	none	com.samsung.android. sdk.chord.Schord Channel.StatusListener	Internal class implementing a Chord channel status listener, which listens for events on a specific channel.

2.2.1.1. ChordChannelListenerAdapter

This class is merely a skeleton implementation of com.samsung.android.sdk.chord.Schord-Channel.StatusListener, used to remove the need for its subclass, ChordTransportInterface.SPS-ChordChannelListenerAdapter, to override the numerous methods in the interface which are not needed by the P/PS library.

2.2.1.2. ChordEventManagerFactory

Method Name	Parameters	Return Type	Description
createEventManager	None	EventManager	Returns an event manager which can be used by the ChordTransportInterface.

2.2.1.3. ChordMessage

Variable Name	Data Type	Description
serialVersionUID	long	Chord build number

Method Name	Parameters	Return Type	Description
getBytes	message (ChordMessage)	Byte[ ]	Returns the ChordMessage, converted into a byte array for sending over the network.
obtainChordMessage	Data (byte [])	ChordMessage	Recreates a ChordMessage object from the given byte array.

2.2.1.4. ChordMessageBuilder

Method Name	Parameters	Return Type	Description
buildMessage	event (Event)	Message	Implements the method of the same name in MessageBuilder. Returns a ChordMessage corresponding to the given Event.
unpackEvent	Message (Message)	Event	Implements the method of the same name in MessageBuilder. Returns an Event corresponding to the given Message.

2.2.1.5. ChordMessageDispatcher

Variable Name	Data Type	Description
transportInterface	TransportInterface	TransportInterface to be interacted with by the ChordMessageDispatcher

Method Name	Parameters	Return Type	Description
sendMessage	isCustomChannel (boolean), message (Message)	void	Sends the specified message through the TransportInterface. If "isCustomChannel" is true, message is sent over a custom channel; otherwise it is sent over the default channel.
receiveMessage	message (Message)	void	Triggers when the device receives a message.
setTransportInterface	transportInterface (TransportInterface)	void	Sets the transport interface that will be used by the ChordMessageDispatcher for transferring messages. In this case, a ChordTransportInterface is expected.

2.2.1.6. ChordNetworkManager

Variable Name	Data Type	Description
chord	com.samsung.android.sdk.chord.Schord	Chord class required to use Chord API
mChordManager	com.samsung.android.sdk.chord.SchordManager	Chord manager provided by Chord API
mChordManagerListener	com.samsung.android.sdk.chord.SchordManager.Status Listener	Listener for the SchordManager

Method Name	Parameters	Return Type	Description
getChordManager	none	com.samsung.android.sdk.chord.Schord Manager	Returns the currently active SchordManager.
initializeChord	none	void	Initialises Chord and checks whether the device supports Chord.
initializeChordManager	none	void	Initialises the SchordManager on the first available interface type.

2.2.1.7. ChordNetworkManagerFactory

Method Name	Parameters	Return Type	Description
createNetworkManager	none	NetworkManager	Creates and initializes a new ChordNetworkManager.

2.2.1.8. ChordTransportInterface

Variable Name	Data Type	Description
channelName	String	Name of the default channel
defaultChannel	com.samsung.android.sdk.chord.SchordChannel	The default channel/session
defaultChannel Listener	com.samsung.android.sdk.chord.SchordChannel.StatusListener	Status listener for the default session
mChannel	com.samsung.android.sdk.chord.SchordChannel	Currently joined custom channel/session

mChordChannel Listener	com.samsung.android.sdk. chord.SchordChannel. StatusListener	Status listener for the current custom session
messageDispatcher	MessageDispatcher	The MessageDispatcher to be used
PAYLOAD_TYPE	String	Payload type, used to identify messages originating from the P/PS API

Method Name	**Parameters**	**Return Type**	**Description**
joinDefaultChannel	none	void	Join the default channel/session
joinCustomChannel	none	void	Gets the custom session currently set from the SessionManager, then joins it
sendToAll	message (Message), isCustomChannel (boolean)	void	Sends the specified message to all devices in the same channel. If isCustomChannel is true, the message is sent over the current custom channel; otherwise it is sent over the default channel.
setMessage Dispatcher	dispatcher (MessageDispatcher)	void	Sets the MessageDispatcher to be used. A ChordMessageDispatcher is expected in this case.
onMessageReceived	receivedMessage (Message)	void	Called when a message is received over the network
send	userToSend (String), message (Message), isCustomChannel (boolean)	void	Sends the given message to the device with name specified as userToSend. If isCustomChannel is true, the message is sent over the current custom channel; otherwise it is sent over the default channel.
joinNetworkSession	none	void	Joins the device to the custom network channel.

2.2.1.9. ChordTransportInterface.SPSChordChannelListenerAdapter

Method Name	Parameters	Return Type	Description
onDataReceived	FromNode (String), fromChannel (String), payloadType (String), payload (byte [] [])	void	Called when the specified payload (byte array) is received over the network, sent by fromNode over fromChannel. The parameter “payloadType” is used to determine from which application the received data originates.
onNodeJoined	fromNode (String), fromChannel (String)	void	Called on all devices in the channel when a device (fromNode) joins a channel (fromChannel)
onNodeLeft	fromNode (String), fromChannel (String)	void	Called on all devices in the channel when a device (fromNode) leaves a channel (fromChannel)
getNodeAlias	none	String []	Returns an array of Strings, with device ID at index 0 and device name at index 1.