Meshee Developer Guide (V0.8.5)

Meshee Networking Technology Ltd.

Contents

1 Meshee Overview

Meshee is communications middleware that can cross operating systems. Meshee can be applied in the following scenarios: 1) Large-coverage high-speed ad-hoc self-organizing networking scenarios; 2) scnearios where cellular data plan is expensive; 3) scenarios where cellular (e.g., 4G) or WiFi networks are not well covered and 4) scenarios where local communications and Internet communications needs collaboration.

When terminals are distributed over a large area, Meshee technology supports them to form a Meshee network. Without using cellular traffic, any two terminals can communicate with each other at high speed through the Meshee network. Communication in Meshee networks does not pass through Internet servers and therefore has privacy advantage naturally.

- Meshee API is now applicable to Android (>= Android 21), IOS, linux, Windows and Linux version are in development.
- Meshee API is based on self-developed Meshee networking and communications protocol, providing local Meshee networking service. The service works in the following two phases.
  - Pre-communication phase : Create, discover and join Meshee networks before communication.
  - In-communication phase: Exchange data, e.g., text, image, file and live stream in a formed Meshee network.

- Meshee API privides all sorts of communications services, including Contact, Conversation, Message (e.g., text, image, customized type, File and live stream). Leveraging the aforementioned services, developers can develop local communication applications easily, such as:
  - Live sharing - Sharing live stream from the Internet in Meshee networks. Currently http-flv format is supported, and the other formats are in development.
  - File sharing - Easy transferring large-size file in Meshee networks.
  - Goup game - Use customized message to develop all sorts of messaging for multi-people games.
  - Social chat/collabtive work - Combine text, image and file messaging to achieve social chat/collabtive work.

2 Meshee features

Meshee has the following features.

- Cross-OS platforms - Meshee is software middleware, which can provide cross-platform communication services.
- Large network coverage - The underlying transferring layer of Meshee is WiFi, which makes network coverage as large as 400 meters.
- Large number or terminals in a Meshee network - a single Meshee network can connect ~100 terminals.

- Large bandwidth - Communication bandwidth between any two terminals in a Meshee network can reach ~100Mbps.
- Simulataneous use of Meshee network and cellular network - A terminal can use both Meshee networks and cellular network at the same time.
- Multi-path transfering - Meshee networking and communications protocol can dynamically allocate bandwidth for multi-path transferring with load-balancing and fairness.
- Resilient access - While the Internet vai 4G/WiFi is available, terminals can use negeliable Internet traffic to obtain access information of Meshee networks, thereby improving accessing capability of Meshee networks significantly.
- Communications format extension - Aiming different application requirements, Meshee supports fast extension of all sorts of multimedia communications format.

3 Meshee API

Meshee API includes:

- Meshee SDK management API for initializing SDK, exiting SDK、getting SDK service and SDK Account management
- Setting Service API for getting and configuring basic SDK setting
- Network Service API for Meshee network related operation

- Contact Service API for Contact related operation
- Conversation Service API for Conversationrelated operation
- Message Service API for Message related operation
- File Service API for File related operation
- Live Service API for Live stream sharing related operation
- Avatar Service API for Avatar related operation

3.1 Meshee SDK Management API

Meshee SDK management API is used for initializing SDK, exiting SDK、getting SDK service and SDK Account management.

- Meshee SDK management API package: cn.meshee.fclib.api
- Meshee SDK management is done by FCClient

3.1.1 FCClient

FCClient’ s Meshee SDK management API includes:

static void	init(Application application, SdkOptions sdkOptions)
static void	init(Application application)
static <T> T	getService(Class<T> clazz)
static void	bindAccount(Contact contact)
static void	bindAccount()
static void	unbindAccount()
static boolean	isAccountBound()
static void	exit()

3.1.1.1 SDK Initialization

- API: public static void init(Application application, SdkOptions sdkOptions)

  (Available since Meshee SDK 0.8.0)

Parameters	
application	Application: The application that uses Meshee SDK.
sdkOptions	SdkOptions: The sdkOptions for initializing SDK.

Before a terminal can use Meshee SDK, SDK must be initialized in the main thread of App. SdkOptions for initialization can customize SdkStorageRootPath and NetworkStrategyType.

SdkStorageRootPath is used for saving content while SDK is running.

NetworkStrategyType is the network type that Meshee SDK is to support - NetworkStrategyType1 or NetworkStrategyType2. NetworkStrategyType1 is a Meshee network with a diameter of 400 meters covering ~100 terminals; NetworkStrategyType2 is a Meshee network with a dimeter of 200 meters covering ~10 terminals. NetworkStrategyType1 has the advantage of larger network coverage, more terminals and more distant communication capability. NetworkStrategyType2 has higher P2P transferring speed and smaller transferring delay in average.

- API: public static void init(Application application)

  (Available since Meshee SDK 0.8.0)

Parameters	
Application	Application: The application that uses Meshee SDK.

Before a terminal can use Meshee SDK, SDK must be initialized in the main

thread of App. When SdkOptions is not designated, the SdkOptions for initialization is defaulted as SdkOptions.DEFAULT. For the default SdkOptions.DEFAULT, SdkStorageRootPath corresponds to the directory named after App PackageName in the root category, and NetworkStrategyType is NetworkStrategyType1.

3.1.1.2 Get SDK Service

- API: public static <T> T getService(Class<T> clazz)

  (Available since Meshee SDK 0.8.0)

Parameters	
clazz	Class<T>: The class of a Meshee service.

Returns	
<T> T	Returns the instance of the class queried.

App can obtain Meshee service by calling the API. The API must be used after initializing SDK. Meshee service includes Setting Service, Network Service, Contact Service, Conversation Service, Message Service, File Service, Live Service and Avatar Service. App can get a certain service by calling the API and then furtuer calls the service' s API. For example:

```
//First get NetworkService
NetworkService networkService = FCClient.getService(NetworkService.class);
//Then call scan() in NetworkService
FCClient.getService(NetworkService.class).scan();
```

3.1.1.3 SDK Account Management

- API: public static void bindAccount(Contact contact)

(Available since Meshee SDK 0.8.0)

Parameters	
contact	Contact: The Contact to be bound to Meshee SDK as the Account.

Bind the Contact to Meshee SDK as the Account. The API must be used after initializing SDK. Until unbinding the Account or exiting SDK, the Contact is used as the only Account.

- API: public static void bindAccount()

  (Available since Meshee SDK 0.8.0)

  Bind the default Contact to Meshee SDK as the Account. The API must be used after initializing SDK. The default Account is generated by Meshee SDK for App. Until unbinding the default Account or exiting SDK, the Contact is used as the only Account.

- API: public static void unbindAccount ()

  (Available since Meshee SDK 0.8.0)

  Unbind the Account that is already bound to Meshee SDK. The API must be used after SDK has bound the Account. Unbinding operation will not exit SDK directly, and SDK only returns to the state in which the Account is not bound.

- API: public static boolean isAccountBound()

  (Available since Meshee SDK 0.8.0)

Returns	
boolean	Returns whether an Account is already bound.

Get whether an Account is bound to Meshee SDK.

3.1.1.4 Exit SDK

- API: public static void exit()

  (Available since Meshee SDK 0.8.0)

  When exiting SDK, App is terminated as well.

3.2 Setting Service API

Setting Service API is used for getting and configuring basic SDK setting. Setting Service must be used after SDK is initialized.

- Setting Service API package: cn.meshee.fclib.api.setting
- Setting Service includes SettingService, which is used for active setting opration.

3.2.1 SettingService

SettingService includes the following APIs:

String	getSdkStorageRootPath()
NetworkStrategyType	getNetworkStrategyType()
void	saveAccount(Contact contact)
List<Contact>	getAllAccounts()
int	getProxyHttpServerPort()

3.2.1.1 Get SDK Setting

- API: String getSdkStorageRootPath()

  (Available since Meshee SDK 0.8.0)

Returns

String	Returns the configured SdkStorageRootPath in SDK initialization.

Get SdkStorageRootPath. If sdkOptions for SDK initialization configures sdkStorageRootPath, the returned result is the configured sdkStorageRootPath. If sdkOptions is not designated for SDK initialization, the returned result is the default sdkStorageRootPath.

- API: NetworkStrategyType getNetworkStrategyType()

  (Available since Meshee SDK 0.8.0)

Returns	
NetworkStrategyType	Returns the configured NetworkStrategyType in SDK initialization.

Get NetworkStrategyType. If sdkOptions for SDK initialization configures NetworkStrategyType, the returned result is the configured NetworkStrategyType. If sdkOptions is not designated for SDK initialization, the returned result is the default NetworkStrategyType, i.e., NetworkStrategyType1.

3.2.1.2 Save and Get Account

- API: void saveAccount(Contact contact)

  (Available since Meshee SDK 0.8.0)

Parameters	
contact	Contact: The Contact to be saved as the Account.

Save the Contact as the persistent Account. Exiting Meshee SDK does not delete the Account saved by the API.

- API: List<Contact> getAllAccounts()

  (Available since Meshee SDK 0.8.0)

Returns	
List<Contact>	Returns all saved Accounts.

Get all previously saved Accounts.

3.2.1.3 Get Local Proxy Server Port

- API: int getLocalProxyServerPort()

  (Available since Meshee SDK 0.8.0)

Returns	
int	Returns the local proxy server port.

Get local http/https proxy port. If proxy server is used, proxy is configured as the local IP "127.0.0.1" and the port # obatained by calling getLocalProxyServerPort(). When the local proxy is configured, Meshee SDK will smartly choose mobile network Interface for Internet communications if WiFi Interface is being used by connection of Meshee networks.

3.3 Network Service API

Network Service API is used for Meshee network related operation.

- Network Service API package: cn.meshee.fclib.api.network
- Network Service includes NetworkService and NetworkServiceObserve. NetworkService is used for active network opration, and NetworkServiceObserve is used for receiving Meshee network related event.

3.3.1 NetworkService

NetworkService includes the following APIs:

void	createNetwork(String groupName, boolean is5GFirst)
void	scan()
void	softScan()
void	stopScan()
Role	getRole()
void	backToIdle()
void	joinNetwork(NetworkPoint networkPoint)
void	setDisconnectWhenBackground(boolean disconnetWhenBackground)
boolean	isDisconnectWhenBackground()
void	setForegroundForNonNativeActivity(boolean foreground)
boolean	isForegroundForNonNativeActivity()
void	restartAutoJoinNetwork()
void	stopAutoJoinNetwork()
boolean	isAutoJoinNetwork()
List<NetworkPoint>	getNetworkPoints()
NetworkPoint	getConnected()

3.3.1.1 About Meshee Network Role

- Role getRole()

  (Available since Meshee SDK 0.8.0)

Returns	
Role	Returns the Network Role. (Note: Idle Role: State when NOT in any Meshee network. Owner Role: State when a Meshee network is already created by itself and can act as a NetworkPoint for connection by other devices.

	Member Role: State when a Meshee network is already joined and can act as a NetworkPoint for connection by other devices.)

Look up the terminal' s Meshee Network Role. Meshee Network Roles include Idle, Owner and Member. Please see the detail in the upper table.

- API: void backToIdle()

  (Available since Meshee SDK 0.8.0)

  A terminal whose Network Role is Owner or Member can call the API to bak to Idle and disconnect from the Meshee network which the terminals is in.

3.3.1.2 Create Meshee Network

- API: void createNetwork(String groupName, boolean is5GFirst)

  (Available since Meshee SDK 0.8.0)

Parameters	
groupName	String: The group name of for the Meshee network to be created.
is5GFirst	boolean: The flag for whether 5GHz Meshee network is created with priority. (Note: Whether 5GHz network can be created successfully also depends on the device capability.)

A terminal can creat a Meshee network. Once a Meshee network is created, the terminal' s Network role becomes Owner, and the terminal becomes a NetworkPoint which can be discovered (i.e., scanned) by other terminals. When a terminal creates a Meshee network, a group is setup. The group name can be customized and groupId is generated for the group by Meshee SDK.

A terminal can choose 2.4GHz band or 5GHz WiFi band for creating a Meshee network. 2.4GHz Meshee networks have strong penetrationg capability, but 5GHz Meshee networks has higer transferring speed when there are no obstructers. From the perspective of terminal penetration rate, 2.4GHz is much higher than 5GHz. A terminal wholse Network Role is Idle, Owner or Member can call the API.

3.3.1.3 Scan Meshee Network

- API: void scan()

  (Available since Meshee SDK 0.8.0)

  Unconditionally scan nearby NetworkPoints of Meshee networks. When a terminal’ s WiFi is disabled (the terminal can be Idle Role or Owner Role), calling the API will enable WiFi and start scanning. When a terminal’ s WiFi is enabled (the terminal can be Idle Role or Member Role), calling the API will start canning immediately.

- API: void softScan()

  (Available since Meshee SDK 0.8.0)

  Conditionally scan nearbly NetworkPoints of Meshee networks. Only when the terminal’ s WiFi is enabled (the terminal can be Idle Role or Member Role), calling the API will start scanning. Otherwise, no scanning operation happens.

- API: void stopScan()

  (Available since Meshee SDK 0.8.0)

  Stop scanning nearby NetworkPoints of Meshee networks.

3.3.1.4 Joint Meshee Network

- API: void joinNetwork(NetworkPoint networkPoint)

  (Available since Meshee SDK 0.8.0)

Parameters	
networkPoint	NetworkPoint: The access point of a Meshee network to join.

Choose a NetworkPoint of a Meshee network to connect for joining the Meshee network. Once joining network is successful, the terminal' s role will become Member. A terminal wholse Network Role is Idle or Member can call the API.

3.3.1.5 Look Up Meshee Network Information

- List<NetworkPoint> getNetworkPoints()

  (Available since Meshee SDK 0.8.0)

Returns	
List<NetworkPoint>	Returns the latest scan list of NetworkPoints of Meshee networks.

Look up the NetworkPoint list of the latest scanning.

- NetworkPoint getConnected()

  (Available since Meshee SDK 0.8.0)

Returns	
NetworkPoint	Returns the currently connected NetworkPoint.

Look up the currently connected NetworkPoint of the Meshee network.

3.3.1.6 About Background Meshee Network Connection

- API: void setDisconnectWhenBackground(boolean disconnetWhenBackground)

  (Available since Meshee SDK 0.8.5)

Parameters	
disconnetWhenBackground	boolean: whether disconnected from Meshee network when App becomes background

Set whether the terminal is disconnected from Meshee network when App becomes background. The API only is effective for the terminal whose Role is Member. The terminal whose Network Role is Owner always remains connected in any case. If true, the terminal keeps connection with a Meshee network only when App is foreground; once App becomes background, the terminal whose Network Role is Member no longer keeps connection with the Meshee network. If false, APP remains connected in any case. The terminal' s connection with the Meshee network is not influenced by calling the API.

- API: boolean isDisconnectWhenBackground()

  (Available since Meshee SDK 0.8.5)

Returns	
boolean	Returns whether isDisconnectWhenBackground.

Look up the bool value of whether isDisconnectWhenBackground.

- API: void setForegroundForNonNativeActivity(boolean foreground)

  (Available since Meshee SDK 0.8.5)

Parameters	
foreground	boolean: true: explicitly tells Meshee SDK that the activity to become foreground will make App become background.

	false: explicitly tells Meshee SDK that the activity become background

The 3rd party Activity integrated by App may make App turn background. Calling setForegroundForNonNativeActivity(true) right before the Activity becomes foreground can make sure App remains foreground. Call setForegroundForNonNativeActivity(false) immediately after the Activity becomed background. For example:

```
NetworkService networkService = FCClient.getService(NetworkService.class);
//Explicitly tells Meshee SDK that the activity for choosing a file to become foreground will make App become background.
networkService.setForegroundForNonNativeActivity(true);
//Non-native 3rd party activity for choosing a file to become foreground
startActivityForResult(Intent.createChooser(intent, "Select a File to Send"), REQUEST_FILE_PICKER);

//Explicitly tells Meshee SDK that Non-native 3rd party activity for choosing a file to become background
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    switch (requestCode) {
        case REQUEST_FILE_PICKER:
        networkService.setForegroundForNonNativeActivity(false);
        if (resultCode == RESULT_OK) {
            ...
        }
    }
}
```

- API: boolean isForegroundForNonNativeActivity()

  (Available since Meshee SDK 0.8.5)

Returns	
boolean	Returns whether isForegroundForNonNativeActivity.

Look up the bool value of whether isForegroundForNonNativeActivity.

3.3.1.7 About Automatic Joining Meshee Network

- API: void restartAutoJoinNetwork()

  (Available since Meshee SDK 0.8.0)

  Restart automaticly joinning Meshee networks, i.e., re-enter autoJoinNetwork status. A terminal whose Network Role is Idle, Member of Owner can call the API. The terminal whose Network Role is Member of Owner will enter Idle Role and then try rejoining Meshee network, after calling the API. The terminal in autoJoinNetwork status wil always automatically connect NetworkPoint when its NetworkRole is Idle and WiFi is enabled.

- API: void stopAutoJoinNetwork()

  (Available since Meshee SDK 0.8.0)

  Stop automatic joining Meshee network, i.e., exit autoJoinNetwork status. A terminal whose Network Role is Idle, Member of Owner can call the API. The terminal whose Network Role is Membert or Owner will not change its Network Role after calling the API. The terminal not in autoJoinNetwork status wil not automatically connect NetworkPoint when its NetworkRole is Idle and WiFi is enabled.

- API: boolean isAutoJoinNetwork()

  (Available since Meshee SDK 0.8.0)

Returns	
boolean	Returns whether in autoJoinNetwork status.

  Look up the bool value of whether in autoJoinNetwork status.

3.3.2 NetworkServiceObserve

NetworkServiceObserve includes the following APIs:

void	observeRoleUpdate(Observer<RoleEvent> observer, boolean register)
void	observeNetworkEvent(Observer<NetworkEvent> observer, boolean register)
void	observeConnectionEvent(Observer<ConnectionState> observer, boolean register)
void	observeNetworkPointsUpdate(Observer<List<NetworkPoint>> observer, boolean register)

3.3.2.1 Monitor Meshee Network Create/Destroy Event

- API: void observeNetworkEvent(Observer<NetworkEvent> observer, boolean register)

  (Available since Meshee SDK 0.8.0)

Parameters	
observer	Observer<NetworkEvent>: The observer of creating/destroying the Meshee network.
register	boolean: The flag for registration or unregistration (true for registration and false for unregistration)

Register/Unregister Observer<NetworkEvent> for receiving Meshee network creating/destroying event. Only a terminal whose Network Role is Owner can destroy the Meshee network created by itself.

3.3.2.2 Monitor Meshee Network Connection Event

- API: void observeConnectionEvent(Observer<ConnectionState> observer, boolean register)

(Available since Meshee SDK 0.8.0)

Parameters	
observer	Observer<ConnectionState>: The observer of connection state with respect to Meshee network. (Disconnected State: device is NOT connected any NetworkPoint of Meshee network Connecting State: Device is connecting to a NetworkPoint of a Meshee network Connected State: Device is already connected to a NetworkPoint of a Meshee network)
register	boolean: The flag for registration or unregistration (true for registration and false for unregistration)

Register/Unregister Observer<ConnectionState> for receiving Meshee networks' NetworkPoint connection event.

3.3.2.3 Monitor Meshee NetworkPoint Scan Event

- API: void observeNetworkPointsUpdate(Observer<List<NetworkPoint>> observer, boolean register)

  (Available since Meshee SDK 0.8.0)

Parameters	
observer	Observer<List<NetworkPoint>>: The observer of List<NetworkPoint>.
register	boolean: The flag for registration or unregistration (true for registration and false for unregistration)

Register/Unregister Observer<List<NetworkPoint>> for receiving latest scan list of NetworkPoint.

3.3.2.4 Monitor Meshee Network Role Event

- API: void observeRoleUpdate(Observer<RoleEvent> observer, boolean

register)

(Available since Meshee SDK 0.8.0)

Parameters	
observer	Observer<RoleEvent>: The observer of RoleEvent. (Note: Idle Role: State when NOT in any Meshee network. Device in this role can create network, scan network and join network; Owner Role: State when a Meshee network is already created by itself and can act as a NetworkPoint for connection by other devices. Member Role: State when a Meshee network is already joined and can act as a NetworkPoint for connection by other devices.)
register	boolean: The flag for registration or unregistration (true for registration and false for unregistration)

Register/Unregister Observer<RoleEvent> for receiving Network Role Change event.

3.4 Contact Service API

Contact Service API is used for Contact related operation.

- Contact Service API package: cn.meshee.fclib.api.contact
- Contact Service includes ContactService and ContactServiceObserve. ContactService is used for active Contact opration, and ContactServiceObserve is used for receiving Contact related event.

3.4.1 ContactService

ContactService includes the following APIs:

boolean	hasContact(String contactUuId)
List<Contact>	queryAllContacts()
void	updateMyselfNickName(String nickName)
Contact	getMyself()
Contact	getContactBy(String contactUuId)
boolean	isMyself(String contactUuId)

3.4.1.1 Look Up Contact

- API: boolean hasContact(String contactUuId)

  (Available since Meshee SDK 0.8.0)

Parameters	
contactUuId	String: The contactUuId to query.

Returns	
boolean	Returns whether the Contact exists by the given contactUuId

  Look up whether the Contact exists by querying the contactUuId.

- API: List<Contact> queryAllContacts()

Returns	
List<Contact>	Returns the Contact list which are connected to the self terminal in the Meshee network. (Note: the returned Contact list always includes the Account Contact)

  Query the Contact list which are connected to the self terminal in the Meshee network (including the Account Contact on the self terminal). When the terminal's Network Role is Idle, the returned Contact list only inludes the Account Contact on the self terminal.

- API: Contact getMyself()

  (Available since Meshee SDK 0.8.0)

Returns	
Contact	Returns the Account Contact on the self terminal

  Get the Account Contact on the self terminal.

- API: Contact getContactBy(String contactUuId)

  (Available since Meshee SDK 0.8.0)

Parameters	
contactUuId	String: The contactUuId to query.

Returns	
Contact	Returns the Contact by the given contactUuId

  Look up the Contact by contactUuid.

- API: boolean isMyself(String contactUuId)

  (Available since Meshee SDK 0.8.0)

Parameters	
contactUuId	String: The contactUuId to query.

Returns	
boolean	Returns whether the Contact is the bound Account by contactUuId

  Look up whether the Contact is the bound Account by contactUuId.

3.4.1.2 Update Contact

- API: void updateMyselfNickName(String nickName)

  (Available since Meshee SDK 0.8.0)

Parameters	
nickname	String: The nickName to be updated for the bound Contact

Update the bound Account’ s nickName.

3.4.2 ContactServiceObserve

ContactServiceObserve includes the following APIs:

void	observeContactChange(Observer<Void> observer, boolean register)

3.4.2.1 Monitor Contact Change Event

- API: void observeContactChange(Observer<Void> observer, boolean register)

(Available since Meshee SDK 0.8.0)

Parameters	
observer	Observer<Void>: The observer of Contact change in the Meshee network in which the local terminal is.
register	boolean: The flag for registration or unregistration (true for registration and false for unregistration)

Register/Unregister Observer<Void> for receiving Contact change event in the Meshee network in which the local terminal is.

3.5 Conversation Service API

Conversation Service API is used for Conversation related operation.

- Conversation Service API package: cn.meshee.fclib.api.conversation
- Conversation Service includes ConversationService and ConversationServiceObserve. ConversationService is used for active Conversation opration, and ConversationServiceObserve is used for

receiving Conversation related event.

3.5.1 ConversationService

ConversationService includes the following APIs:

boolean	hasConversation(String convId)
List<Conversation>	queryAllConversations()
List<Conversation>	queryConverstaionBy(ConversationType conversationType)
Conversation	findP2PConversationBy(Contact peer)
Conversation	findConversationBy(String convId)
Conversation	createP2PConverstaion(Contact peer, String title, Uri portraitUri)
Conversation	createGroupConverstaion(List<Contact> participants, String title, Uri portraitUri)
void	removeConversation(String convId)
void	removeAllConversations()

3.5.1.1 Create Conversation

- API: Conversation createP2PConverstaion(Contact peer, String title, Uri portraitUri)

  (Available since Meshee SDK 0.8.0)

Parameters	
peer	Contact: The peer Contact for creating the P2P Conversation.
title	String: The title of the P2P conversation
portraitUri	Uri: The Uri of the P2P conversation portrait

Returns	
Conversation	Returns the created P2P Conversation (Note: If one Conversation has been already created for the given peer Contact, the previously created one will be

	returned)

Create a P2P Conversation, which is between the Account on the local terminal and the peer Contact.

- API: Conversation createGroupConverstaion(List<Contact> participants, String title, Uri portraitUri)

  (NOT supported yet)

Parameters	
participants	List<Contact>: The Contacts for creating the Group Conversation
title	String: The title of the Group conversation
portraitUri	Uri: The Uri of the Group conversation portrait

Returns	
Conversation	Returns the created Group Conversation (Note: Always a new Group Conversation is created, regardless whether previous Group(s) having the same contacts already exist(s))

Create a Group Conversation, which is between the Account and other Contact(s).

3.5.1.2 Delete Conversation

- API: void removeConversation(String convId)

  (Available since Meshee SDK 0.8.0)

Parameters	
convId	String: The conversationId for removing the Conversation.

Remove a conversation by the given conversationId.

- API: void removeAllConversations()

(Available since Meshee SDK 0.8.0)

Remove all the existing conversations.

3.5.1.3 Look Up Conversation

- API: hasConversation(String convId)

  (Available since Meshee SDK 0.8.0)

Parameters	
convId	String: The conversationId to query.

Returns	
boolean	Returns whether the Conversation exists by the given conversationId

Look up wheteher the conversation exists by conversationId.

- API: List<Contact> queryAllConversations()

  (Available since Meshee SDK 0.8.0)

Returns	
List<Conversation>	Returns the Conversation list

Look up the existing conversation list.

- API: List<Conversation> queryConverstaionBy(ConversationType

  conversationType)

  (Available since Meshee SDK 0.8.0)

Parameters	
conversationType	ConversationType: The ConversationType to query.

Returns	
List<Conversation>	Returns the conversation list of the queried ConversationType

Look up the conversation list which satisfies the given conversationType.

- API: Conversation findP2PConversationBy(Contact peer)

  (Available since Meshee SDK 0.8.0)

Parameters	
peer	Contact: The Contact to query.

Returns	
Conversation	Returns the P2P conversation which has the queried Contact

Look up the P2P Conversation by the peer Contact.

- API: Conversation findConversationBy(String convId)

  (Available since Meshee SDK 0.8.0)

Parameters	
convId	String: The conversationId to query.

Returns	
Conversation	Returns the P2P conversation by the given conversationId

Look the Conversation by conversationId.

3.5.2 ConversationServiceObserve API

ConversationServiceObserve includes the following APIs:

void	observeConversationChange(Observer<Void> observer, boolean register)

3.5.2.1 Monitor Conversation Change Event

- API: void observeConversationChange(Observer<Void> observer, boolean register)

(Available since Meshee SDK 0.8.0)

Parameters	
observer	Observer<Void>: The observer of Conversation change.
register	boolean: The flag for registration or unregistration (true for registration and false for unregistration)

Register/Unregister Observer<Void> for receiving Conversation change event.

3.6 Message Service API

Message Service API is used for Message related operation.

- Message Service API package: cn.meshee.fclib.api.Message
- Message Service includes FcMessageBuilder, MessageService and MessageServiceObserve. FcMessageBuilder is used for creating message, including text, image, file message for sending files and customized message. MessageService is used for active Message opration, and MessageServiceObserve is used for receiving Message related event.

3.6.1 FcMessageBuilder

FcMessageBuilder includes the following APIs:

static FcMessage	createTextMessage(String convId, String text)
static FcMessage	createImageMessage(String convId, String imagePath, String text)
static FcMessage	createFileMessage(String convId, String filePath)
static FcMessage	FcMessage createCustomMessage(String convId, Object extendObject)
static FcMessage	setMessageExtendObject(FcMessage message, Object extendObject)

3.6.1.1 Create FcMessage

- API: static FcMessage createTextMessage(String convId, String text)

  (Available since Meshee SDK 0.8.0)

Parameters	
convId	String: The conversationId for FcMessage
text	String: The text in the created FcMessage

Returns	
FcMessage	Returns the created FcMessage

Create a text FcMessage. Meshee SDK generates a unique messageId for the created text FcMessage.

- API: static FcMessage createImageMessage(String convId, String imagePath, String text)

  (Available since Meshee SDK 0.8.0)

Parameters	
convId	String: The conversationId for FcMessage
imagePath	String: The image path of the image in the FcMessage
text	String: The text in the created FcMessage

Returns	
FcMessage	Returns the created FcMessage

Create an image FcMessage. A text Message can be created when an image FcMessage is created. Meshee SDK generates a unique messageId for the created image FcMessage.

- API: static FcMessage createFileMessage(String convId, String filePath)

  (Available since Meshee SDK 0.8.0)

Parameters	
convId	String: The conversationId for FcMessage
filePath	String: The file path in FcMessage

Returns	
FcMessage	Returns the created FcMessage

Create a File FcMessage. File message is used for exchange file information before sending a file. Meshee SDK generates a unique messageId for the created File FcMessage.

- API: static FcMessage createCustomMessage(String convId, Object extendObject)

  (Available since Meshee SDK 0.8.0)

Parameters	
convId	String: The conversationId for FcMessage
extendObject	Object: The extended object in FcMessage

Returns	
FcMessage	Returns the created FcMessage

Create a customized FcMessage. Meshee SDK generates a unique messageId for the created customized FcMessage.

3.6.1.2 Set FcMessage

- API: static FcMessage setMessageExtendObject(FcMessage message, Object extendObject)

  (Available since Meshee SDK 0.8.0)

Parameters

convId	String: The conversationId for FcMessage
extendObject	Object: The extended object in FcMessage

Returns	
FcMessage	Returns the FcMessage after setting

Set the customized Object in a FcMessage.

3.6.2 MessageService

MessageService includes the following APIs:

void	sendMessage(FcMessage message, RequestCallback<FcMessage> requestCallback)

3.6.2.1 Send FcMessage

- API: void sendMessage(FcMessage message, RequestCallback<FcMessage> requestCallback)

  (Available since Meshee SDK 0.8.0)

Parameters	
message	FcMessage: The FcMessage to send
requestCallback	RequestCallback<FcMessage>: The callback of sending the FcMessage

Send a FcMessage. RequestCallback can have the result of sending a FcMessage, e.g., success, failure, exception.

3.6.3 MessageServiceObserve

MessageServiceObserve includes the following APIs:

void	observeIncomeMessage(Observer<List<FcMessage>> observer, boolean register)

3.6.3.1 Monitor Incoming FcMessage Event

- API: void observeIncomeMessage(Observer<List<FcMessage>> observer, boolean register)

  (Available since Meshee SDK 0.8.0)

Parameters	
observer	Observer<List<FcMessage>>: The observer of the incoming (e.g., received) List<FcMessage>.
register	boolean: The flag for registration or unregistration (true for registration and false for unregistration)

Register/Unregister Observer<List<FcMessage>> for receiving income FcMessage list event.

3.7 File Service API

File Service API is used for File related operation.

- File Service API package: cn.meshee.fclib.api.file
- File Service includes FileService and FileServiceObserve. FileService is used for active File opration, and FileServiceObserve is used for receiving File related event.

3.7.1 FileService

FileService includes the following APIs:

void	requestSendFile(FcMessage message, int ackTimeoutInSeconds)
void	ackReceiveFile(FcMessage message)

3.7.1.1 Send File Message

- API: void requestSendFile(FcMessage message, int ackTimeoutInSeconds)

  (Available since Meshee SDK 0.8.0)

Parameters	
message	FcMessage: The FcMessage for sending a file
ackTimeoutInSeconds	int: The timeout time in which ack the request is effective.

The file sender sends the file FcMessage about a file. AckTimeoutInSeconds specifies the timeout time since sending the file FcMessage. Only when the file sender receives the ack before times out, the sending file starts, or the file is not sent.

3.7.1.2 Ack File Message

- API: void ackReceiveFile(FcMessage message)

  (Available since Meshee SDK 0.8.0)

Parameters	
message	message: The received FcMessage to ack

The file receiver acks the file FcMessage, which is sent by file sender by calling the requestSendFile API.

3.7.2 FileServiceObserve

FileServiceObserve includes the following APIs:

void	observeFileReceiveProgress(Observer<FileProgress> observer, boolean register)
void	observeFileReceiveEvent(Observer<FileMessage> observer,

	boolean register)
void	observeFileSendEvent(Observer<IOException> observer, boolean register)

3.7.2.1 Monitor File Reception Progress Event

- API: void observeFileReceiveProgress(Observer<FileProgress> observer, boolean register)

  (Available since Meshee SDK 0.8.0)

Parameters	
observer	Observer<FileProgress>: The observer of FileProgress of the receiving file.
register	boolean: The flag for registration or unregistration (true for registration and false for unregistration)

Register/Unregister Observer<FileProgress> for receiving FileProgess change event.

3.7.2.2 Monitor File Reception Error Event

- API: void observeFileReceiveEvent(Observer<FileMessage> observer, boolean register)

  (Available since Meshee SDK 0.8.0)

Parameters	
observer	Observer<FileMessage>: The observer of FileMessage of the receiving file.
register	boolean: The flag for registration or unregistration (true for registration and false for unregistration)

Register/Unregister Observer<FileMessage> for receiving the error event when receiving a file. It is mainly about timeout event when acking the File

FcMessage.

3.7.2.3 Monitor File Send Exception Event

- API: void observeFileSendEvent(Observer<IOException> observer, boolean register)

  (Available since Meshee SDK 0.8.0)

Parameters	
observer	Observer<IOException>: The observer of IOException when sending the file.
register	boolean: The flag for registration or unregistration (true for registration and false for unregistration)

Register/Unregister Observer<IOException> for receiving error event when sending a file.

3.8 Live Service API

Live Service API is used for Live stream sharing related operation.

- Live Service API package: cn.meshee.fclib.api.live
- Live Service includes LiveService and LiveServiceObserve. LiveService is used for active Live sharing opration, and LiveServiceObserve is used for receiving Live sharing related event.

3.8.1 LiveService

LiveService includes the following APIs:

List<String>	queryMeshLiveResources()
void	setRemoteLiveHost(String host)

int	getLocalLiveProxyServerPort()

3.8.1.1 Get Local Live Proxy Server Port

- API: int getLocalLiveProxyServerPort()

  (Available since Meshee SDK 0.8.0)

Returns	
int	Returns local proxy server port for pulling remote cloud Live stream.

Get local http proxy port for pulling Internet Live stream. If Live stream proxy server is used, proxy is configured as the local IP "127.0.0.1" and the port # obatained by calling getLocalLiveProxyServerPort(). When the local Live proxy is configured, Meshee SDK will smartly choose mobile network Interface for pulling Internet Live stream if WiFi Interface is being used by connection of Meshee networks.

3.8.1.2 Look Up Live Resource in Meshee Network

- API: List<String> queryMeshLiveResources()

  (Available since Meshee SDK 0.8.0)

Returns	
List<String>	Returns the Mesh live resource list

Query Mesh live resource list. Each resource in the list can be used for pulling 列 Live stream from the Meshee network which the local terminal is in.

3.8.1.3 Set Host of Cloud Stream

- API: void setRemoteLiveHost (String host)

  (Available since Meshee SDK 0.8.0)

Parameters	
host	String: The remote cloud host for pulling Live stream

Set the remote host which is used for pulling Live stream from the Internet.

3.8.2 LiveServiceObserve

LiveServiceObserve includes the following APIs:

void	observeMeshLiveResourceChange(Observer<Void> observer, boolean register)
void	observeRemoteLiveHostExceptionEvent(Observer<IOException> observer, boolean register)

3.8.2.1 Monitor Live Resource Event in Meshee Network

- API: void observeMeshLiveResourceChange(Observer<Void> observer,

  boolean register)

  (Available since Meshee SDK 0.8.0)

Parameters	
observer	Observer<Void>: The observer of Mesh Live resource change.
register	boolean: The flag for registration or unregistration (true for registration and false for unregistration)

Register/Unregister Observer<Void> for receiving Meshee resource change event.

3.8.2.2 Monitor Remote Live Host Exception Event

- API: void observeRemoteLiveHostExceptionEvent(Observer<IOException> observer, boolean register)

  (Available since Meshee SDK 0.8.0)

Parameters	
observer	Observer<IOException>: The observer of IOException when pulling the Live stream from cloud.
register	boolean: The flag for registration or unregistration (true for registration and false for unregistration)

Register/Unregister Observer<IOException> for receiving remote host exception when pulling Live stream from the Internet.

3.9 Avatar Service API

Avatar Service API is used for Avatar related operation.

- Avatar Service API package: cn.meshee.fclib.api.avatar
- Avatar Service includes AvatarService and AvatarServiceObserve. AvatarService is used for active Avatar opration, and AvatarServiceObserve is used for receiving Avatar related event.

3.9.1 AvatarService

AvatarService includes the following APIs:

Avatar	saveMyAvatar(String avatarPath)
void	pushAvatarTo(List<Contact> contacts)
void	pushAvatarToAll()
Uri	getAvatarUri(String contactUuId)

3.9.1.1 Save Account Avatar

- API: Avatar saveMyAvatar(String avatarPath)

  (Available since Meshee SDK 0.8.0)

Parameters	
avatarPath	String: The path of the Avatar to be saved.

Returns	
Avatar	Returns the saved Avatar

Persistently save the Avatar image corresponding to the Avatar path. Exiting Meshee SDK will not detele the persistently saved Avatar. Saved Account Avatars can be read.

3.9.1.2 Push Account Avatar

- API: void pushAvatarTo(List<Contact> contacts)

  (Available since Meshee SDK 0.8.0)

Parameters	
contacts	List<Contact>: The contact list to push the Account Avatar

Push the Account Avatar to all the Contacts in the Contact list. When pushing the Avatar, the nickname of the Account is pushed as well.

- API: void pushAvatarToAll()

  (Available since Meshee SDK 0.8.0)

Push the Account Avatar to all the Contacts connected to the Account of the local terminal. When pushing the Avatar, the nickname of the Account is pushed as well.

3.9.1.3 Look Up Contact Avatar Uri

- API: Uri getAvatarUri(String contactUuId)

  (Available since Meshee SDK 0.8.0)

Parameters	
contactUuId	String: The contactUuId to query.

Returns	
Uri	Returns the Avatar Uri by the given contactUuId

Look up Contact’ s Avatar Uri by contactUuId.

3.9.2 AvatarServiceObserve

AvatarServiceObserve includes the following APIs:

void	observeAvatarChange(Observer<Avatar> observer, boolean register)
void	observeAvatarSendEvent(Observer<IOException> observer, boolean register)

3.9.2.1 Monitor Contact Avatar Change Event

- API: void observeAvatarChange(Observer<Avatar> observer, boolean register)

  (Available since Meshee SDK 0.8.0)

Parameters	
observer	Observer<Avatar>: The observer of Avatar.
register	boolean: The flag for registration or unregistration (true for registration and false for unregistration)

Register/Unregister Observer<Avatar> for receiving Contact Avatar change event.

3.9.2.2 Monitor Account Avatar Push Exception Event

- API: void observeAvatarSendEvent(Observer<IOException> observer, boolean register)

  (Available since Meshee SDK 0.8.0)

Parameters	
observer	Observer<IOException>: The observer of IOException when pushing the Avatar.
register	boolean: The flag for registration or unregistration (true for registration and false for unregistration)

Register/Unregister Observer<IOException> for receiving exception while pushing Account Avatar. Avatar file size can not exceed 512K bytes, or exception is reported.

4 Meshee Model

Meshee model is used by Meshee API. Meshee model includes:

- Meshee SDK management model for initializing SDK, exiting SDK、getting SDK service and SDK Account management
- Setting Service model for getting and configuring basic SDK setting
- Network Service model for Meshee network related operation
- Contact Service model for Contact related operation
- Conversation Service model for Conversationrelated operation
- Message Service model for Message related operation
- File Service model for File related operation

- Live Service model for Live stream sharing related operation
- Avatar Service model for Avatar related operation

4.1 Meshee SDK Management Model

Meshee SDK management model is used for initializing SDK, exiting SDK、getting SDK service and SDK Account management.

- Meshee SDK management model package: cn.meshee.fclib.api.model
- Meshee SDK management model includes NetworkStrategyType and SdkOptions

4.1.1 NetworkStrategyType

- NetworkStrategyType includes: (Available since Meshee SDK 0.8.0)

Fields	
NetworkStrategyType1	enum element: networkStrategy Type of NetworkStrategyType1
NetworkStrategyType2	enum element: networkStrategy Type of NetworkStrategyType2

NetworkStrategyType is the network type that Meshee SDK is to support - NetworkStrategyType1 or NetworkStrategyType2. NetworkStrategyType1 is a Meshee network with a diameter of 400 meters covering ~100 terminals; NetworkStrategyType2 is a Meshee network with a dimeter of 200 meters covering ~10 terminals. NetworkStrategyType1 has the advantage of larger network coverage, more terminals and more distant communication capability. NetworkStrategyType2 has higher P2P transferring speed and smaller transferring

delay in average.

4.1.2 SdkOptions

- SdkOptions includes: (Available since Meshee SDK 0.8.0)

Constants	
DEFAULT	SdkOptions: The default SdkOptions for SDK initialization.
Fields	
sdkStorageRootPath	String: The storage path for SDK
networkStrategyType	NetworkStrategyType: The network strategy type for SDK

SdkOptions is used for initializing SDK. SdkOptions for initialization can customize SdkStorageRootPath and NetworkStrategyType.

4.2 Setting Service Model

Setting Service model is used for getting and configuring basic SDK setting. No such model needed yet.

4.3 Network Service Model

Network Service model is used for Meshee network related operation.

- Network Service model package: cn.meshee.fclib.api.network.model
- Network Service model includes Role, RoleEvent, NetworPoint, NetworkEvent and ConnectionState.

4.3.1 Role

- Role includes: (Available since Meshee SDK 0.8.0)

Fields	
Idle	enum element: State when NOT in any Meshee network.
Owner	enum element: State when a Meshee network is already created by itself and can act as a NetworkPoint for connection by other devices.
Member	enum element: State when a Meshee network is already joined and can act as a NetworkPoint for connection by other devices.)

Network Role model. There are three Network Roles: Idle, Owner and Member. When a terminal is not in any Meshee network, it is IDLE Role; when a Meshee network is created by a terminal, it is Owner Role; when a terminal already joins a Meshee network, it is Member Role.

4.3.2 RoleEvent

- RoleEvent includes: (Available since Meshee SDK 0.8.0)

Fields	
oldRole	Role: Network Role before this RoleEvent happens
newRole	Role: Network Role after this RoleEvent happens

Event model of Network Role change. The event includes the Network Role before the event occurs and the Network Role after the event occurs.

4.3.3 NetworkPoint

- NetworkPoint includes: (Available since Meshee SDK 0.8.0)

Fields	
networkPointId	String: NetworkPointId of the NetworkPoint
networkRole	Role: NetworkRole of the NetworkPoint
groupId	UUID: GroupId of the group which the NetworkPoint lies in
groupName	String: GroupName of the group which the NetworkPoint lies in
rssi	int: Received Signal Strength Indication (RSSI) of the NetworkPoint
frequency	int: Frequency band of the NetworkPoint

NetworkPoint is the model of Meshee access point. It includes the networkPointId of the access point, the groupId and the groupName of the group which the access point is in, and the accessp point' s rssi, frequency band and Network Role.

4.3.4 NetworkEvent

- NetworkPoint includes: (Available since Meshee SDK 0.8.0)

Inner models:	
NetworkEventType	enum: NetworkEvent Type enum
fields:	
networkEventType	NetworkEventType: networkEvent Type of this event
networkPointId	String: NetworkPointId of the NetworkPoint in this event
groupId	UUID: GroupId of the group which the NetworkPoint lies in
groupName	String: GroupName of the group which the NetworkPoint lies in
success	boolean: Whether the network operation in the networkEvent is successful

NetworkEvent is the Network creating/destroying event model. Only the

Owner Role terminal can destroy a Meshee network. NetworkEvent includes networkEventType, the networkPointId of the created/destroyed networkPoint, the grouId and the groupName of the group created by the networkPoint, and success flat of creating/destroying the Meshee network.

- NetworkEventType includes: (Available since Meshee SDK 0.8.0)

Fields	
NETWORK_EVENT_CREATE	enum element: Network event Type for network creation
NETWORK_EVENT_DESTROY	enum element: Network event Type for network destroying

NetworkEvent' s inner model NetworkEventType. NetworkEvent includes Meshee network creating/destroying event.

4.3.5 ConnectionState

- ConnectionState includes: (Available since Meshee SDK 0.8.0)

Inner models:	
State	enum: Connection State enum
fields:	
NetworkPoint	NetworkPoint: networkPoint of this event
state	State: latest connection state of the NetworkPoint in this event

ConnectionState is the state model of connecting a NetworkPoint. ConnectionState includes the latest connection state of the NetworkPoint.

- State includes: (Available since Meshee SDK 0.8.0)

Fields	
DISCONNECTED	enum element: the Initial state before connecting a networkPoint or state when a connected networkPoint is disconnected

CONNECTING	enum element: the state when connecting to a networkPoint
CONNECTED	enum element: the state when a networkPoint is connected
FAILED	enum element: the state when connecting to a networkPoint fails due to timeout or other reasons.

ConnectionState' s inner model State. State includes DISCONNECTED, CONNECTING, CONNECTED AND FAILED of connecting a NetworkPoint. DISCONNECTED is also the initial state of connecting a NetworkPoint.

4.4 Contact Service Model

Contact Service model is used for Contact related operation.

- Contact Service model package: cn.meshee.fclib.api.contact.model
- Contact Service model includes Contact.

4.4.1 Contact

- Contact includes: (Available since Meshee SDK 0.8.0)

Fields	
contactRawId	String: contactRawId for the Contact
nickname	String: nickname for the Contact
contactUuid	UUID: contactUuid for the Contact

Contact model includes contactRawId, nickname and contactUuid, which can be custromized. ContactRawId can be used to describe the 3rd party account, e.g., email, phone number, Whatsapp Id, Facebook Id and Wechat Id. App needs guarantee the contactUuid of each Contact is unique.

4.5 Conversation Service Model

Conversation Service model is used for Conversation related operation.

- Conversation Service model package: cn.meshee.fclib.api.conversation.model
- Conversation Service model includes ConversationType and Conversation.

4.5.1 ConversationType

- ConversationType includes: (Available since Meshee SDK 0.8.0)

Fields	
NONE	enum element: NONE Conversation type
P2P	enum element: P2P Conversation type.
GROUP	enum element: Group Conversation type.

ConversationType model includes three types： NONE, P2P and GROUP. P2P is for P2P conversation, GROUP is for group conversation (GROUP conversationis not supported yet). NONE is and illegal conversation.

4.5.2 Conversation

- Conversation includes: (Available since Meshee SDK 0.8.0)

Fields	
conversationType	ConversationType: the conversation type of the conversation
initiator	Contact: the Contact who initiates the conversation
participant	List<Contact>: the list of participant contacts in the conversation except myself contact
conversationId	String: the conversationId of the conversation
conversationTitle	String: the title of the conversation

portraitUri	String: the Uri of the portrait

Conversation model defines a Conversation, including ConversationType, initiator (i.e., the starter of the Conversation), participant (i.e., all the Contacts in the Conversation except the Account on local the terminal), conversationId, conversationTitle and portraitUri (i.e., the Conversation avatar’ s Uri).

4.6 Message Service Model

Message Service model is used for Message related operation.

- Message Service model package: cn.meshee.fclib.api.message.model
- Message Service model includes ChatType, ChatTypeEnum, MessageType, MessageTypeEnum, MediaName and FcMessage.

4.6.1 ChatType

- ChatType includes: (Available since Meshee SDK 0.8.0)

Constants	
UNKNOWN	int: the unknown ChatType
ONE_ONE_CHAT	int: the ChatType of one-to-one chat
GROUP_CHAT	int: the ChatType of group chat
SYSTEM_MESSAGE	int: the ChatType of system message
FREE_CHAT	int: the ChatType of free chat

ChatType model defines the chat type supported by SDK. ONE_ONE_CHAT is 为点 the peer to peer chat type between two Contacts, SYSTEM_MESSAGE is the chat type generated by system message, GROUP_CHAT is the chat type between at least two contacts, and UNKNOWN is an undefined chat type. FREE_CHAT chat type is not used yet.

4.6.2 ChatTypeEnum

- ChatTypeEnum includes: (Available since Meshee SDK 0.8.0)

Fields	
unknown	enum element: ChatType for UNKNOWN
oneonechat	enum element: ChatType for ONE_ONE_CHAT
groupchat	enum element: ChatType for GROUP_CHAT
systemessage	enum element: ChatType for SYSTEM_MESSAGE
freechat	enum element: ChatType for FREE_CHAT

ChatTypeEnum is enum corresponding to ChatType constant.

4.6.3 MessageType

- MessageType includes: (Available since Meshee SDK 0.8.0)

Constants	
unknown	int: the unknown MessageType
custom	int: the customized MessageType
tip	int: the MessageType of tip
notification	int: the MessageType of notification
avatar	int: the MessageType of avatar
file	int: the MessageType of file
location	int: the MessageType of location
video	int: the MessageType of video
audio	int: the MessageType of audio
image	int: the MessageType of image
text	int: the MessageType of text

Message model defines all message types supported by SDK.

4.6.4 MessageTypeEnum

- MessageTypeEnum includes: (Available since Meshee SDK 0.8.0)

fields	
unknown	enum element: the unknown MessageType

custom	enum element: the customized MessageType
tip	enum element: the MessageType of tip
notification	enum element: the MessageType of notification
avatar	enum element: the MessageType of avatar
file	enum element: the MessageType of file
location	enum element: the MessageType of location
video	enum element: the MessageType of video
audio	enum element: the MessageType of audio
image	enum element: the MessageType of image
text	enum element: the MessageType of text

MessageTypeEnum is enum corresponding to MessageType constant.

4.6.5 MediaName

- MediaName includes: (Available since Meshee SDK 0.8.0)

Fields	
name	String: the name of the media
extension	String: the extension of the media

MediaName defines a media format. MediaName includes mediaName and extension postfix.

4.6.6 FcMessage

- FcMessage is the basic message model used for sending and receiving messages in Meshee network.

4.7 File Service Model

File Service model is used for File related operation.

- File Service model package: cn.meshee.fclib.api.file.model
- File Service model includes FileMessage, FileProgress, FileIdentity,

FileEnvelope, FileLocalSummary, FTcpHandShake, FileStatus, FileAckStatus and FileReceiveStatus.

4.7.1 FTcpHandShake

- FTcpHandShake includes: (Available since Meshee SDK 0.8.0)

Fields	
FTcpSync	enum element: Sync phase in FTcphandshake.
FTcpAck	enum element: FTcpAck phase in FTcphandshake.
FTcpError	enum element: FTcpError phase in FTcphandshake.

The mode is for fTcp handshake of file transferring, including three states: FTcpSync (i.e., file sender sends file FcMessage about the file to send), FTcpAck (i.e., file receiver ack the file FcMessage and get ready to receive the file) and FTcpError (i.e., error occurs during handshake).

4.7.2 FileIdentity

- FileIdentity includes: (Available since Meshee SDK 0.8.0)

Fields	
fcMessageId	String: messageId of the FcMessage which is related to the file for transferring
conversationId	String: conversationId of the file for transferring
fileId	String: fileId of the file for transferring
fileInstanceId	String: fileInstanceId of the file for transferring

FileIdentity model is used to describe all sorts of information related to the file for transferring, including fcMessageId (i.e., messageId of file’ s related file FcMessage), conversationId (i.e., conversationId of the conversation in which file

transferring occurs), fileId (i.e., file Id of the file to transfer) and fileInstanceId (i.e., file instance Id of the file to transfer).

4.7.3 FileEnvelope

- FileEnvelope includes: (Available since Meshee SDK 0.8.0)

Fields	
sender	Contact: the Contact to send the file
receiver	Contact: the Contact to receive the file

FileEnvelop model is used to describe the Contact information of the contacts who sends and receives the file.

4.7.4 FileStatus

- FileStatus includes: (Available since Meshee SDK 0.8.0)

Fields	
fileAckStatus	FileAckStatus: the ack status of the file
fileReceiveStatus	FileReceiveStatus: the receive status of the file

FileStatusmodel is used to describe the status information of the file for transferring, including fileAckStatus and fileReceiveStatus.

4.7.5 FileAckStatus

- FileAckStatus includes: (Available since Meshee SDK 0.8.0)

Inner models:	
AckStatus	enum: Ack Status enum
fields:	
ackTimeout	int: the timeout time in which ack the request is effective.

ackFailure	String: the detailed description of ack failure
ackStatus	AckStatus: the ackStatus of the file for transferring

FileAckStatusis is used to describe the ack status information of the file for transferring, including ackTimeout, ackFailure and ackStatus.

- AckStatus includes: (Available since Meshee SDK 0.8.0)

Fields	
Acked	enum element: the state when file is acked
Not_Acked	enum element: the state when file is not acked
Ack_Failure	enum element: the state when file ack is a failure

FileAckStatus’ s inner model AckStatus includes the ack status of a file for transferring. AckStatus includes Acked, Not_Acked and Ack_Failure.

4.7.6 FileReceiveStatus

- FileReceiveStatus includes: (Available since Meshee SDK 0.8.0)

Inner models:	
ReceiveStatus	enum: Receive Status enum
fields:	
receiveStatus	ReceiveStatus: the receiveStatus of the file for transferring
receivePercentage	int: the receive percentage information of the transferring file
receiveFailure	String: the detailed description of receive failure transferring

FileReceiveStatus model is used for describing the receiving status of a file for transferring, including receiveStatus、receivePercentage and receiverFailure.

- ReceiveStatus includes: (Available since Meshee SDK 0.8.0)

Fields	
Not_Start	enum element: the state when receiving file has not started

InProgress	enum element: the state when file receiving is in progress
Success	enum element: the state when file is received successfully
Failure	enum element: the state when receiving file is a failure

FileReceiveStatus’s inner model ReceiveStatus includes the receive status of a file. FileReceiveStatus includes Not_Start, InProgress, Success and Failure.

4.7.7 FileLocalSummary

- FileLocalSummary includes: (Available since Meshee SDK 0.8.0)

Fields	
localPath	String: the local path of the file
filename	String: the name of the file
fileSize	String: the size of the file

FileLocalSummary model is used for describing a file’s summary, including file’s local path, name and size.

4.7.8 FileMessage

- FileMessage includes: (Available since Meshee SDK 0.8.0)

Fields	
fileIdentity	FileIdentity: Identity of the file
fileEnvelope	FileEnvelope: Envelope of the file
fileLocalSummary	FileLocalSummary: LocalSummary of the file
fileStatus	FileStatus: FileStatus of the file
fTcpHandShake	FTcpHandShake: HandlShake status of the file

FileMessage model is used for describe all sorts of information of a file for transferring, including FileIdentity (i.e., file identity information), FileEnvelope (i.e., file envelope information of the sender Contact and receiver Contact),

FileLocalSummary (i.e., file summary information), FileStatus (i.e., file ack and receive status information) and FTcpHandShake (i.e., file fTcp handshake information).

4.7.9 FileProgress

- FileProgess includes: (Available since Meshee SDK 0.8.0)

Fields	
fileIdentity	FileIdentity: Identity of the file
percent	int: percentile information of the transferring file

FileProgess model is used for describing the progress information of the transferring file: FileIdentity (i.e., file identity information) and percentage information.

4.8 Live Service Model

Live Service model is used for live stream sharing related operation. No such model needed yet.

4.9 Avatar Service Model

Avatar Service Model is used for Avatar related operation.

- Avatar Service model package: cn.meshee.fclib.api.avatar.model
- Avatar Service model includes Avatar.

4.9.1 Avatar

- Avatar includes: (Available since Meshee SDK 0.8.0)

Fields	
contactUuid	UUID: the contactUuid of the avatar
avatarPath	String: the avatar Path of the avatar
avatarMd5	String: MD5 of the avatar

Avatar model is used for describing an Contact' s Avatar information, including contactUuid, avatarPath and avatar MD5.