Meshee SDK 开发指南 (V0.8.0)

目录

1 Meshee 概述

Meshee 主要在需要节省流量费用或蜂窝（例如 4G）网络不发达的场景下使用。当在 400 米范围内有一定数量的终端存在时，Meshee 技术支持这些终端组成 Meshee 网络，任意两个终端之间可以通过这个 Meshee 网络进行高速网络通信，且不使用蜂窝流量。Meshee 网络中的通信信息不通过 Internet cloud 服务器，具有天然的隐私优势。

- Meshee API 现阶段适用于 Android 终端（>=Android API 21），IOS、windows 终端待开发。
- Meshee API 基于自主开发的 Meshee 网络与通信协议，提供了基于本地 Meshee 网络的网络服务功能。这些功能在如下两个阶段进行使用:
  - Pre-communication 阶段：在还未进行 Meshee 网络通信前，创建、发现、加入 Meshee 网络。
  - In-communication 阶段：在一个形成的 Meshee 网络中进行交互数据，数据形式可以为文字，图片，文件，流媒体等。
- Meshee API 提供全面的通信服务接口，包括用于通信的 Contact、Conversation、Message（包括文字、图片、自定义类型）、File、流媒体等。利用以上接口，开发者可以方便实现各种本地通信场景，例如：
  - 视频分享 - 在 Meshee 网络上分享各种视频，支持 http-flv 格式，其它格式待开发。
  - 文件分享 - 在 Meshee 网络上轻松实现大容量文件传输功能。
  - 多人游戏 - 使用自定义消息类型实现游戏用消息类型扩展。
  - 社交聊天/协作办公 - 结合文字，图片，文件的消息类型组合实现社交聊天/

协作办公的需求。

2 Meshee 的特点

Meshee 具有如下特点：

- 网络覆盖面积大 – Meshee 网络底层传输基于 WiFi 构建，网络直径最大可达 400 米。
- 网络终端数量多 – 一个 Meshee 网络最多可以容纳~100 台终端。
- 传输带宽大 – 网络中任意两点的最大传输带宽可达~100Mbps。
- Meshee 网络与蜂窝网络同时使用 – 终端在使用 Meshee 网络时，可以同时使用蜂窝网络连接 Internet。
- 多路传输 – Meshee 网络通信协议动态分配带宽给同时并发的多路传输，保证多路传输的公平性。
- 接入能力强大 – 当蜂窝网络可用时，终端可以使用极小的蜂窝网络流量获得 Meshee 网络接入点信息进行接入，以提高终端接入 Meshee 网络的能力。
- 通信消息格式扩展 – 针对不同的应用需求，Meshee 支持针对各种文本、多媒体通信格式的快速扩展。

3 Meshee API

Meshee API 包括：

- Meshee SDK 管理 API，用于初始化 SDK、退出 SDK、获取 SDK service 与管理

SDK Account。

- Setting Service API，用于获取与配置基本的 SDK setting。
- Network Service API，用于 Meshee 网络相关的操作。
- Contact Service API，用于 Contact 相关的操作。
- Conversation Service API，用于 Conversation 相关的操作。
- Message Service API，用于 Message 相关的操作。
- File Service API，用于 File 相关的操作。
- Live Service API，用于 Live stream 分享相关的操作。
- Avatar Service API，用于 Avatar 相关的操作。

3.1 Meshee SDK 管理 API

Meshee SDK 管理 API，用于初始化 SDK、退出 SDK、获取 SDK service 与管理 SDK Account。

- Meshee SDK 管理 API 所在包：cn.meshee.fclib.api。
- Meshee SDK 管理由 FCClient 完成。

3.1.1 FCClient

FCClient 的 Meshee SDK 管理 API 包括如下 API：

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 初始化 SDK

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

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

一台终端在使用 Meshee SDK 前，首先必须使用 App 的 main thread 初始化 SDK。初始化的 SdkOptions 可以定制 SdkStorageRootPath 与 NetworkStrategyType。

SdkStorageRootPath 为保存 SDK 运行过程中保存内容的路径。

NetworkStrategyType 为 SDK 将支持的网络模式：NetworkStrategyType1 与 NetworkStrategyType2。NetworkStrategyType1 是一个 Meshee 网络覆盖范围最大 400 米，终端数量最多~100 台；NetworkStrategyType2 是一个 Meshee 覆盖范围最大 200 米，终端数量最多~10 台。NetworkStrategyType1 的优势在于网络覆盖范围大，终端数量多，通信距离远。NetworkStrategyType2 的优势在于，任意两个终端之间的 Peer-to-Peer 平均传输速度更快，平均传输时延更小。

- API: public static void init(Application application)

Parameters	
Application	Application: The application that uses Meshee SDK.

一台终端在使用 Meshee SDK 前，首先必须使用 App 的 main thread 初始化 SDK。当不指定 SdkOptions 时，初始化的 SdkOptions 为缺省配置 SdkOptions.DEFAULT。在缺省配置 SdkOptions.DEFAULT 下，SdkStorageRootPath 对应的目录为系统根目录中以 App PackageName 命名的目录，NetworkStrategyType 为 NetworkStrategyType1。

3.1.1.2 获取 SDK service

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

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

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

App 通过调用该 API 获取 Meshee service。该 API 必须在 SDK 初始化后才能使用。Meshee service 包括 Setting Service，Network Service，Contact Service，Conversation Service，Message Service，File Service，Live Service 与 Avatar Service。App 通过调用该 API 获取特定的 service 后，可进一步调用 service 的 API。例如：

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

3.1.1.3 管理 SDK Account

- API: public static void bindAccount(Contact contact)

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

绑定指定的 Contact 到 Meshee SDK 作为 Account。该 API 必须在 SDK 初始化后才能使用。一直到解绑该 Account 或 SDK 退出，此 Account 将作为当前在使用 SDK 的唯一 Account。

- API: public static void bindAccount()

绑定默认的 Contact 到 Meshee SDK 作为 Account。该 API 必须在 SDK 初始化后才能使用。该默认 Account 将由 Meshee SDK 为 App 生成。直到解绑该 Account 或 SDK 退出，此默认 Account 将作为当前在使用 SDK 的唯一 Account。

- API: public static void unbindAccount ()

解绑已绑定到 Meshee SDK 的 Account。该 API 必须在 SDK 绑定 Account 后才能使用。解绑操作不会直接退出 SDK，SDK 将回到初始化完成且尚未绑定 Account 的状态。

- API: public static boolean isAccountBound()

Returns	
boolean	Returns whether an Account is already bound.

获取当前是否有 Account 绑定到 Meshee SDK。

3.1.1.4 退出 SDK

- API: public static void exit()

退出 SDK，App 也将随之退出。

3.2 Setting Service API

Setting Service API 用于获取与配置基本的 SDK setting。Setting Service 在 SDK 初始化后才能使用。

- Setting Service API 所在包：cn.meshee.fclib.api.setting。
- Setting Service 包含 SettingService，用于主动进行 setting 操作。

3.2.1 SettingService

SettingService 包括如下 API：

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

3.2.1.1 获取 SDK 初始化配置

- API: String getSdkStorageRootPath()

Returns	
String	Returns the configured SdkStorageRootPath in SDK initialization.

获取 SdkStorageRootPath。如果 public static void init(Application application, SdkOptions sdkOptions)的 sdkOptions 中配置了 sdkStorageRootPath，则此 API 返回的为配置的 sdkStorageRootPath，如果未进行配置或使用 public static void init(Application application)进行了 SDK 初始化，则返回缺省的 sdkStorageRootPath。

- API: NetworkStrategyType getNetworkStrategyType()

Returns	
NetworkStrategyType	Returns the configured NetworkStrategyType in SDK initialization.

获取 NetworkStrategyType。如果 public static void init(Application application, SdkOptions sdkOptions)的 sdkOptions 中配置了 NetworkStrategyType，则此 API 返回的为配置的 networkStrategyType，如果未进行配置或使用 public static void init(Application application)进行了 SDK 初始化，则返回缺省的 NetworkStrategyType。

3.2.1.2 Contact Account 保存与获取

- API: void saveAccount(Contact contact)

Parameters

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

保存 Contact 到终端作为永久 Account。Meshee SDK 退出不会删除调用该 API 保存的 Account。

- API: List<Contact> getAllAccounts()

Returns	
List<Contact>	Returns all saved Accounts.

获取所有保存的 Accounts。

3.2.1.3 获取 Local proxy server port

- API: int getLocalProxyServerPort()

Returns	
int	Returns the local proxy server port.

获取本地 http/https 代理端口。如需使用该 proxy server，代理配置成 local IP "127.0.0.1"与 getLocalProxyServerPort()所对应的 port 号。使用该本地代理时，如果 WiFi Interface 被 Meshee 网络使用，Meshee SDK 会智能选择使用 mobile network Interface 进行 Internet 通信。

3.3 Network Service API

Network Service API 用于 Meshee 网络相关的操作。Network Service 必须在 SDK 绑定 Account 后才能使用。

- Network Service API 所在包：cn.meshee.fclib.api.network。
- Network Service 包含 NetworkService 与 NetworkServiceObserve。NetworkService 用于主动进行 network 操作，NetworkServiceObserve 用于被动接收与 Meshee 网络相关的 event。

3.3.1 NetworkService

NetworkService 包括如下 API：

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

3.3.1.1 关于 Meshee 网络角色

- Role getRole()

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.)

查询终端当前所处的 Meshee 网络角色。Meshee 网络角色包括 Idle，Owner 与 Member。详见上表。

- API: void backToIdle()

一台 Meshee 网络角色为 Owner 或 Member 的终端可以调用该 API 返回 Idle 角色，同时退出原来所在的 Meshee 网络。

3.3.1.2 创建 Meshee 网络

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

Parameters	
groupName	String: The group name of for the Meshee network to be created.
need5G	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.)

一台终端可以创建一个 Meshee 网络。一旦创建成功，该终端的网络角色会变成 Ownen，同时该终端变成 NetworkPoint 可供其它终端进行扫描发现。终端在创建一个 Meshee 网络时会同时创建一个 group，groupName 可在创建网络时定制，SDK 会为 group 生成一个独一无二的 groupId。

终端可以选择使用 2.4GHz 判断或者 5GHz 频段有先进行创建 Meshee 网络，2.4GHz 的网络穿越遮挡物能力更好，但是 5GHz 的网络在遮挡不严重的条件下传输速度更快。从终端普及率来看，2.4GHz 大大高于 5GHz。一台处于 Idle 角色、Owner 角色或 Member 角色的终端可以调用该 API。

3.3.1.3 扫描 Meshee 网络

- API: void scan()

无条件扫描附近的 Meshee 网络的 NetworkPoint。当终端的 WiFi 处于 disabled 状

态（终端可处于 Idle 角色或 Owner 角色）时，调用该 API 将首先 enable WiFi 并开始扫描；当终端的 WiFi 处于 enabled 状态（终端可处于 Idle 角色或 Member 角色）时，调用该 API 将立即开始扫描。

- API: void softScan()

  有条件扫描附近的 Meshee 网络的 NetworkPoint。即只有当终端处于 WiFi enabled 的状态（终端可处于 Idle 角色或 Member 角色）时，调用该 API 才能开始扫描。

- API: void stopScan()

  中止扫描附近的 Meshee 网络的 NetworkPoint。

3.3.1.4 加入 Meshee 网络

- API: void joinNetwork(NetworkPoint networkPoint)

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

选择一个 Meshee 网络的一个 NetworkPoint 进行连接，加入该 Meshee 网络。一旦加入网络成功，该终端的网络角色会变成 Member。一台处于 Idle 角色或 Member 角色的终端可以调用该 API。

3.3.1.5 查询 Meshee 网络信息

- List<NetworkPoint> getNetworkPoints()

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

查询最近一次扫描获得的 Meshee 网络的 NetworkPoint 列表。

- NetworkPoint getConnected()

Returns	
NetworkPoint	Returns the currently connected NetworkPoint.

查询当前所连接的 Meshee 网络的 NetworkPoint。

3.3.1.6 关于自动加入 Meshee 网络

- API: void restartAutoJoinNetwork()

重启自动加入 Meshee 网络，即重新进入 autoJoinNetwork 状态。一台处于 Idle 角色、Member 角色或 Owner 角色的终端可调用该 API。处于 Member 角色或 Owner 角色的终端调用该 API 将先进入 Idle 角色，然后尝试重启自动加入 Meshee 网络。调用该 API 进入 autoJoinNetwork 状态后，终端一旦进入 Idle 角色且 WiFi 处于 enabled 的状态，将自动连接 NetworkPoint。

- API: void stopAutoJoinNetwork()

停止自动加入 Meshee 网络，即退出 autoJoinNetwork 状态。一台处于 Idle 角色、Member 角色或 Owner 角色的终端可调用该 API。处于 Member 角色或 Owner 角色的终端调用该 API 将不会改变当前网络角色。调用该 API 之后退出 autoJoinNetwork 后，终端一旦进入 Idle 角色且 WiFi 处于 enabled 的状态，将不会自动连接 NetworkPoint。

- API: boolean isAutoJoinNetwork()

Returns	
boolean	Returns whether in autoJoinNetwork status.

查询终端是否处于 autoJoinNetwork 状态。

3.3.2 NetworkServiceObserve

NetworkServiceObserve 包括如下 API：

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 监视 Meshee 网络创建/销毁事件

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

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)

注册 Observer<NetworkEvent>，接收 Meshee 网络的创建/销毁事件。只有处于 Owner 角色的终端才可以销毁所在的 Meshee 网络。

3.3.2.2 监视 Meshee 网络连接状态事件

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

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)

注册 Observer<ConnectionState>，接收其关于 Meshee 网络的 NetworkPoint 的连接状态事件。

3.3.2.3 监视 Meshee 网络 NetworkPoint 扫描事件

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

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)

注册 Observer<List<NetworkPoint>>，接收最新扫描的 NetworkPoint 列表。

3.3.2.4 监视 Meshee 网络角色事件

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

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)

注册 Observer<RoleEvent>，接收终端网络角色变化的事件。

3.4 Contact Service API

Contact Service API 用于 Contact 相关的操作。Contact Service 必须在 SDK 绑定 Account 后才能使用。每个终端绑定的 Contact 都是独一无二的。

- Contact Service API 所在包：cn.meshee.fclib.api.contact。
- Contact Service 包含 ContactService 与 ContactServiceObserve 。ContactService 用于主动进行 Contact 操作，ContactServiceObserve 用于被动接收与 Contact 相关的 event。

3.4.1 ContactService

ContactService 包括如下 API：

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

3.4.1.1 查询 Contact

- API: boolean hasContact(String contactUuId)

Parameters	
contactUuId	String: The contactUuId to query.

Returns	
boolean	Returns whether the Contact exists by the given contactUuId

通过 contactUuId 查询 Contact 是否存在。

- API: List<Contact> queryContacts()

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

获取与 Account Contact 处于连接可通信状态的 Contact 列表。当终端处于 Idle 角色时，则返回的 Contact 列表只包含 Account Contact。

- API: Contact getMyself()

Returns	
Contact	Returns the Account Contact

获取已绑定的 Account Contact。

- API: Contact getContactBy(String contactUuId)

Parameters	
contactUuId	String: The contactUuId to query.

Returns	
Contact	Returns the Contact by the given contactUuId

通过 contactUuId 查询 Contact。

- API: boolean isMyself(String contactUuId)

Parameters	
contactUuId	String: The contactUuId to query.

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

通过 contactUuId 查询是否为 Account Contact。

3.4.1.2 更新 Contact

- API: void updateMyselfNickName(String nickName)

Parameters	
nickname	String: The nickname to be updated for the Account Contact

更新 Account Contact 的 nickName。

3.4.2 ContactServiceObserve

ContactServiceObserve 包括如下 API：

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

3.4.2.1 监视 Contact 变化事件

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

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

注册 Observer<Void>，接收与 Account Contact 连接的 Contact 变化事件。

3.5 Conversation Service API

Conversation Service API 用于 Conversation 相关的操作。Conversation Service 必须在 SDK 绑定 Account 后才能使用。

- Conversation Service API 所在包：cn.meshee.fclib.api.conversation。
- Conversation Service 包含 ConversationService 与 ConversationServiceObserve，ConversationService 用于主动进行 Conversation 操作，ConversationServiceObserve 用于被动接收与 Conversation 相关的 event。

3.5.1 ConversationService

ConversationService 包括如下 API：

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 创建 Conversation

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

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)

创建一个 P2P Conversation。

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

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))

创建一个 Group Conversation。

3.5.1.2 删除 Conversation

- API: void removeConversation(String convId)

Parameters	
convId	String: The conversationId for removing the Conversation.

根据 conversationId 删除 Conversation。

- API: void removeAllConversations()

删除所有的 Conversation。

3.5.1.3 查询 Conversation

- API: hasConversation(String convId)

Parameters	
convId	String: The conversationId to query.

Returns	
boolean	Returns whether the Conversation exists by the given conversationId

通过 conversationId 查询 Conversation 是否存在。

- API: List<Contact> queryAllConversations()

Returns	
List<Conversation>	Returns the Conversation list

获取 Conversation List。

- API: List<Conversation> queryConverstaionBy(ConversationType

conversationType)

Parameters	
conversationType	ConversationType: The ConversationType to query.

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

查询满足 conversationType 条件的 Conversation 列表。

- API: Conversation findP2PConversationBy(Contact peer)

Parameters	
peer	Contact: The Contact to query.

Returns	
Conversation	Returns the P2P conversation which has the queried Contact

通过 peer Contact 查询其 P2P Conversation。

- API: Conversation findConversationBy(String convId)

Parameters	
convId	String: The conversationId to query.

Returns	
Conversation	Returns the P2P conversation by the given conversationId

通过 conversationId 查询 Conversation。

3.5.2 ConversationServiceObserve API

ConversationServiceObserve 包括如下 API：

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

3.5.2.1 监视 Conversation 变化事件

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

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

注册 Observer<Void>，接收 Conversation 变化事件。

3.6 Message Service API

Message Service API 用于 Message 相关的操作。Message Service 必须在 SDK 绑定 Account 后才能使用。

- Message Service API 所在包：cn.meshee.fclib.api.message。
- Message Service 包含 FcMessageBuilder 、 MessageService 与 MessageServiceObserve。FcMessageBuilder 用于创建 message（包括 text，image，用于文件发送的文件消息，自定义消息等）， MessageService 用于主动进行 Message 操作，MessageServiceObserve 用于被动接收与 Message 相关的 event。通过 MessageService 发送的消息在 Meshee 网络中作为一个 packet 进行发送。

3.6.1 FcMessageBuilder

FcMessageBuilder 包括如下 API：

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 创建 FcMessage

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

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

Returns	
FcMessage	Returns the created FcMessage

创建一条文本 FcMessage。创建一条文本 FcMessage 时 Meshee SDK 会生成独一无二的 messageId。

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

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

创建一条图片 FcMessage。在图片 FcMessage 中可以存在文本消息。创建一条图片 FcMessage 时 Meshee SDK 会生成独一无二的 messageId。

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

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

Returns	
FcMessage	Returns the created FcMessage

创建一条文件 FcMessage。该 FcMessage 用于发送文件钱交互用的文件消息。创建一条文件 FcMessage 时 Meshee SDK 会生成独一无二的 messageId。

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

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

Returns	
FcMessage	Returns the created FcMessage

创建一条自定义 FcMessage。该 FcMessage 可以用于发送自定义的消息。

3.6.1.2 设置 FcMessage

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

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

Returns	
FcMessage	Returns the FcMessage after setting

设置 FcMessage 中的自定义 Object。

3.6.2 MessageService

MessageService 包括如下 API：

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

3.6.2.1 发送 FcMessage

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

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

发送一条 FcMessage。requestCallback 可以得到消息发送成功、失败、异常等回掉结果。

3.6.3 MessageServiceObserve

MessageServiceObserve 包括如下 API：

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

3.6.3.1 监视 FcMessage 收到事件

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

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)

注册 Observer<List<FcMessage>>，接收收到的 income FcMessage 列表事件。

3.7 File Service API

File Service API 用于 File 相关的操作。File Service 必须在 SDK 绑定 Account 后才能使用。

- File Service API 所在包：cn.meshee.fclib.api.file。
- File Service 包含 FileService 与 FileServiceObserve。FileService 用于主动进行 File 操作，FileServiceObserve 用于被动接收与 File 相关的 event。

3.7.1 FileService

FileService 包括如下 API：

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

3.7.1.1 发送文件消息

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

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

文件发送侧发送某个文件的 FcMessage。ackTimeoutInSeconds 规定了从发送 FcMessage 开始收到 ack 的超时时间。只有当文件发送测收到 ack 未超时时，才开始发

送文件，否则不发送文件。

3.7.1.2 回复文件消息

- API: void ackReceiveFile(FcMessage message)

Parameters	
message	message: The received FcMessage to ack

文件接收侧 ack 文件发送侧调用 requestSendFile API 发送的 FcMessage 消息。

3.7.2 FileServiceObserve

FileServiceObserve 包括如下 API：

void	observeFileDownloadProgress(Observer<FileProgress> observer, boolean register)
void	observeFileReceiveEvent(Observer<FileMessage> observer, boolean register)
void	observeFileSendEvent(Observer<IOException> observer, boolean register)

3.7.2.1 监视 File 接收进度事件

- API: void observeFileDownloadProgress(Observer<FileProgress> observer,

  boolean register)

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)

注册 Observer<FileProgress>，接收 FileProgess 变化事件。

3.7.2.2 监视 File 接收错误事件

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

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)

注册 Observer<FileMessage>，接收在接收文件时的错误事件，主要是 ack 文件消息的 timeout 事件。

3.7.2.3 监视 File 发送异常事件

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

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)

注册 Observer<IOException>，接收在发送文件时的错误事件。

3.8 Live Service API

Live Service API 用于 Live stream 分享相关的操作。Live Service 必须在 SDK 绑定 Account 后才能使用。

- Live Service API API 所在包：cn.meshee.fclib.api.live。
- Live Service 包含 LiveService 与 LiveServiceObserve。LiveService 用于主动进

行 Live stream 操作，LiveServiceObserve 用于被动接收与 live stream 相关的 event。

3.8.1 LiveService

LiveService 包括如下 API：

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

3.8.1.1 获取 Local Live proxy server port

- API: int getLocalLiveProxyServerPort()

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

获取本地 http 代理端口拉取 remote cloud 的 Live stream。如需使用该 proxy server，代理配置成 local IP“127.0.0.1”与 getLocalLiveProxyServerPort()所对应的 port 号。使用该本地代理时，如果 WiFi Interface 被 Meshee 网络使用，Meshee SDK 会智能选择 mobile network Interface 进行 cloud 拉流。

3.8.1.2 查询 Meshee 网络的 Live 资源

- API: List<String> queryMeshLiveResources()

Returns	
List<String>	Returns the Mesh live resource list

获取 Mesh live resource 列表。列表中的每个 resource 都可用于从 Meshee 网络内部拉取 Live stream。

3.8.1.3 设置 Cloud 拉流的 host

- API: void setRemoteLiveHost (String host)

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

设置用于从 cloud 拉取 Live stream 的 remote host。

3.8.2 LiveServiceObserve

LiveServiceObserve 包括如下 API：

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

3.8.2.1 监视 Meshee 网络 Live 资源变化事件

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

  boolean register)

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)

注册 Observer<Void>，接收 Meshee 网络内 Live stream 资源的变化事件。

3.8.2.2 监视 Remote Live Host 异常事件

- API: void observeRemoteLiveHostExceptionEvent(Observer<IOException>

  observer, boolean register)

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)

注册 Observer<IOException>，接收在从 cloud 拉取 Live stream 时的 remote host 异常。

3.9 Avatar Service API

Avatar Service API 用于 Avatar 相关的操作。Avatar Service 必须在 SDK 绑定 Account 后才能使用。

- Avatar Service API 所在包：cn.meshee.fclib.api.avatar。
- Avatar Service 包含 AvatarService 与 AvatarServiceObserve。AvatarService 用于主动进行 Avatar 操作，AvatarServiceObserve 用于被动接收与 Avatar 相关的 event。

3.9.1 AvatarService

AvatarService 包括如下 API：

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

3.9.1.1 保存 Account Avatar

- API: Avatar saveMyAvatar(String avatarPath)

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

Returns	
Avatar	Returns the saved Avatar

永久保存 Avatar path 对应的图片到 Account Avatar。Meshee SDK 退出不会删除调用该 API 保存的 Avatar。已保存的 Account Avatar 在绑定相应的 Account Contact 时读取。

3.9.1.2 推送 Account Avatar

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

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

推送 Account Avatar 到 Contac 列表中的所有 Contact。在推送 Avatar 的同时，Contact 的 nickname 也会一起推送。

- API: void pushAvatarToAll()

推送 Account Avatar 到所有与 Account Contact 连接的 Contact 列表。在推送 Avatar 的同时，Contact 的 nickname 也会一起推送。

3.9.1.3 查询 Contact Avatar Uri

- API: Uri getAvatarUri(String contactUuId)

Parameters	
contactUuId	String: The contactUuId to query.

Returns	
Uri	Returns the Avatar Uri by the given contactUuId

通过 contactUuId 查询 Contact 的 Avatar Uri。

3.9.2 AvatarServiceObserve

AvatarServiceObserve 包括如下 API：

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

3.9.2.1 监视 Contact Avatar 变化事件

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

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

注册 Observer<Avatar>，接收 Contact 的 Avatar 变化事件。

3.9.2.2 监视 Account Avatar 推送异常事件

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

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)

注册 Observer<IOException>，接收在推送 Account Avatar 时的异常。Avatar 图片大小不能超过 512K bytes，否则上报异常事件。

4 Meshee Model

Meshee model 是 Meshee 的重要数据模型，被 Meshee API 使用。Meshee model 包括：

- Meshee SDK 管理 model，用于初始化 SDK、退出 SDK、获取 SDK service 与管理 SDK Account。
- Setting Service model，用于获取与配置基本的 SDK setting。
- Network Service model，用于 Meshee 网络相关的操作。
- Contact Service model，用于 Contact 相关的操作。
- Conversation Service model，用于 Conversation 相关的操作。
- Message Service model，用于 Message 相关的操作。
- Live Service model，用于 Live stream 分享相关的操作。
- File Service model，用于 File 相关的操作。
- Avatar Service model，用于 Avatar 相关的操作。

4.1 Meshee SDK 管理 Model

Meshee SDK 管理 model，用于初始化 SDK、退出 SDK、获取 SDK service 与管理 SDK Account。

- Meshee SDK 管理 Model 所在包：cn.meshee.fclib.api.model。
- Meshee SDK 管理 Model 包括 NetworkStrategyType 与 SdkOptions。

4.1.1 NetworkStrategyType

- Model: NetworkStrategyType 包括：

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

NetworkStrategyType 为 SDK 将支持的网络模式：NetworkStrategyType1 与 NetworkStrategyType2。NetworkStrategyType1 是一个 Meshee 网络覆盖范围最大 400 米，终端数量最多~100 台；NetworkStrategyType2 是一个 Meshee 覆盖范围最大 200 米，终端数量最多~10 台。NetworkStrategyType1 的优势在于网络覆盖范围大，终端数量多，通信距离远。NetworkStrategyType2 的优势在于，任意两个终端之间的 Peer-to-Peer 平均传输速度更快，平均传输时延更小。

4.1.2 SdkOptions

- Model: SdkOptions 包括：

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 在初始化 SDK 时使用。SDK 初始化时可以定制 SdkStorageRootPath 与 NetworkStrategyType。

4.2 Setting Service Model

Setting Service Model 用于获取与配置基本的 SDK setting。暂无。

4.3 Network Service Model

Network Service model 用于 Meshee 网络相关的操作。

- Network Service Model 所在包：cn.meshee.fclib.api.network.model。
- Network Service Model 包括 Role、RoleEvent、NetworPoint、NetworkEvent 与 ConnectionState。

4.3.1 Role

- Model: Role 包括：

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.)

网络角色的 model。有三种网络角色：Idle， Owner 与 Member。当终端不处于任何 Meshee 网络中时，终端处于 Idle 角色；当终端自身创建 Meshee 网络，可作为 NetworkPoint 可以被其它终端连接时，终端处于 Owner 角色；当终端通过连接其它 NetworkPoint 已经加入一个 Meshee 网络时，终端处于 Member 角色。

4.3.2 RoleEvent

- Model: RoleEvent 包括：

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

网络角色发生变化的事件的 model。事件中包含该事件发生前的网络角色与该事件发生后的网络角色。

4.3.3 NetworkPoint

- Model: NetworkPoint 包括：

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 是 Meshee 网络接入点的 model。它包括该接入点的 networkPoint Id，所在 group 的 groupId，所在 group 的 groupName，信号强度以及网络角色。

4.3.4 NetworkEvent

- Model: NetworkPoint 包括：

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 是网络创建或销毁的事件 model。网络的销毁只能由网络角色为 Owner 的终端完成。NetworkEvent 包含网络事件类型，网络创建或销毁的 networkPoint 的 networkPointId，networkPoint 所创建的 groupId 与 groupName，以及网络创建或销毁是否成功。

- Model: NetworkEventType 包括：

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

NetworkEvent 的内嵌 model NetworkEventType，包含网络创建事件与网络销毁事件。

4.3.5 ConnectionState

- Model: ConnectionState 包括：

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 是终端连接 NetworkPoint 的状态事件 model。ConnectionState 包含连接的 networkPoint 与最新连接状态。

- Model: State 包括：

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 的内嵌 model State，包含连接 NetworkPoint 的 DISCONNECTED、CONNECTING、CONNECTED 与 FAILED 状态。其中 DISCONNECTED 状态也是连接一个 NetworkPoint 的初始状态。

4.4 Contact Service Model

Contact Service model 用于 Contact 相关的操作。

- Contact Service Model 所在包：cn.meshee.fclib.api.contact.model。
- Contact Service Model 包括 Contact 与 ContactException。

4.4.1 Contact

- Model: Contact 包括：

Fields

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

Contact model 包含 contactRawId，nickname 与 contactUuid，均可由 App 定制。其中 contactRawId 可以用作描述 App 的第三方接口账号，例如微信号，email 等，App 在生成 Contact 时需要保证 contactUuid 的唯一性。

4.5 Conversation Service Model

Conversation Service model 用于 Conversation 相关的操作。

- Conversation Service Model 所在包：cn.meshee.fclib.api.conversation.model。
- Conversation Service Model 包括 ConversationType 与 Convesation。

4.5.1 ConversationType

- Model: ConversationType 包括：

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

ConversationType model 包括三种 Type：NONE，P2P 与 GROUP。其中 P2P 表示点对点 conversation，GROUP 表示群组 conversation。GROUP conversation 暂不支持。NONE 为非法 conversation。

4.5.2 Conversation

- Model: Conversation 包括：

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 定义了一个 Conversation，包括 ConversationType，initiator （即 Conversation 的发起者），participant（及除 Conversation 包括的 self Contact 之外的所有 Contact，P2P conversation 只包括一个 participant），conversationId，conversationTitle 与 portraitUri（即 Conversation 的头像所在的 Uri)。

4.6 Message Service Model

Message Service model 用于 Message 相关的操作。

- Message Service Model 所在包：cn.meshee.fclib.api.message.model。
- Message Service Model 包括 ChatType、ChatTypeEnum、MessageType、MessageTypeEnum、MediaName 与 FcMessage。

4.6.1 ChatType

- Model: ChatType 包括：

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 定义了 SDK 所支持的聊天类型。ONE_ONE_CHAT 为点到点的两个 Contact 聊天类型，SYSTEM_MESSAGE 为系统消息生成的聊天类型，GROUP_CHAT 为至少两个 Contact 的 group 聊天类型，UNKNOWN 为未知聊天类型，APP 暂时不使用 FREE_CHAT 聊天类型。

4.6.2 ChatTypeEnum

- Model: ChatTypeEnum 包括：

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 为对应 ChatType constant 的 enum。

4.6.3 MessageType

- Model: MessageType 包括：

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 定义了 SDK 所支持的 Message 类型。

4.6.4 MessageTypeEnum

- Model: MessageTypeEnum 包括：

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 为对应 MessageType constant 的 enum。

4.6.5 MediaName

- Model: MediaName 包括：

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

MediaName 定义了 media 的名称格式。它包括 media 的名称与扩展名。

4.6.6 FcMessage

- Model: FcMessage 是 Meshee 网络发送接收消息的基本类型。它可以支持 MessagType 中的各种消息类型。

4.7 File Service Model

File Service API 用于 File 相关的操作。

- File Service Model 所在包：cn.meshee.fclib.api.file.model。
- File Service Model 包括 FileMessage、FileProgress、FileIdentity、FileEnvelope、FileLocalSummary、FTcpHandShake、FileStatus、FileAckStatus 与 FileReceiveStatus。

4.7.1 FTcpHandShake

- Model: FTcpHandShake 包括：

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

文件传输的 fTcp 握手阶段的 model，包括三个阶段：FTcpSync（即文件发送端发送了文件请求消息）、FTcpAck（即文件接受端 ack 文件请求消息，准备接收文件）与 FTcpError（在文件传输握手中发生了错误）。

4.7.2 FileIdentity

- Model: FileIdentity 包括：

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 用于描述与传输文件相关的各种标识信息，包括 fcMessageId（即与传输文件相关的 FcMessage 的 messageId）、conversationId（即与传输文件相关的 conversation 相关的 FcMessage 的 conversationId）、fileId（即传输文件的文件 Id）与 fileInstanceId（即传输文件的实例 Id）。

4.7.3 FileEnvelope

- Model: FileEnvelope 包括：

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

FileEnvelop model 用于描述与传输文件相关的收发 Contact 信息。

4.7.4 FileStatus

- Model: FileStatus 包括：

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

FileStatusmodel 用于描述与传输文件相关的状态信息，包括 fileAckStatus 与 fileReceiveStatus。

4.7.5 FileAckStatus

- Model: FileAckStatus 包括：

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

FileAckStatus 用于描述传输文件的 ack 状态，包括文件有效 ack 的超时时间、ack 失败的描述与 ackStatus。

- Model: AckStatus 包括：

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 的内嵌 model AckStatus，包含 ack 一个传输文件的状态，包括 Acked、Not_Acked 与 Ack_Failure。

4.7.6 FileReceiveStatus

- Model: FileReceiveStatus 包括：

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 用于描述传输文件的接收状态，包括文件 receiveStatus、receivePercentage 与接收失败的描述。

- Model: ReceiveStatus 包括：

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 的内嵌 model ReceiveStatus，包含接收一个文件的状态，包括 Not_Start、InProgress、Success 与 Failure。

4.7.7 FileLocalSummary

- Model: FileLocalSummary 包括：

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 用于描述文件的概述信息，包括文件的本地路径、名称与大小。

4.7.8 FileMessage

- Model: FileMessage 包括：

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 用于描述 File 传输中的各种信息: FileIdentity（即文件的标识信息）、FileEnvelope（即文件的收发 Contact 信息），FileLocalSummary（即文件的概述信息）、FileStatus（即文件的收发状态信息），FTcpHandShake（即文件传输的握手

信息）。

4.7.9 FileProgress

- Model: FileProgess 包括：

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

FileProgess model 用于描述 File 传输的进度信息: FileIdentity（即文件的标识信息）与进度百分比进度信息。

4.8 Live Service Model

Live Service Model 用于 Live stream 分享相关的操作。暂无。

4.9 Avatar Service Model

Avatar Service Model 用于 Avatar 相关的操作。

- Avatar Service Model 所在包：cn.meshee.fclib.api.avatar.model。
- Avatar Service Model 包括 Avatar。

4.9.1 Avatar

- Model: Avatar 包括：

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

Avatar model 用于描述某个 Contact 的头像信息，包括 contactUuid，avatarPath

与 avatar MD5。