ON SIGHT

# USER MANUAL

Onsight Embedded for Android SDK
Software Version 7.1.10

June 2016

LIBRESTREAM

# Table of Contents

Document Revision

Librestream

Onsight Connect for Windows Release Notes

Doc #: 400248-02

June 2016

Information in this document is subject to change without notice.

Reproduction in any manner whatsoever without the written permission of Librestream is strictly forbidden.

Name of Librestream Software Onsight Connect

Patents Notice: United States Patent # 7,221,386, together with additional patents pending in Canada, the United States and other countries, all of which are in the name of Librestream Technologies Inc.

# 1 Overview

Onsight Embedded powers real-time video collaboration on over 10 million smartphones, tablets and computers today. Enterprises have integrated Onsight Embedded into their own uniquely branded and customized mobile apps, adding the value of live video collaboration to an existing workflow.

Backed by a robust and secure enterprise-grade infrastructure, advanced collaboration capabilities, and a unique visual experience, Onsight Embedded allows you to integrate this ability to 'collaborate on things' visually in the field, taking days out of service delivery and opening up new sources of revenue for enterprises. Key benefits include:

- Accessing Central Support Teams
- Interacting Directly with Customers
- Mentoring Technicians in the Field

The Onsight Embedded use case is designed to minimize training requirements for the field user by allowing the expert access to the field camera functions remotely. For example, the expert can turn on the light, pause video, take pictures, and zoom the camera in the field. Once the session is established between the expert and the remote user, the expert controls the video session.

## 1.1 Onsight Features

- Live Audio/Video
- Freeze Frame
- Telestration
- HD Image Capture
- Media File sharing
- Quick Search
- Session Recording
- Remote Camera Control
- Guest Invites
- Multiple Participants
- Call Continuity
- Enterprise Grade Security/Controls
- Low Bandwidth Optimization
- Bandwidth Adaptive Streaming
- Centralized Management
- Firewall Traversal
- Integrates with existing video systems

## 1.2 IDE Requirements

Onsight Embedded for Android SDK is built for Eclipse with the ADT plugin. It is also compatible with Android Studio, please contact Librestream for details.

## 1.3 Target Equipment

Onsight Embedded for Android SDK is intended for application development for Android v4.1 (Jelly Bean) or higher on the following devices.

### 1.3.1 Android Smartphones

- Samsung Galaxy S7, S6, S5, S4, S3, S2
- LG G2
- Motorola Moto X
- HTC One M7
- Nexus 5, 7
- Sony Xperia Tablet Z

# 2 OclInterface

The Onsight Embedded Library (also referred to as Onsight Connect Library (OCL)) provides basic Onsight Connect app functionality that can be embedded into third party applications.

The API is a Java interface layer that gives enough functionality for the Host Application (HostApp) to invoke and interact with the embedded OCL. The HostApp uses the OclInterfaceFactory to retrieve an instance of an object implementing OclInterface, and all communication with the OCL happens through it.

## 2.1 OclInterface Enums

The following enums are used by OclInterface.

### 2.1.1 EOclSipRegistrationState

The EOclSipRegistrationState identifies the Session Initiation Protocol (SIP) registration state.

#### *2.1.1.1 OCL_SIP_REGISTERING*

The SIP registration process starts automatically after the OCL reaches the OCL_LOGGED_IN state. The OCL_SIP_REGISTERING state indicates that the SIP registration process has begun.

#### *2.1.1.2 OCL_SIP_REGISTERED*

The OCL_SIP_REGISTERED state is the end goal of the SIP registration process. It marks the point at which the SIP VOIP call can be made.

#### *2.1.1.3 OCL_SIP_REGISTER_FAILED*

The SIP Registration process has failed. Likely causes of this state are that the SIP registration parameters were invalid, or the SIP registrar could not be reached.

#### *2.1.1.4 OCL_SIP_REGISTER_FAILED_RETRYING*

The SIP Registration has failed, but the OCL is still trying to SIP register.

#### *2.1.1.5 OCL_SIP_UNREGISTERED*

SIP registration has moved to the OCL_SIP_UNREGISTERED state. This can occur in normal operation when the SIP call is complete and the endpoint no longer needs to be SIP registered. The state can also be reached if the SIP registration process has failed and has given up trying to contact the SIP registrar.

### 2.1.2 EOclCallState

The EOclCallState enum identifies the SIP call states.

#### *2.1.2.1 OCL_CALL_DISCONNECTED*

The default state where there are no active, or pending incoming or outgoing calls. After a call is hung up and the call is fully cleaned up, the call status moves to OCL_CALL_DISCONNECTED.

#### *2.1.2.2 OCL_CALL_INCOMING*

The OCL_CALL_INCOMING state signals that there is an incoming call. The HostApp would typically play a ring and present some UI to the user asking if the user wants to accept or decline the incoming call. When the user decides to accept the call, the acceptCall method is used. If the user decides to decline the call, the HostApp would call the endCall method.

#### *2.1.2.3 OCL_CALL_IN_PROGRESS*

The OCL_CALL_IN_PROGRESS call state indicates that an outgoing call is being made.

#### 2.1.2.4 OCL_CALL_ANSWERING

The OCL_CALL_ANSWERING state means that an outgoing call has been accepted by the remote endpoint and is in the process of being connected.

#### 2.1.2.5 OCL_CALL_CONNECTED

The OCL_CALL_CONNECTED state indicates that an outgoing or incoming call is fully connected, and the two participants are able to communicate with each other.

#### 2.1.2.6 OCL_CALL_DISCONNECTING

The OCL_CALL_DISCONNECTING state is entered when either endpoint has initiated an end to the SIP call. While in this state, the call is in the process of being shut down and cleaned up. This state is short-lived and when the call is completely shut down, the call state quickly transitions to the OCL_CALL_DISCONNECTED state.

### 2.1.3 EOclVideoState

The EOclVideoState enum represents the state of the video stream. While the state is OCL_VIDEO_STARTED, video is being sent from one endpoint to the other. While in the OCL_VIDEO_STOPPED state, there is NO video being sent. The other two states indicate that the OCL is transitioning into the corresponding states.

#### 2.1.3.1 OCL_VIDEO_START_REQUESTED

The video stream is about to be started.

#### 2.1.3.2 OCL_VIDEO_STARTED

The video stream is up and running.

#### 2.1.3.3 OCL_VIDEO_STOP_REQUESTED

The video stream is about to be stopped.

#### 2.1.3.4 OCL_VIDEO_STOPPED

The video stream has been stopped.

#### 2.1.3.5 OCL_VIDEO_ERROR

The video stream encountered an error during starting or stopping.

### 2.1.4 EOclReturnValue

The following are possible return values from the OCL.

#### 2.1.4.1 OCL_SUCCESS

The method call was successful.

#### 2.1.4.2 OCL_NOT_INITIALIZED

The OCL library was not initialized and could not handle the method call. Call the initialize method first.

#### 2.1.4.3 OCL_NOT_SIP_REGISTERED

The method was not allowed because SIP registration was not successful or the current SIP registration is in progress and has not yet reached the OCL_SIP_REGISTERED state.

#### 2.1.4.4 OCL_NOT_ALLOWED

The method call was not allowed at this time. For example, the login method is called when the user has already logged in.

#### 2.1.4.5 OCL_INVALID_PARAMETER

The method call was not allowed because one of the parameters was invalid.

#### 2.1.4.6 OCL_NOT_SIP_UNREGISTERED

The method call was not allowed because the OCL SIP registration state needs to be OCL_SIP_UNREGISTERED before the method can be used.

#### 2.1.4.7 OCL_INTERNAL_ERROR

The method call was not allowed because of an internal system error.

#### *2.1.4.8 OCL_NOT_SUPPORTED*

The OCL library was not initialized because one or more features are not available on the platform.

#### *2.1.4.9 OCL_MISSING_PERMISSION*

The requested operation requires a permission or feature that is not available or enabled.

#### *2.1.4.10 OCL_REMOTE_CAPTURE_NOT_ALLOWED*

If the HostApp is part of an Onsight conference and a remote endpoint is the live video source, requesting an image capture from the remote live video source is not allowed.

### 2.1.5 EOclInitId

The OCL can be in the following three states:

#### *2.1.5.1 OCL_INIT_ID_INITIALIZED*

The OCL is initialized.

#### *2.1.5.2 OCL_INIT_ID_DEINITIALIZED*

The OCL is not initialized.

#### *2.1.5.3 OCL_INIT_ID_FAILURE*

The OCL initialization attempt failed.

### 2.1.6 EOclErrorId

The OCL errors that could occur:

#### *2.1.6.1 OCL_ERROR_ID_NETWORK_NOT_FOUND*

If activities to access the network fail to find a network, this error is produced.

#### *2.1.6.2 OCL_ERROR_ID_NETWORK_ERROR*

If OCL activities using the network fail, this error is produced.

#### *2.1.6.3 OCL_ERROR_ID_AUDIO_INIT_FAILURE*

If an Onsight call connects but the OCL cannot access the audio hardware of the phone, this error is passed to the HostApp via OclErrorNotification. The most likely cause of this error is when an Onsight call connects while a cell phone call is using the audio hardware. As part of **Onsight's** Call Continuity feature, an Onsight call can connect without audio with the assumption that the cell phone and Onsight calls are between the same participants, with the voice communication happening over the cell phone call and the video communication happening over the Onsight call.

#### *2.1.6.4 OCL_ERROR_ID_CAMERA_INIT_FAILURE*

There was a problem initializing the camera, most likely due to the camera being in use by another application. If you choose to continue the call without video, the OCL will periodically attempt to acquire access to camera.

#### *2.1.6.5 OCL_ERROR_ID_CAMERA_RESTORED*

If a prior attempt to access the camera failed and an OCL_ERROR_ID_CAMERA_INIT_FAILURE notification was received, this notification indicates that camera access was subsequently acquired and video should now be available to the OCL.

#### *2.1.6.6 OCL_ERROR_ID_AUDIO_INTERRUPT_BEGAN*

If an Onsight call (including voice communication) is in progress and a cell phone call is made or received, OCL sends the OCL_ERROR_ID_AUDIO_INTERRUPT_BEGAN event to inform the HostApp that the **Onsight's** use of the audio hardware has been interrupted by the cell phone call, since cell phone calls have priority access to the audio hardware. At this point, the Onsight call has no voice communication, but stays connected for continued live video communication or image sharing. This is part of **Onsight's** Call Continuity feature.

#### *2.1.6.7 OCL_ERROR_ID_AUDIO_INTERRUPT_END*

If an Onsight call has had its voice communication interrupted by a cell phone call (see previous section), once the cell phone call disconnects, the OCL_ERROR_ID_AUDIO_INTERRUPT_END event is used to inform the HostApp that voice communication has been restored to the Onsight call. This is part of **Onsight's** Call Continuity feature.

#### *2.1.6.8 OCL_ERROR_ID_IN_BACKGROUND_CALL_REMINDER*

Unused in Android.

#### 2.1.6.9 OCL_ERROR_ID_IN_AUDIO_INIT_SUCCESS

If an Onsight call is established while a cell phone call is connected, the audio hardware will be inaccessible and the HostApp will receive the OCL_ERROR_ID_AUDIO_INIT_FAILURE event from OCL. If during the Onsight call, the cell phone call disconnects, the OCL attempts to gain access to the audio hardware again and if successful, informs the HostApp via the OCL_ERROR_ID_AUDIO_INIT_SUCCESS event. At this point, the Onsight call supports voice communication. This is part of **Onsight's** Call Continuity feature.

### 2.1.7 EOclEndCallReason

When an OclCallStateNotification::OCL_CALL_DISCONNECTED notification is received, the disconnect message contains a parameter that specifies the reason why the call ended.

#### 2.1.7.1 OCL_END_CALL_UNKNOWN

The reason for the call disconnect could not be determined.

#### 2.1.7.2 OCL_END_CALL_NORMAL

The call disconnected because either side hung up the call, which is the normal ending to a call.

#### 2.1.7.3 OCL_END_CALL_NOT_AVAILABLE

The remote participant was not available.

#### 2.1.7.4 OCL_END_CALL_BUSY

The remote participant was already in a call.

#### 2.1.7.5 OCL_END_CALL_DECLINED

The remote participant declined the call.

#### 2.1.7.6 OCL_END_CALL_TIMEOUT

The call to the remote endpoint timed out waiting to connect.

#### 2.1.7.7 OCL_END_CALL_SHUTTING_DOWN

The call was disconnected because the OCL is shutting down.

#### 2.1.7.8 OCL_END_CALL_NOT_SUPPORTED

The call could not be connected because the call parameters did not match.

#### 2.1.7.9 OCL_END_CALL_NOT_REGISTERED

The call could not be connected because the remote participant is not SIP registered.

#### 2.1.7.10 OCL_END_CALL_NETWORK_ERROR

The call had been connected but disconnected because the two participants lost network connectivity. This is the error that the HostApp may want to look for to trigger an attempt to reconnect.

### 2.1.8 EOclLoginState

The EOclLoginState enum represents the state of logging into **Librestream's Onsight Account Manager (OAM)** server.

#### 2.1.8.1 OCL_NOT_LOGGED_IN

The OCL_NOT_LOGGED_IN state indicates that the OCL is not involved in any login activity. This is the state when the OCL is initialized, when an OAM login attempt has failed, and after a logout has completed.

#### 2.1.8.2 OCL_LOGGING_IN

The OCL_LOGGING_IN state indicates that a login has been initiated but has not completed.

#### 2.1.8.3 OCL_LOGGED_IN

The OCL_LOGGED_IN state indicates that a login attempt has been successful. At this point, the OCL will automatically use the logged in user's SIP credentials received from the OAM server to attempt to SIP register. The progress of this SIP registration can be tracked using the OclSipRegistrationStateNotification.

### 2.1.9 EOclLoginReason

The EOclLoginReason enum provides extra information to the OclLoginStateNotification. The reason is typically used to help understand why a login attempt did not succeed. When a login is attempted, the login state moves from OCL_NOT_LOGGED_IN to OCL_LOGGING_IN. Then if the login fails, the login state moves from OCL_LOGGING_IN back to OCL_NOT_LOGGED_IN. Accompanying this login state notification is an EOclLoginReason parameter that gives an indication of why the failure happened. There are many values in the enum and their names are typically self explanatory. When a login failure reason shows up frequently, it can be used to help debug the failure with Librestream customer support.

### 2.1.10 EOclBandwidthAdaptiveStreaming

The EOclBandwidthAdaptiveStreaming enum provides the values to control the Bandwidth Adaptive Streaming (BAS) feature of the library. It contains the following values:

#### *2.1.10.1 OCL_BAS_DISABLED*

The OCL_BAS_DISABLED value indicates that BAS is disabled.

#### *2.1.10.2 OCL_BAS_ENABLED*

The OCL_BAS_ENABLED value indicates that BAS is enabled for both the WiFi and cellular data interfaces.

#### *2.1.10.3 OCL_BAS_ENABLED_FOR_CELLULAR_ONLY*

The OCL_BAS_ENABLED_FOR_CELLULAR_ONLY value indicates that BAS is enabled only for the cellular data interface.

#### *2.1.10.4 OCL_BAS_USE_DEFAULT*

The OCL_BAS_USE_DEFAULT value is used by the HostApp to tell the OCL to use the default value for BAS. If the internal BAS setting is changed as a result, the HostApp is notified of the change via the OclSettingChangedNotification notification.

### 2.1.11 EOclCaptureState

The EOclCaptureState enum represents the state of an image capture. The first three states refer to the actual image capture, while the last value may occur if the OCL cannot write the captured image into the device’s camera album when the saveImagesToCameraAlbum property is set to true.

The enum contains the following values:

#### *2.1.11.1 OCL_CAPTURE_START*

The OCL_CAPTURE_START value indicates that an image capture has been started.

#### *2.1.11.2 OCL_CAPTURE_COMPLETE*

The OCL_CAPTURE_COMPLETE value indicates that an image capture into the app’s sandbox has successfully completed.

#### *2.1.11.3 OCL_CAPTURE_ERROR*

The OCL_CAPTURE_ERROR value indicates that an image capture has failed.

#### *2.1.11.4 OCL_CAPTURE_CAMERA_ALBUM_ERROR*

The OCL_CAPTURE_CAMERA_ALBUM_ERROR value indicates that, after the capture completed, the attempt to write the image to the camera album failed.

### 2.1.12 EOclCaptureStatusCode

The EOclCaptureStatusCode enum represents extra information related to a still image capture event. It contains the following values:

#### *2.1.12.1 OCL_CAPTURE_NONE*

The OCL_CAPTURE_NONE value indicates that there is no additional information.

#### *2.1.12.2 OCL_CAPTURE_LOCATION_LOCAL*

The OCL_CAPTURE_LOCATION_LOCAL value indicates that the still image capture was initiated locally.

#### *2.1.12.3 OCL_CAPTURE_LOCATION_REMOTE*

The OCL_CAPTURE_LOCATION_REMOTE value indicates that the still image capture was initiated remotely.

#### *2.1.12.4 OCL_CAPTURE_ERROR_SANDBOX_UNKNOWN*

The OCL_CAPTURE_ERROR_SANDBOX_UNKNOWN value indicates that the capture failed because of a general failure.

#### *2.1.12.5 OCL_CAPTURE_ERROR_SANDBOX_CAMERA_FAILURE*

The OCL_CAPTURE_ERROR_SANDBOX_CAMERA_FAILURE value indicates that the capture failed because the camera device had issues. An example of this would be if the capture was attempted when the device was in the background and the app did not have access to the camera hardware.

#### 2.1.12.6 OCL_CAPTURE_ERROR_SANDBOX_WRITE_FAILURE

The OCL_CAPTURE_ERROR_SANDBOX_WRITE_FAILURE value indicates the capture failed to write the still image to the file system. Possible causes might be that the file system is full, the app cannot write to the disk, or the disk file system is corrupt.

#### 2.1.12.7 OCL_CAPTURE_ERROR_CAMERA_ALBUM_UNKNOWN

The OCL_CAPTURE_ERROR_CAMERA_ALBUM_UNKNOWN value indicates that the captured still image could not be written to the camera album for an unknown reason.

#### 2.1.12.8 OCL_CAPTURE_ERROR_CAMERA_ALBUM_OUT_OF_SPACE

The OCL_CAPTURE_ERROR_CAMERA_ALBUM_OUT_OF_SPACE value indicates that the capture still image could not be written to the camera album because there is no available disk space.

#### 2.1.12.9 OCL_CAPTURE_ERROR_CAMERA_ALBUM_NOT_AUTHORIZED

The OCL_CAPTURE_ERROR_CAMERA_ALBUM_NOT_AUTHORIZED value indicates that the captured still image could not be written to the camera album because the app does not have authorization to access the camera album.

#### 2.1.12.10 OCL_CAPTURE_ERROR_CAMERA_ALBUM_WRITE_FAILURE

The OCL_CAPTURE_ERROR_CAMERA_ALBUM_WRITE_FAILURE value indicates that the captured still image could not be written to the camera album because there was a write failure.

### 2.1.13 EOclSystemHealth

The EOclSystemHealth enum represents the state of an individual system health item. It contains the following values:

#### 2.1.13.1 OCL_SYSTEM_HEALTH_NONE

The OCL_SYSTEM_HEALTH_NONE value indicates that there is no system health information available for the item (i.e. it’s not available or is disabled).

#### 2.1.13.2 OCL_SYSTEM_HEALTH_GOOD

The OCL_SYSTEM_HEALTH_GOOD value indicates that the system health item is good condition (connected, registered, logged in, etc.).

#### 2.1.13.3 OCL_SYSTEM_HEALTH_CONNECTING

The OCL_SYSTEM_HEALTH_CONNECTING value indicates that the system health item is in the process of connecting (to the network, to a server, etc.).

#### 2.1.13.4 OCL_SYSTEM_HEALTH_BAD

The OCL_SYSTEM_HEALTH_BAD value indicates that the system health item is in bad condition. This generally means that an attempt to configure the item has taken place and failed.

### 2.1.14 EOclInviteGuestResult

The EOclInviteGuestResult enum represents the result of a guest invite request. It contains the following values:

#### 2.1.14.1 OCL_INVITE_INTERNAL_ERROR

The OCL_INVITE_INTERNAL_ERROR value indicates that there is an internal error.

#### 2.1.14.2 OCL_INVITE_SUCCESS

The OCL_INVITE_SUCCESS value indicates that the OAM server has successfully sent an email to a prospective caller.

#### 2.1.14.3 OCL_INVITE_EMAIL_ALREADY_EXISTS

The OCL_INVITE_EMAIL_ALREADY_EXISTS value indicates that the email address is already registered.

#### 2.1.14.4 OCL_INVITE_NO_USER_LICENSES_AVAILABLE

The OCL_INVITE_NO_USER_LICENSES_AVAILABLE value indicates that there are no user licenses available to invite a guest.

#### 2.1.14.5 OCL_INVITE_SIP_ACCOUNT_POOL_EMPTY

The OCL_INVITE_SIP_ACCOUNT_POOL_EMPTY value indicates that the SIP account pool is empty.

#### 2.1.14.6 OCL_INVITE_INVALID_EXPIRY_TIME

The OCL_INVITE_INVALID_EXPIRY_TIME value indicates that the expiry time is invalid.

#### *2.1.14.7 OCL_INVITE_INVITES_NOT_ALLOWED*

The OCL_INVITE_INVITES_NOT_ALLOWED value indicates that the guest invitation is not allowed in the OAM domain.

#### *2.1.14.8 OCL_INVITE_SIP_SERVER_TEMPORARILY_UNAVAILABLE*

The OCL_INVITE_SIP_SERVER_TEMPORARILY_UNAVAILABLE value indicates that the SIP server is temporarily unavailable.

#### *2.1.14.9 OCL_INVITE_SMS_INVITES_NOT_ALLOWED*

The OCL_INVITE_SMS_INVITES_NOT_ALLOWED value indicates that SMS guest invitations are not allowed in the OAM domain.

#### *2.1.14.10 OCL_INVITE_INVALID_PHONE_NUMBER*

The OCL_INVITE_INVALID_PHONE_NUMBER value indicates the phone number is invalid.

#### *2.1.14.11 OC_INVITE_MAX_MESSAGE_LENGTH_EXCEEDED*

The OCL_INVITE_MAX_MESSAGE_LENGTH_EXCEEDED value indicates that message to the guest user is too long.

### 2.1.15 EOclChangePasswordResult

The EOclChangePasswordResult enum represents the result of a change password request. It contains the following values:

#### *2.1.15.1 OCL_CHANGE_PASSWORD_INTERNAL_ERROR*

The OCL_CHANGE_PASSWORD_INTERNAL_ERROR value indicates that there is an internal error.

#### *2.1.15.2 OCL_CHANGE_PASSWORD_SUCCESS*

The OCL_CHANGE_PASSWORD_SUCCESS value indicates that the OAM server has successfully changed the password for the OAM user.

#### *2.1.15.3 OCL_CHANGE_PASSWORD_POLICY_VIOLATION*

The OCL_CHANGE_PASSWORD_POLICY_VIOLATION value indicates that the new password violates the password policy of the OAM domain.

#### *2.1.15.4 OCL_CHANGE_PASSWORD_USER_DOES_NOT_EXIST*

The OCL_CHANGE_PASSWORD_USER_DOES_NOT_EXIST value indicates that the username does not exist in the OAM domain.

#### *2.1.15.5 OCL_CHANGE_PASSWORD_INVALID_RESET_PASSWORD_CODE*

The OCL_CHANGE_PASSWORD_INVALID_RESET_PASSWORD_CODE value indicates that the reset password code is invalid.

#### *2.1.15.6 OCL_CHANGE_PASSWORD_OLD_PASSWORD_INCORRECT*

The OCL_CHANGE_PASSWORD_OLD_PASSWORD_INCORRECT value indicates that the old password provided by the user is incorrect.

#### *2.1.15.7 OCL_CHANGE_PASSWORD_PASSWORD_ALREADY_USED*

The OCL_CHANGE_PASSWORD_PASSWORD_ALREADY_USED value indicates that the new password provided by the user is already used.

#### *2.1.15.8 OCL_CHANGE_PASSWORD_ACCOUNT_LOCKED*

The OCL_CHANGE_PASSWORD_ACCOUNT_LOCKED value indicates that the OAM account is locked.

### 2.1.16 EOclIntentActivityRegistrationType

The EOclIntentActivityRegistrationType is used with the registerIntentActivity method to identify which OCL generated Intent for which the HostApp wants to register an Activity.

#### *2.1.16.1 OCL_INCOMING_CALL*

When the HostApp has no OclEventListener objects registered to receive OCL events (via the initialize or addOclEventListener methods) and there is an incoming call, the HostApp can have an Intent sent to a given Activity for the incoming call by registering the Activity via registerIntentActivity using the OCL_INCOMING_CALL enum. If there is an OclEventListener object registered, it will receive the incoming call notification (see OclCallStateNotification) and the OCL_INCOMING_CALL Intent will NOT be sent.

When an Activity launches and initializes OCL, it typically passes itself as the OCL event listener. As long as the Activity is not destroyed, it will receive incoming Onsight call events via the OclCallStateNotification. If the Activity is destroyed and there are no other OCL event listeners to receive the incoming call event, the Activity registered to receive incoming call Intents will receive an incoming call Intent. The Action of the intent is set to Intent.ACTION_ANSWER and the Intent's extended data will contain the incoming call handle, identified by the OCL_CALL_HANDLE_KEY. A common setup would be for the HostApp to create an IncomingCall Activity with Accept/Decline buttons that would handle the incoming call Intent.

#### 2.1.16.2 OCL_SERVICE_NOTIFICATION

The HostApp uses the OCL_SERVICE_NOTIFICATION enum with the registerIntentActivity method to register an Activity that will receive an Intent when the OCL Notification in the Notification Manager is pressed. Typically, the HostApp would set this Activity to the main activity, which includes the android.intent.action.MAIN / android.intent.category.LAUNCHER intent filter in the AndroidManifest.xml file.

### 2.1.17 EOclSettingChanged

The EOclSettingChanged enum is used by OCL to inform the HostApp of changes to OCL settings.

#### 2.1.17.1 OCL_SETTING_CHANGED_BAS

The Bandwidth Adaptive Streaming (BAS) setting has been changed. The new value can be read from OCL's basConfiguration property.

### 2.1.18 EOclImageShareState

The EOclImageShareState enum is used by OCL to inform the HostApp of changes to the image sharing state via the imageShareState property and the OclImageShareStateNotification.

Typically, the image share state transitions through the values in the following order:

- OCL_IMAGE_SHARE_STOPPED – no image sharing going on, default state
- Someone requests image sharing:
  - If locally (via the shareImage method): OCL_IMAGE_SHARE_REQUESTED_LOCAL
  - If remotely: OCL_IMAGE_SHARE_REQUESTED_REMOTE
- OCL_IMAGE_SHARE_STARTED – the image is transferred, and both participants are now viewing the same image
- OCL_IMAGE_SHARE_STOPPED – either participant has ended image sharing (locally via the stopImageSharing method)

#### 2.1.18.1 OCL_IMAGE_SHARE_STOPPED

The OCL_IMAGE_SHARE_STOPPED state indicates that there is currently no image being shared. Also, as part of the OclImageShareStateNotification, it indicates that image sharing has just ended.

#### 2.1.18.2 OCL_IMAGE_SHARE_REQUESTED_LOCAL

The OCL_IMAGE_SHARE_REQUESTED_LOCAL state indicates that there has been a local request to share an image and OCL is awaiting acceptance from the remote endpoint that it is ok to proceed with sharing the chosen image. If the remote endpoint accepts, the state changes to OCL_IMAGE_SHARE_STARTED; otherwise, the state changes to OCL_IMAGE_SHARE_STOPPED.

#### 2.1.18.3 OCL_IMAGE_SHARE_REQUESTED_REMOTE

The OCL_IMAGE_SHARE_REQUESTED_REMOTE state indicates that there has been a remote request to share an image. If OCL locally accepts the request based on internal conditions, the state will change to OCL_IMAGE_SHARE_STARTED; otherwise, the state will change to OCL_IMAGE_SHARE_STOPPED.

#### 2.1.18.4 OCL_IMAGE_SHARE_STARTED

The OCL_IMAGE_SHARE_STARTED state indicates that a local or remote request to share an image has been accepted and the image share has officially started. At this point, the image is transferred from requesting side to receiving side and displayed on both screens.

#### 2.1.18.5 OCL_IMAGE_SHARE_ERROR

The OCL_IMAGE_SHARE_ERROR state indicates that there was an error while trying to share an image.

### 2.1.19 EOclImageShareErrorCode

The EOclImageShareErrorCode enum is used with the OclImageShareErrorNotification event to provide the cause of a still image sharing error.

#### 2.1.19.1 OCL_IMAGE_SHARE_ERROR_UNKNOWN

The OCL_IMAGE_SHARE_ERROR_UNKNOWN error code indicates something went wrong but details are unknown.

#### 2.1.19.2 OCL_IMAGE_SHARE_ERROR_COPY_FAILED

The OCL_IMAGE_SHARE_ERROR_COPY_FAILED error code indicates that an attempt to copy the shared image into the file system location from where it is to be shared, has failed.

## 2.2 OclInterfaceFactory Methods

The OclInterfaceFactory class is used to obtain an instance of the OclInterface. The OclInterfaceFactory class contains the following methods.

### 2.2.1 getOclInterface

The getOclInterface method returns a singleton instance of the main OclInterface implementation.

static OclInterface getOclInterface()

#### *2.2.1.1 Parameters*

- None

#### *2.2.1.2 Return Value*

- An object implementing OclInterface.

#### *2.2.1.3 Discussion*

To use OCL, the getOclInterface method must be used to retrieve an instance of the OclInterface. The main OclInterface object is a singleton object, so if this method is called multiple times it will always return the same OclInterface instance.

## 2.3 OclInterface Methods

OclInterface contains the following methods.

Note: All OCL methods calls must be made on the HostApp's main thread. The Android methods Activity.runOnUiThread() or Looper.getMainLooper() can be used, if needed. The OCL is not architected to support simultaneous method invocations from different threads.

### 2.3.1 initialize

The initialize method is called to initialize the OCL. It must be called after the OclInterface object is created and before any other methods are called.

EOclReturnValue initialize(Context applicationContext, OclEventListener listener)
EOclReturnValue initialize(Context applicationContext, OclEventListener listener, String loginServer)
EOclReturnValue initialize(Context applicationContext, OclEventListener listener, String loginServer, String callsPath)

#### *2.3.1.1 Parameters*

- applicationContext
  - The HostApp application context.
- listener
  - An object implementing OclEventListener.
- loginServer
  - [optional] The Onsight Account Manager (OAM) server to login to when making a call to loginWithUsername or loginWithToken. Accepted values are:
    - onsight.librestream.com – used to specify the production OAM servers. This server is the default when calling initialize without a loginServer parameter.
    - oam.test.libreeng.com – used to specify the test system OAM servers.
    - null – uses the default login server
- callsPath
  - [optional] The path to where the call files (snapshots) will be saved. The default is to use the private internal files area, making the call files accessible only to the application that created them.

#### *2.3.1.2 Return Value*

- OCL_SUCCESS if the method completed successfully; or a specific value of EOclReturnValue indicates the error.

#### *2.3.1.3 Discussion*

The initialize method is asynchronous. The OclInitializeNotification::OCL_INIT_ID_INITIALIZED notification tells the HostApp when the initialization of OCL is complete. The OCL is not usable until it is fully initialized. If OCL function calls are made before the library is initialized, the function return value will be OCL_NOT_INITIALIZED.

This method allocates resources needed by the OCL, including starting the OclService. The calls to initialize and deinitialize should be called in pairs so that the resources are returned to the system when OCL functionality is no longer needed.

There are two approaches that could be used here:

1. Call initialize once at program start up and deinitialize when the program exits. Since initialize takes some time, the benefit of this approach is that the OCL is ready to be used at any time without much delay. The problem is that the OCL is taking up (possibly wasting) resources when the OCL is not in use.
2. Call initialize when the OCL is needed, use the OCL, and then call deinitialize when the use of the OCL is complete. The pro of this approach is that resources are not wasted while the OCL is not being used. The con of this approach is that initializing the OCL takes time and extends the time that it takes to make a call.

### 2.3.2 deinitialize

The deinitialize method is called when the OCL is no longer needed.

EOclReturnValue deinitialize()

#### *2.3.2.1 Parameters*

- None

#### *2.3.2.2 Return Value*

- OCL_SUCCESS if the method completed successfully; or a specific value of EOclReturnValue indicates the error.

#### *2.3.2.3 Discussion*

This function frees resources that were allocated by the initialize method or accumulated during normal library use, including stopping the OclService. Once this function is called, the OCL can no longer be used. The deinitialize method is asynchronous. The OclInitializeNotification::OCL_INIT_ID_DEINITIALIZED notification tells the HostApp when the de-initialization of OCL is complete. After this, the OCL is unusable until another initialize method is called, and no further notifications will be received.

### 2.3.3 getCallsPath

The getCallsPath method returns the path to where the call files (snapshots) will be saved.

String getCallsPath()

#### *2.3.3.1 Parameters*

- None

#### *2.3.3.2 Return Value*

- The path to where the call files (snapshots) will be saved.

#### *2.3.3.3 Discussion*

The getCallsPath method returns the path to where the call files (snapshots) will be saved. The default is to use the private internal files area, making the call files accessible only to the application that created them. The initialize method has an optional parameter that allows this path to be changed, making it possible to store the call files in an alternate location such as the external private or public files areas.

### 2.3.4 makeCall

The makeCall method initiates a SIP call from the local device to a remote endpoint.

EOclReturnValue makeCall(String remoteUri)
EOclReturnValue makeCall(String remoteUri, boolean reconnect)

EOclReturnValue makeCall(OclContact callee)
EOclReturnValue makeCall(OclContact callee, boolean reconnect)

#### *2.3.4.1 Parameters*

- remoteUri or callee:
    - remoteUri - The SIP URI of the remote endpoint.

- callee - The OclContact object representing the remote endpoint. The call will be made to the OclContact object's address property, which represents the contact's SIP URI.

- reconnect:
  - [optional] This parameter signifies if this call is an attempt to reconnect the immediately previous call. The OCL must have just previously been in a call to the same participant represented by the remoteUri parameter (or callee's address property). If the remoteUri parameter (or callee's address property) does not match with the previous call, this parameter has no effect. If reconnect is not present, the default is false.

#### 2.3.4.2 Return Value

- OCL_SUCCESS indicates that outgoing call was successfully initiated; or a specific value of EOclReturnValue indicates the error if the call could not be started.

#### 2.3.4.3 Discussion

The makeCall method kicks off the asynchronous process of making a call. The HostApp is notified of changes in the call state via the OclCallStateNotification. Once the HostApp receives an OclCallStateNotification::OCL_CALL_CONNECTED notification, the OclActivity should be displayed by the HostApp. See the Notifications and OclActivity sections below for details.

### 2.3.5 acceptCall

The acceptCall method is used to accept an incoming call.

EOclReturnValue acceptCall(long callHandle)

#### 2.3.5.1 Parameters

- callHandle:
  - The handle to the call that is being accepted. The callHandle is a value that uniquely identifies the specific call to the OCL.

#### 2.3.5.2 Return Value

- OCL_SUCCESS indicates that incoming call was successfully accepted; or a specific value of EOclReturnValue indicates the error if the call could not be accepted.

#### 2.3.5.3 Discussion

The HostApp receives an OclCallStateNotification::OCL_CALL_INCOMING notification from OCL when an incoming call is received. If the user wants to accept the call, the HostApp calls the acceptCall method. After a short time, the OclCallStateNotification notifies the HostApp that the call has reached the OCL_CALL_CONNECTED state, at which time the HostApp should display the OclActivity. If the user wants to decline the call, the HostApp calls the endCall method. See the Notifications and OclActivity sections below for details.

### 2.3.6 endCall

The endCall method ends an existing call, or declines an incoming call.

EOclReturnValue endCall(long callHandle)

#### 2.3.6.1 Parameters

- callHandle:
  - The handle to the call that is ending or being declined.

#### 2.3.6.2 Return Value

- OCL_SUCCESS indicates that the call was successfully ended; or a specific value of EOclReturnValue indicates the error if ending/declining the call caused an error.

#### 2.3.6.3 Discussion

The endCall method is used to end a connected call, decline an incoming call, or cancel a call that is the process of being established.

### 2.3.7 loginWithPassword

The loginWithPassword method is used by the HostApp to log into the Librestream OAM server using the standard username and password pair as credentials.

EOclReturnValue loginWithPassword(String username, String password)

#### 2.3.7.1 Parameters

- username:

  - the OAM username
- password:
  - the OAM user password corresponding to the username

#### 2.3.7.2 Return Value

- OclSuccess indicates the method has successfully kicked off the process of logging in to the OAM server.
- Other return values of EOclReturnValue enum indicate the reason for the failure. In these cases, the OAM login was not initiated and the login state remains OCL_NOT_LOGGED_IN.

#### 2.3.7.3 Discussion

The method is asynchronous and if it returns OCL_SUCCESS, the HostApp needs to listen for the OclLoginStateNotification for the result of the login attempt. If the method does not return OCL_SUCCESS, the login attempt was abandoned and there is no login activity.

When the login succeeds, the HostApp receives an OclLoginStateNotification with the OCL_LOGGED_IN state as a parameter. The OCL receives from the OAM server, among other things, Teamlink and SIP login credentials that OCL internally uses to try to Teamlink (if necessary) and SIP register. The HostApp monitors the SIP registration state by receiving the OclSipRegistrationStateNotification. Once the OCL_SIP_REGISTERED state is reached, the OCL is in a state in which a call can be attempted.

### 2.3.8 loginWithToken

The loginWithToken method is used by the HostApp to log into the Librestream OAM server using the username and token pair as credentials.

EOclReturnValue loginWithPassword(String username, String token)

#### 2.3.8.1 Parameters

- username:
  - the OAM username
- token:
  - the OAM token (received from Librestream OAM Token Service)

#### 2.3.8.2 Return Value

- OCL_SUCCESS indicates the method has successfully kicked off the process of logging in to the OAM server.
- Other return values of EOclReturnValue indicate the reason for the failure. In these cases, the OAM login was not initiated and the login state remains OCL_NOT_LOGGED_IN.

#### 2.3.8.3 Discussion

The method is asynchronous and if it returns OCL_SUCCESS, the HostApp needs to listen for the OclLoginStateNotification for the result of the login attempt. If the method does not return OCL_SUCCESS, the login attempt was abandoned and there is no login activity.

When the login succeeds, the HostApp receives an OclLoginStateNotification with the OCL_LOGGED_IN state as a parameter. The OCL receives from the OAM server, among other things, Teamlink and SIP login credentials that OCL internally uses to try to Teamlink (if necessary) and SIP register. The HostApp monitors the SIP registration state by receiving the OclSipRegistrationStateNotification. Once the OCL_SIP_REGISTERED state is reached, the OCL is in a state in which a call can be attempted.

### 2.3.9 logout

The logout method is used by the HostApp to log out of the OAM server.

EOclReturnValue logout()

#### 2.3.9.1 Parameters

- none

#### 2.3.9.2 Return Value

- OCL_SUCCESS is returned when the call was successful.
- Other return values of EOclReturnValue indicate the reason for the failure.

#### 2.3.9.3 Discussion

The method is asynchronous and if it returns OCL_SUCCESS, the HostApp needs to listen to the OclLoginStateNotification for the result of the logout attempt. If the method does not return OCL_SUCCESS, the logout attempt was abandoned and there is no logout activity.

When the logout method is successfully invoked, the OCL first internally SIP unregisters and when the SIP unregistered state is reached, the OAM logout is initiated. When the logout succeeds, the OCL_NOT_LOGGED_IN state is reached. The HostApp monitors the SIP registration state with the OclSipRegistrationStateNotification, and the login state with the OclLoginStateNotification.

### 2.3.10 overrideUserNames

The overrideUserNames method is used to override the names of the user that logs in to the OAM login server.

EOclReturnValue overrideUserNames(String firstName, String lastName)

#### 2.3.10.1 Parameters

- firstName:
  - the override for the first name of the OAM user
- lastName:
  - the override for the last name of the OAM user

#### 2.3.10.2 Discussion

The method may only be used after OCL has been initialized and before the OAM user login is attempted. It will return an error if used at other times.

To remove the override, use empty strings (“”).

The first and last names will override the OAM user’s first and last names that are retrieved from the OAM server when the login succeeds in the following ways:

- In the friendly name presented in the remote endpoint’s participant panel; and
- In the file name of captured still images, provided that the useFirstLastNamesInImageName property is set to true.

### 2.3.11 captureImage

The captureImage method is used by the HostApp to initiate a snapshot (image capture) and save the resulting image to a file.

EOclReturnValue captureImage()

#### 2.3.11.1 Parameters

- none

#### 2.3.11.2 Return Value

- OCL_SUCCESS is returned when the call is successful.
- Other return values of EOclReturnValue indicate the reason for the failure.

#### 2.3.11.3 Discussion

The method is asynchronous and if it returns OCL_SUCCESS, the HostApp needs to listen to the OclCaptureStateNotification for the result of the image capture attempt. If the method does not return OCL_SUCCESS, the image capture was abandoned and there is no further activity.

When the captureImage method is successfully invoked, the OCL initiates an image capture and reports its progress via the OclCaptureStateNotification. The HostApp will receive OCL_CAPTURE_START when the image capture starts, OCL_CAPTURE_COMPLETE when it ends, and OCL_CAPTURE_ERROR if an error occurs. Additionally, if the saveImageToCameraAlbum property is true, the image is saved to the camera album. If this process fails, there is an OCL_CAPTURE_CAMERA_ALBUM_ERROR notification indicating the error cause.

### 2.3.12 inviteGuestWithEmail

The inviteGuestWithEmail method is used by the HostApp to send an “invite guest” email to a prospective caller.

EOclReturnValue inviteGuestWithEmail(String firstName, String lastName, String emailAddress, String message, int expiryHours, boolean ignoreDuplicateEmail, boolean callInviterAutomatically)

#### 2.3.12.1 Parameters

- firstName:
  - first name of the person being invited
- lastName:
  - last name of the person being invited
- emailAddress:
  - email address of the person being invited
- message:
  - message to be sent to the person being invited
- expiryHours:
  - number of hours after which the OAM account associated with the invitation expires
- ignoreDuplicateEmail:
  - indicates whether it should ignore any duplicate email address that is already registered and continue sending the invitation
- callInviterAutomatically:
  - indicates whether the guest user should call the inviter automatically after logging in

#### 2.3.12.2 Return Value

- OCL_SUCCESS is returned when the call is successful.
- Other return values of EOclReturnValue indicate the reason for the failure.

#### 2.3.12.3 Discussion

The method sends an invite guest request to the OAM server. If it returns OCL_SUCCESS, an invite guest request has successfully been sent to the OAM server. Any other return value is an error code indicating what went wrong. The result of an invite guest request is then posted in an OclInviteGuestNotification. If the result is OCL_INVITE_SUCCESS, the OAM server has successfully created the OAM account associated with the invitation and sent an email to the prospective guest inviting them to download and use the Onsight Connect application. Any other result is an error code indicating what went wrong.

### 2.3.13 inviteGuestWithSms

The inviteGuestWithSms method is used by the HostApp to send an "invite guest" text message to a prospective caller.

EOclReturnValue inviteGuestWithSms(String firstName, String lastName, String phoneNumber, String message, int expiryHours, boolean callInviterAutomatically)

#### 2.3.13.1 Parameters

- firstName:
  - first name of the person being invited
- lastName:
  - last name of the person being invited
- phoneNumber:
  - mobile phone number of the person being invited
- message:
  - message to be sent to the person being invited
- expiryHours:
  - number of hours after which the OAM account associated with the invitation expires
- callInviterAutomatically:
  - indicates whether the guest user should call the inviter automatically after logging in

#### 2.3.13.2 Return Value

- OCL_SUCCESS is returned when the call is successful.
- Other return values of EOclReturnValue indicate the reason for the failure.

#### 2.3.13.3 Discussion

The method sends an invite guest request to the OAM server. If it returns OCL_SUCCESS, an invite guest request has successfully been sent to the OAM server. Any other return value is an error code indicating what went wrong. The result of an invite guest request is then posted in an OclInviteGuestNotification. If the result is OCL_INVITE_SUCCESS, the OAM server has successfully created the OAM account associated with the invitation and sent a text message to the prospective guest inviting them to download and use the Onsight Connect application. Any other result is an error code indicating what went wrong.

### 2.3.14 changePassword

The changePassword method is used by the HostApp to change the password of the OAM user.

EOclReturnValue changePassword(String userName, String oldPassword, String newPassword)

#### *2.3.14.1 Parameters*

- userName:
  - the OAM username
- oldPassword:
  - the old or expired password corresponding to the username
- newPassword:
  - the new password corresponding to the username

#### *2.3.14.2 Return Value*

- OCL_SUCCESS is returned when the call is successful.
- Other return values of EOclReturnValue indicate the reason for the failure.

#### *2.3.14.3 Discussion*

The method sends a change password request to the OAM server. If it returns OCL_SUCCESS, a change password request has successfully been sent to the OAM server. Any other return value is an error code indicating what went wrong. The result of a change password request is then posted in an OclChangePasswordNotification. If the result is OCL_CHANGE_PASSWORD_SUCCESS, the OAM server has successfully changed the password for the OAM user. Any other result is an error code indicating what went wrong.

There will be two cases where this method is used:

1. [optional] If the HostApp wants to provide a way for the current logged in user to change his/her password, the HostApp needs to provide UI to accept the old and new passwords that get passed into this method.
2. [mandatory when using inviteGuest] If the guest user initially logs in with the temporary password provided by the OAM server, the OAM server will force the guest user to change the password. In this case, the HostApp will receive the OclLoginStateNotification with EOclLoginReason of OCL_LOGIN_PASSWORD_EXPIRED. In response to this notification, the HostApp must present UI to accept the old and new passwords from the guest user. The old and new passwords are passed into this method. The HostApp needs to wait for OCL_CHANGE_PASSWORD_SUCCESS, after which the guest user is allowed to login again with the new password.

### 2.3.15 addOclEventListener

The addOclEventListener method is used to register an OclEventListener object that will receive notifications.

EOclReturnValue addOclEventListener(OclEventListener listener)

#### *2.3.15.1 Parameters*

- listener:
  - an object implementing OclEventListener

#### *2.3.15.2 Return Value*

- OCL_SUCCESS is returned when the call is successful.
- Other return values of EOclReturnValue indicate the reason for the failure.

#### *2.3.15.3 Discussion*

Once addOclEventListener has been called, the provided OclEventListener object will begin to receive notifications. This method only needs to be called if you want to add a listener object other than the one that was already provided in the call to initialize. For more information, refer to the OclInterface Notifications section of this document.

### 2.3.16 removeOclEventListener

The removeOclEventListener method is used to unregister an OclEventListener implementation, so that it will no longer receive notifications.

EOclReturnValue removeOclEventListener(OclEventListener listener)

#### 2.3.16.1 Parameters

- listener:
    - an object implementing OclEventListener

#### 2.3.16.2 Return Value

- OCL_SUCCESS is returned when the call is successful.
- Other return values of EOclReturnValue indicate the reason for the failure.

#### 2.3.16.3 Discussion

Once removeOclEventListener has been called, the provided OclEventListener object will no longer receive notifications. The OclEventListener passed in here must be an instance that was previously passed into addOclEventListener or initialize. Often the listener is an Activity that has registered when the Activity was created. In this case, it is important that the Activity.onDestroy() method contains a removeOclEventListener since the Activity won't be around anymore to receive events.

If removeOclEventListener is not called, all registered OclEventListener instances will be unregistered automatically after calling deinitialize. For more information, refer to the OclInterface Notifications section of this document.

### 2.3.17 registerIntentActivity

The registerIntentActivity method is used to register/unregister Activities that should receive Intents when certain situations occur within OCL.

```
EOclReturnValue registerIntentActivity(EOclIntentActivityRegistrationType iar, Class<?> cls)
```

#### 2.3.17.1 Parameters

- iar:
    - one of the values of EOclIntentActivityRegistrationType to indicate the Intent for which the given Activity is being registered
- cls:
    - the Activity class that should receive the Intent for the event specified by the iar parameter. This parameter can be null to unregister the Activity from the specified event.

#### 2.3.17.2 Return Value

- OCL_SUCCESS is returned when the call is successful.
- Other return values of EOclReturnValue indicate the reason for the failure.

#### 2.3.17.3 Discussion

The registerIntentActivity method can only be called when OCL is in the initialized state. See the description of EOclIntentActivityRegistrationType to get more information regarding the Intents for which Activities can be registered.

### 2.3.18 onPause

The onPause method is used by the HostApp to inform the OCL that the parent activity is entering the paused state.

```
EOclReturnValue onPause()
```

#### 2.3.18.1 Parameters

- none

#### 2.3.18.2 Return Value

- OCL_SUCCESS is returned when the call is successful.
- Other return values of EOclReturnValue indicate the reason for the failure.

#### 2.3.18.3 Discussion

The HostApp's activity should call this method when its onPause method is called. This allows the OCL to clean up any resources that are not required (e.g. camera).

### 2.3.19 onResume

The onResume method is used by the HostApp to inform the OCL that the parent activity is entering the resumed state.

```
EOclReturnValue onResume(Activity activity)
```

#### 2.3.19.1 Parameters

- activity:
  - the parent Activity object

#### 2.3.19.2 Return Value

- OCL_SUCCESS is returned when the call is successful.
- Other return values of EOclReturnValue indicate the reason for the failure.

#### 2.3.19.3 Discussion

The HostApp's activity should call this method when its onResume method is called.  This allows the OCL to re-acquire any necessary resources (e.g. camera).

### 2.3.20 shareImage

The shareImage method is used to share an image with the remote participant.

EOclReturnValue shareImage(String filename)

#### 2.3.20.1 Parameters

- filename:
  - the file name of the image to be shared

#### 2.3.20.2 Return Value

- OclSuccess is returned when the call is successful.
- Other return values of EOclReturnValue indicate the reason for the failure.
  - OclNotInialized – indicates OCLib has not been initialized.
  - OclNotAllowed – image sharing is only allowed when connected in an Onsight call to another Onsight participant

#### 2.3.20.3 Discussion

The shareImage method shares a specified image with the remote participant. As the image share progresses, image sharing state changes are passed to the HostApp via the OclImageShareStateNotification. Once the OCL_IMAGE_SHARE_STARTED state is reached, the specified image is sent to the remote participant. During the transfer, an image transfer window provides progress. When it reaches 100%, the transfer window clears, the image is now shared and the collaboration on the image can commence.

If an existing image is already being shared, calling shareImage with a new file name causes the existing shared image to be saved with the current telestration, the new shared image to be transferred to the remote endpoint and the new shared image to be displayed on the screen.

### 2.3.21 stopImageSharing

The stopImageSharing method is used to end image sharing.

EOclReturnValue stopImageSharing()

#### 2.3.21.1 Parameters

- None

#### 2.3.21.2 Return Value

- OclSuccess is returned when the call is successful.
- Other return values of EOclReturnValue indicate the reason for the failure.
  - OclNotInialized – indicates OCLib has not been initialized.
  - OclNotAllowed –
    - Image sharing related methods can only be called when connected in an Onsight call to another participant.
    - Trying to end image sharing when in the OCL_IMAGE_SHARE_STOPPED state.

#### 2.3.21.3 Discussion

If image sharing is in progress, this method ends the image sharing regardless of whether the image sharing was started locally or remotely. At this point, the telestration that was drawn on the shared image will be saved into the image so it can be drawn on the image if the image is shared again.

## 2.4 OclInterface Properties

OclInterface has the following properties giving easy access to current state.

Attempting to read a property before OCL is initialized will result in the default value for that property being returned. Attempting to set a property before OCL is initialized will cause the setter for that property to return OCL_NOT_INITIALIZED, otherwise it will return OLC_SUCCESS.

### 2.4.1 initialized

boolean isInitialized()

#### *2.4.1.1 Discussion*

The initialized property retains the initialized/deinitialized state of the OCL. It matches the value of the last OclInitializeNotification event.

The initial value is false.

### 2.4.2 sipRegistrationState

EOclSipRegistrationState getSipRegistrationState()

#### *2.4.2.1 Discussion*

The sipRegistrationState property allows the HostApp to check the current SIP registration state. It matches the value of the last OclSipRegistrationNotification event.

The initial value is OCL_SIP_UNREGISTERED.

### 2.4.3 callState

EOclCallState getCallState()

#### *2.4.3.1 Discussion*

The callState property allows the HostApp to check the state of the current incoming or outgoing SIP call. It matches the value of the last OclCallStateNotification event.

The initial value is OCL_CALL_DISCONNECTED.

### 2.4.4 videoState

EOclVideoState getVideoState()

#### *2.4.4.1 Discussion*

The videoState property allows the HostApp to check the state of the video stream. It matches the value of the last OclVideoStateNotification event.

The initial value is OCL_VIDEO_STOPPED.

### 2.4.5 version

String getVersion()

#### *2.4.5.1 Discussion*

The version property presents the version of the OCL as a string.

### 2.4.6 deleteImagesAfterTransfer

boolean isDeleteImagesAfterTransfer()
EOclReturnValue setDeleteImagesAfterTransfer(boolean deleteImagesAfterTransfer)

#### *2.4.6.1 Discussion*

The deleteImagesAfterTransfer property allows the HostApp to control whether the library deletes captured still image files after they are successfully transferred to the remote endpoint, and when the library de-initializes.

The initial value is false.

### 2.4.7 useFirstLastNameInImageName

boolean isUseFirstLastNameInImageName()

EOclReturnValue setUseFirstLastNameInImageName(boolean useFirstLastNameInImageName)

#### 2.4.7.1 Discussion

The useFirstLastNameInImageName property controls whether the first and last names of the current user are used in the name of captured still images. If this property is set to true, the name of still images follow the form IMG_nnnn_<First><Last>.jpg, where nnnn is an increasing number and <First> and <Last> represent the first and last name of the user. If this property is set to false, the image name format is IMG_nnnn.jpg.

The initial value is false.

### 2.4.8 saveImagesToCameraAlbum

boolean isSaveImagesToCameraAlbum()
EOclReturnValue setSaveImagesToCameraAlbum(boolean saveImagesToCameraAlbum)

#### 2.4.8.1 Discusson

The saveImagesToCameraAlbum property controls whether captured still images are saved to the device’s camera album. The location of the stored images is determined by the cameraAlbumName property.

The initial value is false.

### 2.4.9 burnTelestrationOnCameraAlbumImages

boolean isBurnTelestrationOnCameraAlbumImages()
EOclReturnValue setBurnTelestrationOnCameraAlbumImages(boolean burnTelestrationOnCameraAlbumImages)

#### 2.4.9.1 Discussion

The burnTelestrationOnCameraAlbumImages property controls whether the captured image’s telestration is burned onto the image before they are saved into the device’s camera album. Typically, still images captured in the Librestream Onsight system store telestration into an Exif tag so that when the image is displayed, the telestration’s visibility is controllable. However, external photo applications do not support viewing telestration saved in the Exif tag so if this property is set, the telestration is burned onto the image.

The initial value is true.

### 2.4.10 cameraAlbumName

String getCameraAlbumName()
EOclReturnValue setCameraAlbumName(String cameraAlbumName)

#### 2.4.10.1 Discusson

The cameraAlbumName property controls the location that images are stored when saveImagesToCameraAlbum is set to true.

Camera album images are saved to the public pictures directory, as determined by the getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES) Android API call. The cameraAlbumName property is used to specify the destination sub-folder within the public pictures directory. If the specified cameraAlbumName is not a valid folder name, taking snapshots will fail with error OCL_CAPTURE_ERROR_CAMERA_ALBUM_WRITE_FAILURE.

If no cameraAlbumName is provided, the default sub-folder will be the name of the HostApp.

### 2.4.11 loginState

EOclLoginState getLoginState()

#### 2.4.11.1 Discusson

The loginState property indicates the login state of the OCL represented by the enum EOclLoginState.

The initial value is OCL_NOT_LOGGED_IN.

### 2.4.12 basConfiguration

EOclBandwidthAdaptiveStreaming getBasConfiguration()
EOclReturnValue setBasConfiguration(EOclBandwidthAdaptiveStreaming basConfiguration)

#### 2.4.12.1 Discusson

The basConfiguration property is used to set the BAS feature of the OCL and find out what the current setting is. Setting the basConfiguration property to OCL_BAS_USE_DEFAULT, changes the internal BAS setting to the domain default provided by the login server (OAM). Changes made to the property result in OclSettingChangedNotification events being sent to the HostApp.

The initial value is OCL_BAS_ENABLED_FOR_CELLULAR_ONLY.

### 2.4.13 enableGpsLocationUpdates

boolean isEnableGpsLocationUpdates()
EOclReturnValue setEnableGpsLocationUpdates(boolean enableGpsLocationUpdates)

#### 2.4.13.1 Discusson

The enableGpsLocationUpdates property controls whether to save location information (GPS coordinates) into the Exif tags of captured images. Enabling this property requires that the HostApp request the ACCESS_FINE_LOCATION permission in the AndroidManifest.xml file.

If Location Services on Android are disabled, attempting to enable location updates will return OCL_MISSING_PERMISSION. You can use this return value to inform the user that they must enable Location Services on their device. There is no need to call setEnableGpsLocationUpdates again after Location Services are enabled; the OCL will begin receiving location updates as soon as Location Services is enabled.

The initial value is false.

### 2.4.14 callInfo

OclCallInfo getCallInfo()

#### 2.4.14.1 Discussion

The callInfo property returns an OclCallInfo object containing information about the current call, including:

- String callerName – name of the caller
- String callerAddress – address of the caller
- boolean incoming – true for incoming calls, false for outgoing calls
- boolean encrypted – true for encrypted calls, false otherwise
- boolean remoteRecording – true if the caller is recording, false if not
- String appVersion – version number of the caller’s program

The remoteRecording and appVersion fields become available when the HostApp receives OclRemoteRecordNotification and OclRemoteVersionNotificaton notifications, respectively. If the program is not in a call, all returned fields are undefined.

### 2.4.15 systemHealth

OclSystemHealth getSystemHealth()

#### 2.4.15.1 Discussion

The systemHealth property returns an OclSystemHealth object containing information about the OCL’s health, including:

- EOclSystemHealth networkInterface – state of the network interface
- EOclSystemHealth onsightAccountService – state of the Onsight Account Service
- EOclSystemHealth sipRegistration – state of the SIP registration
- EOclSystemHealth ciscoRegistration – state of the Cisco registration
- EOclSystemHealth teamLinkRegistration – state of the TeamLink registration
- EOclSystemHealth remoteManagement – state of the remote management service

### 2.4.16 personalContacts

ArrayList<OclContact> getPersonalContacts()

#### 2.4.16.1 Discussion

The personalContacts property is used to obtain the personal contacts of the currently logged in user. After a user successfully logs into OAM, the user’s personal contacts are received by OCL from the login server (OAM), and the HostApp is informed of their availability with the OclPersonalContactsUpdatedNotification event. After this event, the personalContacts property contains an array of zero or more OclContact objects.

If the logged in user's personal contacts are updated elsewhere (on OAM or a full Onsight client application), OAM sends these changes to OCL, which notifies the HostApp of the changes with the same OclPersonalContactsUpdatedNotification event. This event indicates that the personalContacts property has a new array of OclContact objects with the changes included.

### 2.4.17 imageShareState

EOclImageShareState getImageShareState()

#### *2.4.17.1 Discussion*

The imageShareState property indicates the current state of image sharing within OCL. Notifications of changes to the state are provided to the HostApp using the OclImageShareStateNotification event.

## 2.5 OclEventListener Notifications

The OCL uses the Java Event Listener pattern to notify the HostApp about interesting events, status and errors that occur in the library.

The HostApp must register an object with OCL that implements OclEventListener when it calls initialize. If required, additional listeners can be registered through a call to addOclEventListener.

When an event occurs within OCL that the HostApp must be notified of, the appropriate callback on the provided OclEventListener instances will fire.

When the HostApp no longer wishes to receive notifications, it can call removeOclEventListener to unregister an OclEventListener instance. Any remaining registered OclEventListener instances will be unregistered automatically when the HostApp calls deinitialize.

Note: All OCL notification callbacks occur on the HostApp's main thread.

### 2.5.1 OclSipRegistrationStateNotification

void onLoginStateNotification(OclEvents.OclLoginStateNotification e)

#### *2.5.1.1 Discusson*

Before a SIP call can be placed or received, the local endpoint must be registered with a SIP registrar (server). SIP registration is initiated by OCL automatically after it reaches the OCL_LOGGED_IN state. As the SIP registration progresses, SIP registration state updates are sent to the HostApp via the OclSipRegistrationStateNotification.

The OclSipRegistrationStateNotification object contains the following public fields:

- sipRegistrationState: The new SIP registration state, which can take one of the values of the enum EOclSipRegistrationState.

The notifications are mostly informational in nature; the HostApp doesn't need to respond in any way with the OCL. The HostApp may want to present some sort of UI to let the user know that the SIP registration through OCL is progressing.

Once the OCL_SIP_REGISTERED state is reached, the OCL is ready to accept incoming or make outgoing SIP calls.

### 2.5.2 OclCallStateNotification

void onCallStateNotification(OclEvents.OclCallStateNotification e)

#### *2.5.2.1 Discussion*

When an outgoing call is made with the makeCallWtihUri method, call progress state updates are sent to the HostApp via the OclCallStateNotification.

The OclCallStateNotification object contains the following public fields:

- callState: The new call state, which can take one of the values of the enum EOclCallState.
- callHandle: The call handle for the specific call to which this notification refers. The call handle is used as a parameter for acceptCall and endCall methods. The call handle is first identified to the HostApp in the OCL_CALL_INCOMING notification for incoming calls, and in the OCL_CALL_IN_PROGRESS notification for outgoing calls.

- callEndReason: This field is only relevant if callState is set to OCL_CALL_DISCONNECTED, which signifies the end of a call. It gives a reason why the call ended and takes one of the values of EOclEndCallReason.
- callerName: The name of the caller. This is only present if the callState is set to OCL_CALL_INCOMING.
- callerAddress: The address of the caller. This is only present if the callState is set to OCL_CALL_INCOMING.

Some of the notifications are informational in nature. The HostApp may want to present some sort of UI to let the user know that the call through OCL is progressing.

There are some notifications that require the HostApp to take action:

- OclCallStateNotification::OCL_CALL_CONNECTED: When this notification arrives, the HostApp should display the OclActivity. See the OclActivity section below.
- OclCallStateNotification::OCL_CALL_INCOMING: When this notification arrives, the HostApp should present some UI to notify the user of the incoming call and present Accept or Decline options. When the user chooses, the HostApp must call one of the acceptCall or endCall methods to accept or decline the incoming call. This is the first time that the HostApp sees the call handle (via the callHandle field) for incoming calls. It should be saved at this point so that the HostApp can work with the call (like declining or accepting it).
- OclCallStateNotification::OCL_CALL_IN_PROGRESS: When this notification arrives, if the HostApp has made the outgoing call, this is the first time that the HostApp sees the call handle (via the callHandle field). It should be saved at this point so that the HostApp can work with the call (like cancelling it).

### 2.5.3 OclVideoStateNotification

void onVideoStateNotification(OclEvents.OclVideoStateNotification e)

#### *2.5.3.1 Discusson*

When the OclActivity starts streaming video to the remote endpoint, the HostApp is notified using the OclVideoStateNotification.

The OclVideoStateNotification object contains the following public fields:

- videoState: The new video stream state, which can take one of the values of the enum EOclVideoState.

### 2.5.4 OclErrorNotification

void onErrorNotification(OclEvents.OclErrorNotification e)

#### *2.5.4.1 Discusson*

The OclErrorNotification identifies errors that have asynchronously occurred in the OCL.

The OclErrorNotification object contains the following public fields:

- errorId: The specific error, which can take one of the values of the enum EOclErrorId.

### 2.5.5 OclInitializeNotification

void onInitializeNotification(OclEvents.OclInitializeNotification e)

#### *2.5.5.1 Discusson*

The OclInitializeNotification identifies the initialization state of the OCL.

The OclInitializeNotification object contains the following public fields:

- initId: The specific initialized state, which can take one of the values of the enum EOclInitId, as follows:

    - OCL_INIT_ID_INITIALIZED – The OCL posts this notification to indicate that the OCL is now initialized and ready for use. After the initialize method is called, the HostApp must wait for this notification before using the rest of the OCL.
    - OCL_INIT_ID_DEINITIALIZED – The OCL posts this notification to indicate that the OCL has been deinitialized and can no longer be used. After the deinitialize method is called, this notification signals that the de-initialization process is complete.
    - OCL_INIT_ID_FAILURE – The OCL posts this notification to indicate that the initialization of the OCL has failed.

### 2.5.6 OclLoginStateNotification

void onLoginStateNotification(OclEvents.OclLoginStateNotification e)

#### 2.5.6.1 Discusson

The OclLoginStateNotification identifies the login state of the OCL.

The OclLoginStateNotification object contains the following public fields:

- loginState: identifies the new login state that was just reached, and takes one of the values of the EOclLoginState enum.
- loginReason: identifies further information about the login state change, and takes one of the values of the EOclLoginReason enum.
- loginStatusCode: identifies a standard HTTP status code if the loginReason is equal to OCL_LOGIN_HTTP_CODE or an internal server status code if the loginReason is equal to OCL_LOGIN_SERVER_CODE. For all other values of loginReason, this parameter is irrelevant.

### 2.5.7 OclCaptureStateNotification

void onCaptureStateNotification(OclEvents.OclCaptureStateNotification e)

#### 2.5.7.1 Discusson

The OclCaptureStateNotification identifies the image capture state of the OCL.

The OclCaptureStateNotification object contains the following public fields:

- captureState: identifies the new image capture state that was just reached and takes one of the values of the EOclCaptureState enum.
- captureFileName: identifies the file name of the captured image if captureState is equal to OCL_CAPTURE_COMPLETE. For all other values in captureState, this parameter is irrelevant.
- captureStatusCode: identifies the initiator (local or remote) of the image capture if captureState is equal to OCL_CAPTURE_STARTED, or an error code if it's equal to OCL_CAPTURE_ERROR. For all other values in captureState, this parameter is irrelevant.

A normal image capture goes through the following process:

- When the image capture is initiated, an OclCaptureStateNotification event is received by the app with the capture state set to OclCaptureStateStart, and a status code parameter indicating whether the capture was initiated locally or remotely.
- When the capture completes, an OclCaptureStateNotification event is received with the capture state set to OclCaptureComplete. At this point, the image was been written into the app's sandbox (its own area of the device's flash storage).
- If the saveImagesToCameraAlbum property is set to true, the OCL will also save the image into the camera album. If this process gets an error, the app will receive an OclCaptureStateNotification event with the capture state set to OclCaptureCameraAlbumError.

The OclCaptureStateNotification event with the OclCaptureStateError capture state indicates that the capture to the sandbox has failed, and there is no attempt to write the image to the camera album.

### 2.5.8 OclSystemHealthNotification

void onSystemHealthNotification(OclEvents.OclSystemHealthNotification e)

#### 2.5.8.1 Discusson

The OclSystemHealthNotification occurs when some part of the system health has changed.

The OclSystemHealthNotification object contains the following public fields:

- systemHealth: The current system health.

### 2.5.9 OclRemoteVersionNotification

void onRemoteVersionNotification(OclEvents.OclRemoteVersionNotification e)

#### 2.5.9.1 Discusson

The OclRemoteVersionNotification occurs when the remote app version number has been obtained.

The OclRemoteVersionNotification object contains the following public fields:

- remoteVersion: The version number of the remote endpoint. You can also query the appVersion field of the callInfo property.

### 2.5.10 OclRemoteRecordNotification

void onRemoteRecordNotification(OclEvents.OclRemoteRecordNotification e)

#### *2.5.10.1 Discusson*

The OclRemoteRecordNotification occurs when the remote participant's recording state has changed.

The OclRemoteRecordNotification object contains the following public fields:

- remoteRecording: The state of remote record. You can also query the remoteRecording field of the callInfo property.

### 2.5.11 OclInviteGuestNotification

void onInviteGuestNotification(OclEvents.OclInviteGuestNotification e)

#### *2.5.11.1 Discusson*

The OclInviteGuestNotification occurs when the OAM server has processed a guest invitation request.

The OclInviteGuestNotification object contains the following public fields:

- inviteGuestResult: The result of a guest invitation request, which can take one of the values of the EOclInviteGuestResult.

### 2.5.12 OclChangePasswordNotification

void onChangePasswordNotification(OclEvents.OclChangePasswordNotification e)

#### *2.5.12.1 Discusson*

The OclChangePasswordNotification occurs when the OAM server has processed a change password request.

The OclChangePasswordNotification object contains the following public fields:

- changePasswordResult: The result of a change password request, which can take one of the values of the EOclChangePasswordResult.

### 2.5.13 OclImageShareStateNotification

void onImageShareStateNotification(OclEvents.OclImageShareStateNotification e)

#### *2.5.13.1 Discussion*

The OclImageShareStateNotification event occurs when there is a change in the image share state. The OclImageShareStateNotification object contains the following public field:

- imageShareState: The new image sharing state. The image sharing state can always be checked with the imageShareState property of the OCL.

### 2.5.14 OclImageShareNewImageNotification

void onImageShareNewImageNotification(OclEvents.OclImageShareNewImageNotification e)

#### *2.5.14.1 Discussion*

The OclImageShareNewImageNotification event occurs when a new image is being shared. Once the image sharing state reaches OclImageShareStarted, the shared image can be changed by calling the shareImage method again with a different image file name. The remote participant can also change the shared image. When a different image is shared, OCL lets the HostApp know using the OclImageShareNewImageNotification event. The OclImageShareNewImageNotification object parameter contains the following public field:

- newImageFileName: The file name of the new shared image.

### 2.5.15 OclImageShareErrorNotification

void onImageShareErrorNotification(OclEvents.OclImageShareErrorNotification e)

#### *2.5.15.1 Discussion*

The OclImageShareErrorNotification event occurs when there is an error somewhere in the image sharing process. The OclImageShareErrorNotification object parameter contains the following public field:

- imageShareErrorCode: The share error code.

### 2.5.16 OclPersonalContactsUpdatedNotification

void onPersonalContactsUpdateNotification()

#### 2.5.16.1 Discussion

The OclPersonalContactsUpdateNotification event tells the HostApp that, after a successful login, the user's personal contacts are now available via the personalContacts property. The event also can be triggered if the user's personal contacts list has been changed elsewhere (on the OAM website or at a full Onsight client) and the updated list is now available via OCL's personalContacts property.

### 2.5.17 OclSettingChangedNotification

void onSettingChangedNotification(OclEvents.OclSettingChangedNotification e)

#### 2.5.17.1 Discussion

The OclSettingChangedNotification event informs the HostApp that one of the settings type properties have changed internally within OCL. The OclSettingChangedNotification object parameter contains the following public field:

- settingChanged: The setting that has changed.

## 2.6 OclService

In order to ensure that calls remain connected when the HostApp goes into the background, the OCL runs in the context of an Android foreground service called OclService. The service is started automatically when the HostApp calls initialize, and is stopped when the HostApp calls deinitialize.

When OCL is initialized, a notification for the service will appear in the status bar to indicate to the user that the OclService is still active, even if the HostApp is not in the foreground.

### 2.6.1 Foreground Service Notification Resources

The OclService requires a Notification object when it is promoted into a foreground service. The notification includes a few strings, colors and images that are specified as resources in OCLib. They are:

- ocl_service_name – the foreground service name (string) in the notification area
- ocl_service_text_not_logged_in – the foreground service text (string) in the notification area when OCL has been initialized but there is no OAM user logged in
- ocl_service_color_not_logged_in – the accent color (color) in the notification area when OCL has been initialized but there is no OAM user logged in
- ocl_service_text_idle – the foreground service text (string) in the notification area when OCL has been initialized, an OAM user is logged in but there is no connected Onsight call
- ocl_service_color_idle – the accent color (color) in the notification area when OCL has been initialized, an OAM user is logged in but there is no connected Onsight call
- ocl_service_text_connected – the foreground service text (string) in the notification area when OCL has been initialized, an OAM user is logged in and there is a connected Onsight call
- ocl_service_color_connected – the accent color (color) in the notification area when OCL has been initialized, an OAM user is logged in and there is a connected Onsight call
- ocl_service_image – the foreground service image (drawable) in the notification area

These resources can be overridden by the HostApp by including like-named resources within the **HostApp's** resources.

## 2.7 Using OCL in Your Project

The following section describes how to use OCL in your project.

### 2.7.1 Importing OCL into Your Workspace

To use the OCL in your project using Eclipse with ADT:

1. Copy the entire OCLib library project folder to your hard disk.

2. Import the library project into your workspace. Navigate to File > Import, select Android > Existing Android Code Into Workspace and choose the location of the OCLib library project folder.

3. Add a reference to the OCL in your HostApp project by following the instructions listed in Referencing a Library Project here:

   https://developer.android.com/tools/projects/projects-eclipse.html#ReferencingLibraryProject

### 2.7.2 Android Manifest

The OCL requires the following permissions, activities and services to be declared in the AndroidManifest.xml of the HostApp:

```
<uses-feature android:name="android.hardware.bluetooth" android:required="false" />
<uses-feature android:name="android.hardware.camera" android:required="false"/>
<uses-feature android:name="android.hardware.camera.any" android:required="true" />
<uses-feature android:name="android.hardware.camera.autofocus" android:required="false"/>
<uses-feature android:name="android.hardware.location" android:required="false"/>
<uses-feature android:name="android.hardware.location.gps" android:required="false"/>

<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BROADCAST_STICKY"/>
<!-- The READ_PHONE_STATE permission is required to listen to cell call state changes using the
TelephonyManager.PhoneStateListener in Android pre-6.0 -->
<uses-permission android:name="android.permission.READ_PHONE_STATE” android:maxSdkVersion=”22"/>

<!-- Optional permissions -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />

<application>
    <activity
        android:name="com.librestream.oclib.OclActivity"
              android:screenOrientation="sensorLandscape"
              android:configChanges="keyboardHidden|orientation|screenSize">
    </activity>

    <service
        android:exported="false"
        android:name="com.librestream.oclib.OclService" />
</application>
```

These can either be added manually to the manifest file of the HostApp, or the manifest merger tool can be used to bring them in automatically by adding manifestmerger.enabled=true to the project.properties file of the HostApp.

Note that the Bluetooth feature is specified as being optional, as it is not required in order for OCL to operate. More information on features is available in the Features section.

If GPS location information is not required in captured images, the ACCESS_FINE_LOCATION permission can be omitted.

### 2.7.3 Features

The OCL requires the following hardware features in order to operate fully. These features are declared in the OCL's AndroidManifest.xml. They are separated into two types:

- The features that are mandatory for OCL to initialize and work properly. The HostApp can mark these features as not required, but then if HostApp is installed on a device that doesn't have all these features, OCL will fail to initialize and cannot be used. This may be useful in situations where the HostApp provides functionality other than that provided by OCL.
    - android.hardware.camera.any
    - android.hardware.microphone
    - android.hardware.screen.landscape
- The features that are not mandatory for OCL to initialize. The OCL will work but functionality associated with these features will be missing.
    - android.hardware.bluetooth

- android.hardware.camera
- android.hardware.camera.autofocus
- android.hardware.location
- android.hardware.location.gps

Unless otherwise specified, the Google Play store will prevent the HostApp from being installed on devices that are missing the above listed features. To override this behavior, you can explicitly declare the above features as optional in the AndroidManifest.xml of the HostApp. For example:

```
<uses-feature android:name="android.hardware.camera" android:required="false"/>
```

This would allow the HostApp to be installed on a device without a back camera.

An attempt to initialize OCL will fail with a return value of OCL_NOT_SUPPORTED if any required features are unavailable.

### 2.7.4 ProGuard

If you are using ProGuard to obfuscate your HostApp, in order prevent ProGuard from removing or renaming required classes within OCL it is recommended that you configure the proguard-project.txt file in your HostApp to keep everything under the com.librestream.oclib.* package.

## 2.8 OclInterface Constants

The following constants are used by OCL.

### 2.8.1 OCL_CALL_HANDLE_KEY

When an Incoming Call Intent is received (see initialize method) by an Activity class, the Intent contains an extended data dictionary. The OCL_CALL_HANDLE_KEY is used to retrieve the call handle from the dictionary. The call handle value associated with this key is a long:

- getIntent().getLongExtra(OclInterface.OCL_CALL_HANDLE_KEY, 0)

# 3 OclActivity

The OclActivity is a full screen activity that presents a minimalist Onsight Connect collaboration screen. When the OclCallConnected state is reached, the HostApp should start the activity, such that it appears on the display.

For example:

```
Intent intent = new Intent(this, OclActivity.class);
        startActivity(intent);
```

The OclActivity has the following visual components:

- When the activity appears, it immediately starts the viewfinder functionality that shows what the user is pointing the device's camera at (the video is NOT being shared (streamed) to the remote participant at this point).
- The activity has a single button in the top right corner that allows the user to hang up the call.
- Immediately to the left of the hang up button, the activity displays a set of status icons that indicate the following:
    - on Video sharing on/off – The video sharing capability is remotely initiated in the HostApp by the remote participant using the Onsight Connect for Windows client software. While the video is being shared from the Android device to the remote participant, this icon will be visible.
    - ⚠ Warnings available – Communications warnings occur whenever there are transmission problems with the host. When this happens, the warnings are added to a list and this icon becomes visible. While visible, the user can tap anywhere in the status bar region to display the list of warnings. Tapping anywhere outside the list will cause it to go away.
    - ◉ Recording active – Whenever the remote host is recording the current call, this icon will be visible.
    - Microphone muted – Whenever the remote host has muted the device's microphone, this icon will be visible.

    - 🔇 Speakers muted – Whenever the remote host has muted the device's audio output, this icon will be visible.
- The image capture capability is remotely initiated in the HostApp by the remote participant using the Onsight Connect for Windows client software. When the image is captured, a pop-up window appears in the OclActivity showing progress as the still image is captured and transferred from the Android device to the remote participant.

## 3.1 OclActivity Methods

The OclActivity class contains the following methods.

### 3.1.1 showModal

The showModal method is called to display a popup view modally.

```
void showModal(View view)
void showModal(View view, boolean tapOutsideToClose)
void showModal(View view, boolean tapOutsideToClose, EModalPlacement placement)
```

#### *3.1.1.1 Parameters*

- view:
    - The view to be displayed modally.
- tapOutsideToClose:
    - [optional] Specifies whether the view should be hidden if the user taps the screen outside the view. If tapOutsideToClose is not provided the default value of false will be used.
- placement:
    - [optional] Specifies the location of the view and must be one of the following values:
        - MODAL_CENTER (default if placement is not provided)
        - MODAL_TOP_LEFT
        - MODAL_TOP_RIGHT
        - MODAL_BOTTOM_LEFT
        - MODAL_BOTTOM_RIGHT

#### *3.1.1.2 Return Value*

- None

#### *3.1.1.3 Discussion*

The showModal method is used to display a popup modal view over top of the OclActivity. While the view is displayed, all other controls on the screen are grayed out and disabled.

### 3.1.2 hideModal

The hideModal method is called to hide a modal view.

```
void hideModal()
void hideModal(View view)
```

#### *3.1.2.1 Parameters*

- view:
    - [optional] The modal view to be hidden. If view is not specified, the currently displayed modal popup will be hidden.

#### *3.1.2.2 Return Value*

- None

#### *3.1.2.3 Discussion*

The hideModal method is used to hide a modal popup view as displayed by the showModal method. If view is passed into the method and it is the current modal view, it is hidden. If it isn't the current modal view, no action is taken. If view is not passed into the method the currently displayed modal view will be hidden.

### 3.1.3 addWarning

The addWarning method is called to add a warning message to the list of warnings.

```
static int addWarning(String text)
```

#### 3.1.3.1 Parameters

- text:
  - The warning message to be added to the list.

#### 3.1.3.2 Return Value

- The return value is an identifier that is associated with the new message. This identifier can be used later to remove the warning message from the list via the removeWarning method.

#### 3.1.3.3 Discussion

The addWarning method adds a warning message to the internal list of warnings. If the list was previously empty, this will also cause the warning status icon to be displayed.

### 3.1.4 removeWarning

The removeWarning method removes a warning message from the list of warnings.

```
static void removeWarning(int warningId)
```

#### 3.1.4.1 Parameters

- warningId:
  - The identifier (as returned by addWarning) of the warning message to be removed from the list.

#### 3.1.4.2 Return Value

- None

#### 3.1.4.3 Discussion

The removeWarning method removes a warning message from the internal list of warnings. If the removal makes the list empty, this will also cause the warning status icon to be hidden.

### 3.1.5 removeAllWarnings

The removeAllWarnings method is called to remove all existing warning messages from the list of warnings.

```
static void removeAllWarnings()
```

#### 3.1.5.1 Parameters

- None

#### 3.1.5.2 Return Value

- None

#### 3.1.5.3 Discussion

The removeAllWarnings method removes all existing warning messages from the internal list of warnings. This will also cause the warning status icon to be hidden.

### 3.1.6 handleHangupButton

The handleHangupButton method is called when the hang-up button of the OclActivity is pressed.

```
public boolean handleHangupButtonPress(View view)
```

#### 3.1.6.1 Parameters

- view:
  - the hang-up button view

#### 3.1.6.2 Return Value

- boolean:
  - true if the button press was fully handled; otherwise false

#### 3.1.6.3 Discussion

By default, the hang-up button press handling ends the current call and closes the OclActivity. If the HostApp wants to do something else, it can subclass OclActivity and override this method to extend the handler for the hang-up button. If the default processing is still required, don't forget to call the base handler: super.handleHangupButton(view).

### 3.1.7 getMainLayout

The getMainLayout method returns the main layout view for the activity.

RelativeLayout getMainLayout()

#### *3.1.7.1 Parameters*

- None

#### *3.1.7.2 Return Value*

- RelativeLayout
  - the topmost layout view

#### *3.1.7.3 Discussion*

The getMainLayout method returns the main layout view for the activity. All other views are contained within this view. At a minimum this will include the OclRenderer view and a LinearLayout containing the hangup button and status bar.

## 3.2 OclActivity Properties

The OclActivity class has the following properties:

### 3.2.1 hangupButton

ImageButton getHangupButton()

#### *3.2.1.1 Discussion*

The hangupButton property gives the HostApp access to the hang-up button that is displayed in the top-right corner of the OclActivity.

### 3.2.2 statusBar

OclStatusBar getStatusBar()

#### *3.2.2.1 Discussion*

The statusBar property gives the HostApp access to the status bar that is displayed immediately to the left of the hang-up button in the top-right corner of the OclActivity. The status bar is used to display a series of icons that represent the following conditions:

- video or image is being shared
- warnings are available
- the current call is being recorded
- the microphone is muted
- the speaker is muted

When there are warnings available, the user can tap on the status bar to display the warning messages in a pop-up window.

For branding purposes you can change the sharing icon with the setSharingImage method as follows:

void setSharingImage(Bitmap sharingImage)

#### *3.2.2.2 Parameters*

- sharingImage:
  - The image to be used for the sharing status icon.

#### *3.2.2.3 Discussion*

In order for the image to fit properly within the status bar, it should have dimensions based on the DPI as follows:

| | |
|---|---|
| LDPI | 12 x 12 |
| MDPI | 24 x 24 |

| | |
|---|---|
| HDPI | 36 x 36 |
| XDPI | 48 x 48 |
| XXDPI | 72 x 72 |
| XXXDPI | 86 x 86 |

### 3.2.3 pinchAndZoomEnabled

static boolean isPinchAndZoomEnabled()
static void setPinchAndZoomEnabled(boolean pinchAndZoomEnabled)

#### *3.2.3.1 Discussion*

The pinchAndZoomEnabled property enables the OCL pinch and zoom functionality which permits the user to locally resize the viewfinder video using pinch and zoom gestures.

The initial value is true.

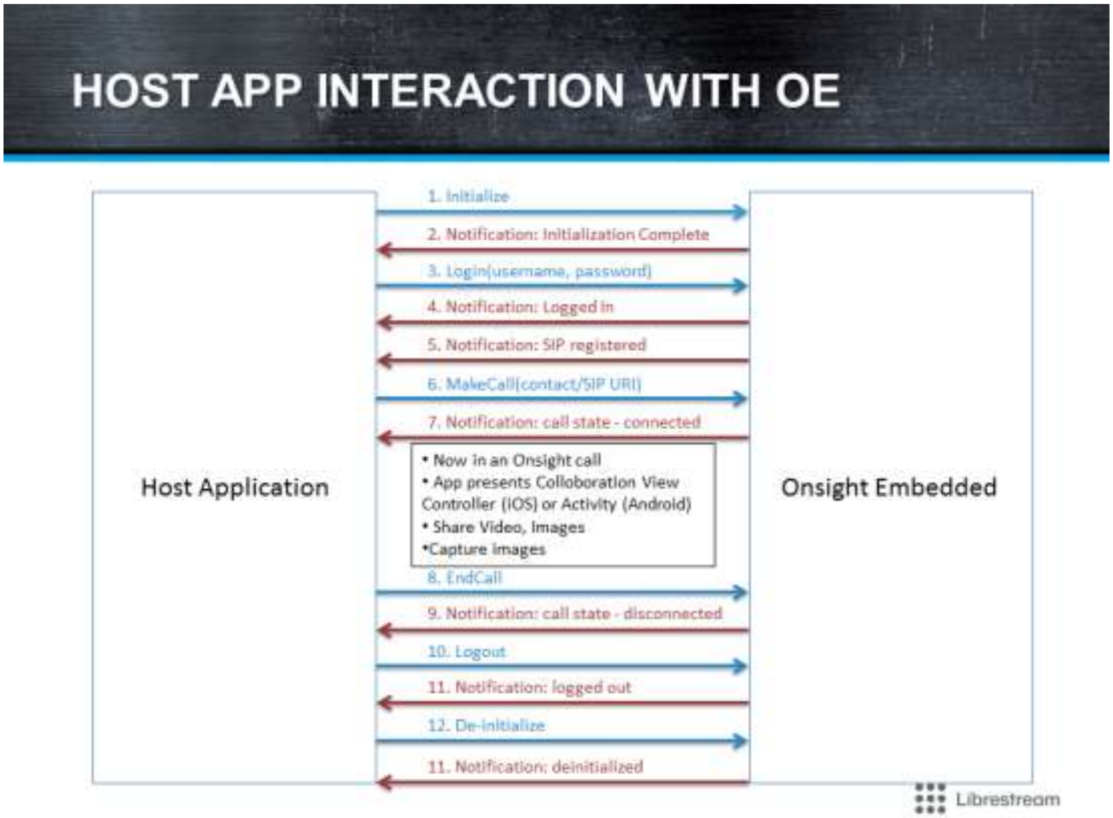
## 3.3 Overlaying UI Controls

The OclActivity allows the HostApp to overlay other UI controls on top of the viewfinder OpenGL view. This can be accomplished by creating an activity that extends OclActivity. Within the derived activity, overlayed UI controls can be added and their events handled. The OESample app that comes with the OCL contains an OclTestActivity class that shows how to do this.

# 4 Onsight Embedded Sample Application

The typical interaction between the Onsight Embedded Sample Application (HostApp) and the Onsight Embedded Library (or OCL) is detailed in this section. The HostApp provides the user interface to the Onsight Embedded Library.

The basic user interface of the HostApp is meant only to show developers how to use the Onsight Embedded Library API. It is not meant as an example of how to create a UI.  Providing access to Onsight features through the User Interface is optional. Not all features must be accessible to the user.



HOST APP INTERACTION WITH OE

Host Application

Onsight Embedded

1. Initialize
2. Notification: Initialization Complete
3. Login(username, password)
4. Notification: Logged in
5. Notification: SIP registered
6. MakeCall(contact/SIP URI)
7. Notification: call state - connected

- Now in an Onsight call
- App presents Collaboration View Controller (iOS) or Activity (Android)
- Share Video, Images
- Capture images

8. EndCall
9. Notification: call state - disconnected
10. Logout
11. Notification: logged out
12. De-initialize
11. Notification: deinitialized

## 4.1 Setup

The HostApp must be initialized (Init) and a user account logged into the Server (Login) before the remaining UI elements become active. Status is displayed at the bottom of the screen. The Onsight Connect logs may be exported to the Documents folder.

## 4.2 Onsight Call Process

1. Launch the HostApp.
2. Press Init.
3. The HostApp instantiates the OclInterface object.
4. The HostApp instantiates an OclViewController object, passing it the OclInterface object created in the previous step.
5. The HostApp calls the OclInterface::initialize: method to initialize the OCL.
6. The HostApp waits for the OclInitializeNotification::OclInitId_Initialized notification to indicate that the OCL is ready to go.
7. Enter the Server login information, (the login information is provided by the OAM Administrator):
8. Server: oam.test.librestream
9. User Name: username
10. Password: password or time limited Token (received from OAM Token Service)
11. Press Login.
12. The HostApp calls the OclInterface::loginWithUsername:token: method using the OAM Username and OAM Password or Token as parameters.
13. The HostApp presents status UI indicating that the OAM Login is progressing (using OclLoginStateNotifcation)
14. Once the login state OclLoggedIn state is reached, the OCL internally initiates the SIP registration process with SIP credentials received from the OAM server after login.
15. The HostApp waits for the OclSipRegistrationStateNotification:: **OclSip**Registered notification which indicates that the SIP registration process is complete.
16. At this point, the OCL is ready for calls: the HostApp can make calls by invoking the OclInterface::makeCallWithUri: method. Or the HostApp can also accept incoming calls with the acceptCallWithHandle: method.

17. Enter a Contact: Manual SIP URI as a parameter e.g. john.smith@sip.librestream.com.
18. Press Call.
19. TheOclCallStateNotification notification can be used to track changes in call state for either situation, including OclCallIncoming state which notifies the HostApp of incoming calls.
20. The HostApp watches for the OclCallStateNotification:: **OclCall**InProgress notification and takes note of the call handle.
21. The HostApp watches for the OclCallStateNotification:: **OclCall**Connected notification and displays the OclViewController when it is received.
22. Call Established
23. The remote participant initiates video sharing from the OCL endpoint using the UI of the Onsight Connect for Windows software.
24. The remote participant captures still images using the UI controls of the Onsight Connect for Windows software.
25. End Call
26. The OCL participant presses the Hang Up call button on the OclViewController UI,
27. The HostApp receives the OclButtonPressedNotification notification with the OclButtonId_HangUp parameter indicating that the Hang Up button was pressed.
28. The HostApp responds by calling the OclInterface::endCallWithHandle: method.
29. The remote participant presses the Hang Up button in the Onsight Connect for Windows UI.
30. The HostApp receives the OclCallStateNotification:: **OclCall**Disconnected notification at which point it removes the OclViewController from the display.
31. Logout
32. The HostApp calls the OclInterface::logout: method to initiate the process of SIP unregistration and OAM logout.
33. The HostApp waits for the OclLoginStateNotification::**OclNotLoggedIn** notification to indicate that the logout is complete.
34. Deinit
35. The HostApp calls the OclInterface::deinitialize: method to de-initialize the OCL.
36. The HostApp waits for the OclInitializeNotification::OclInitId_Deinitialized notification, at which point the use of the OCL is complete.

## 4.3 Power Management

To facilitate receiving calls when the device has gone into a sleep or power-saving mode (which typically occurs shortly after the device's screen has turned off), the OCL will periodically wake-up the device so that it can maintain its various network connections (SIP, OAM, etc). In order for receiving calls while sleeping to function properly, the following conditions must be met:

- The HostApp must include the WAKE_LOCK permission in its AndroidManifest.xml file. This allows the OCL to obtain a partial wake lock in order to allow it enough time to complete any required network transactions.

- Advanced power saving options that disable network interfaces during sleep must be turned off. For example, the Keep Wi-Fi on during sleep setting should be set to Always if you wish to receive calls over Wi-Fi while the device is sleeping.

- When an incoming call arrives, the OCL will notify the HostApp. It is up to the HostApp to process the incoming call, including turning on the screen so the user can accept the call, if required. For example, the HostApp could show an incoming call Activity that causes the screen to turn on by ensuring the appropriate flags are set:

```
getWindow().addFlags(
        WindowManager.LayoutParams.FLAG_DISMISS_KEYGUARD |
        WindowManager.LayoutParams.FLAG_SHOW_WHEN_LOCKED |
        WindowManager.LayoutParams.FLAG_TURN_SCREEN_ON);
```

If receiving calls while the device is asleep is required, regardless of which power saving options are enabled on the device, the HostApp can optionally choose to hold its own full or partial wake locks to keep the device fully awake at all times. Note, however, that doing so may have a significant impact on battery life.

### 4.3.1 Doze Mode in Android 6.0

Android 6.0 (Marshmallow) introduced a power-saving feature called Doze mode, which is meant to reduce battery consumption when the device is unused for long periods of time. When the device is in Doze mode, the HostApp's access to the network is disabled, and the HostApp will not be permitted to hold the partial wake locks described in the previous section.

As a result, in order to support receiving calls when the device has gone to sleep on Android 6.0, the HostApp needs to be whitelisted in order to be exempt from Doze mode restrictions. To configure whitelisting, the HostApp can either:

- Fire the ACTION_IGNORE_BATTERY_OPTIMZATION_SETTINGS intent, which will bring the user to the Battery Optimization screen. From there, they will be able to add the HostApp to the system's whitelist.

- Include the REQUEST_IGNORE_BATTERY_OPTIMIZATIONS permission in the HostApp's AndroidManifest.xml file. This will allow the HostApp to instead fire the ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS intent, which will prompt the user to whitelist the HostApp in a dialog without ever leaving the app. *Note that in order to include this permission in your HostApp, it must meet certain conditions outlined by Google.* More information about these conditions, and about Doze mode in general, can be found in the Android developer documentation:

  http://developer.android.com/training/monitoring-device-state/doze-standby.html

## 4.4 Onsight Features

The basic user interface of the HostApp is provided as an example to show developers how to use the Onsight Embedded Library API. It is not meant as an example of how to create a UI.  Providing access to Onsight features through the User Interface is optional. Not all features must be accessible to the user.

1. BAS Configuration: Set the state of BAS for a call – Disabled, Enabled or Cellular Only.
2. Use Names in Image Name.
3. Delete Images after Send.
4. Save Images to Camera Roll.
5. Burn Telestration before Save.
6. Pinch and Zoom Enabled.
7. Enable GPS Location Updates.
8. Guest Invites:
   a. Guest First Name
   b. Guest Last Name
   c. Invite Method: Email or SMS
   d. Guest Email
   e. Guest Phone
   f. Call me After Login
   g. Account Expiry
   h. Message to Guest
   i. Send Invitation
9. Change Password
   a. Old Password
   b. New Password
   c. Confirm Password
10. Change the Sharing Image
11. Status Messages

Dump Logs into Documents – creates a folder which contain the OCLib log files. See pkg/Android/data/com.librestream.oesample/files/Documents/logs. *These logs can be provided to Librestream to help with troubleshooting.*