ON SIGHT

USER MANUAL

Onsight Embedded for iOS SDK
Software Version 7.1.10

March 2016

LIBRESTREAM

LIBRESTREAM.COM

Table of Contents

Document Revision

Librestream

Onsight Connect for Windows Release Notes

Doc #: 400230-02

March 2016

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

iOS 8 is the minimum required operating system for running applications created with the Onsight Embedded for iOS Library. The Onsight Embedded Library is distributed as a framework bundle and an Onsight Embedded sample application, OESample.

Onsight Embedded relies on having an active Onsight Account Manager domain with available user licenses. Onsight Embedded for iOS SDK requires minimum Xcode version 7.

The library also depends on many iOS system frameworks, such as AVFoundation, CoreVideo, OpenGLES, Security, and UIKit. For a complete list of required frameworks, see the Link Binary with Libraries list within the Build Phases section of the OESample target project settings.

1.3 Target Equipment

Onsight Embedded is intended for application development for iOS version v8.0 or higher on the following iOS devices:

iPod

iPod 5th and 6th generation

iPad
iPad Air, iPad 2, 3, and 4, iPad mini 2, 3, and 4

iPhone
iPhone 4S, 5, 5c, 5s, 5se
iPhone 6, 6 Plus, 6s and 6s Plus

2 OclInterface Class

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

The API is an Objective-C interface layer that gives enough functionality for the Host Application (HostApp) to invoke and interact with the embedded OCL. The HostApp instantiates an OclInterface object and all communication with the OCL happens through it.

2.1 OclInterface Enums

The following enums are used by the OclInterface class.

2.1.1 EOclSipRegistrationState

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

2.1.1.1 OclSipRegistering

The SIP registration process starts after the OCL reaches the OclLoggedIn state. The OclSipRegistering state indicates that the SIP registration process has begun.

2.1.2 OclSipRegistered

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

2.1.3 OclSipRegisterFailed

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.

2.1.4 OclSipRegisterFailedRetrying

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

2.1.5 OclSipUnregistered

SIP registration has moved to the OclSipUnregistered state. This can occur in normal operation when the SIP call is complete and the HostApp calls the logout method because it 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.

2.1.6 EOclCallState

The EOclCallState enum identifies the SIP call states.

2.1.7 OclCallDisconnected

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 OclCallDisconnected.

2.1.8 OclCallIncoming
The OclCallIncoming 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 acceptCallWithHandle: method is used. If the user decides to decline the call, the HostApp would call the endCallWithHandle: method.

2.1.9 OclCallInProgress
The OclCallInProgress call state indicates that an outgoing call is being made.

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

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

2.1.12 OclCallDisconnecting
The OclCallDisconnecting 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 OclCallDisconnected state.

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

2.1.14 OclVideoStartRequested
The video stream is about to be started.

2.1.15 OclVideoStarted
The video stream is up and running.

2.1.16 OclVideoStopRequested
The video stream is about to be stopped.

2.1.17 OclVideoStopped
The video stream has been stopped.

2.1.18 OclVideoError
The video stream encountered an error during starting or stopping.

2.1.19 EOclReturnValue
The following are possible return values from the OCL.

2.1.20 OclSuccess
The method call was successful.

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

2.1.22 OclNotSipRegistered
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 EOclSipRegistrationState:: OclSipRegistered state.

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

2.1.24 OclInvalidParameter
The method call was not allowed because on of the parameters was invalid.

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

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

2.1.27 OclMissingPermission
OCLib cannot be used because of a missing permission. See the Permission section below.

2.1.28 OclRemoteCaptureNotAllowed
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.

2.1.29 EOclAppStateChange
The OCL needs to know when the HostApp is changing application states. The HostApp must watch for application state changes with the AppDelegate life cycle methods (applicationWillResignActive, applicationDidEnterBackground, etc.) and inform the OCL of these changes via the OclInterface::appStateTransition: selector using the EOclAppStateChange enum as a parameter.

2.1.30 OclDidBecomeActive
The HostApp has become active, via the AppDelegate::applicationDidBecomeActive: life-cycle method.

2.1.31 OclWillResignActive
The HostApp is about to become inactive, via the AppDelegate:: applicationWillResignActive: life-cycle method.

2.1.32 OclDidEnterBackground
The HostApp has entered the background state, via the AppDelegate:: applicationDidEnterBackground: life-cycle method.

2.1.33 OclWillEnterForeground
The HostApp is about to return to the foreground, via the AppDelegate:: applicationWillEnterForeground: life-cycle method.

2.1.34 OclWillTerminate
The HostApp is about to terminate, via the AppDelegate:: applicationWillTerminate: life-cycle method.

2.1.35 EOclInitId
The OCL can be in the following three states:

2.1.36 OclInitId_Initialized
The OCL is initialized.

2.1.37 OclInitId_Deinitialized
The OCL is not initialized.

2.1.38 OclInitId_Failure
The OCL initialization attempt failed.

2.1.39 EOclErrorId
The OCL errors that could occur:

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

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

2.1.42 OclError_AudionInitFailure
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 or facetime 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.

This error can also occur if the HostApp has not been given permission to use the Microphone. In this case, the remote participant's voice audio is audible, but the Microphone isn't functional so the local participant cannot be heard by the remote participant.

2.1.43 OclErrorId_CameraInitFailure
This error happens when an attempt to use the phone's camera hardware is unsuccessful, usually meaning that permission to use the camera has been denied or there is something wrong with the camera hardware (a reboot of the phone is recommended).

2.1.44 OclErrorId_CameraRestored
If the previous CameraInitFailure error occurs and subsequent access to the camera hardware is successful, this OclErrorId_CameraRestored event (not really an "error") informs the HostApp that the camera hardware access has been restored.

2.1.45 OclErrorId_AudioInterruptBegan
If an Onsight call (including voice communication) is in progress and a cell phone or facetime call is made or received, OCL sends the OclErrorId_AudioInterruptBegan event to inform the HostApp that the **Onsight's** use of the audio hardware has been interrupted by the cell phone or facetime call, since cell phone and facetime 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.

2.1.46 OclErrorId_AudioInterruptEnd
If an Onsight call has had its voice communication interrupted by a cell phone or facetime call (see previous section), once the cell phone or facetime call disconnects, the OclErrorId_AudioInterruptEnd 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.

2.1.47 OclErrorId_InBackgroundCallReminder
As part of **Onsight's** Call Continuity feature, an Onsight call can be connected without voice communication. While in this state, if the HostApp is put into the background, OCL informs the HostApp of this condition periodically using the OclErrorId_InBackgroundCallReminder event (not really an "error"). This happens because of iOS's background processing restrictions which limit the amount of time that a background app can stay active to 180 seconds. After 180 seconds in the background, the HostApp will suspend (loses access to the CPU) and the connected Onsight call will be starved of CPU time and eventually disconnect. The HostApp creators are encouraged to use local notifications that encourage the user to bring the HostApp back into the foreground within the 180 second limit so that the Onsight call doesn't prematurely disconnect.

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

This event can also occur if the HostApp had previously received the OclErrorId_AudioInitFailure event because permission to use the Microphone had not been granted, but then subsequently permission to use the Microphone has been granted by the user.

2.1.49 EOclEndCallReason
When an OclCallStateNotification::OclCallDisconnected event is received, the disconnect message contains a parameter that specifies the reason why the call ended.

2.1.49.1 OclEndCallUnknown
The reason for the call disconnect could not be determined.

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

2.1.49.3 OclEndCallNotAvailable
The remote participant was not available.

2.1.49.4 OclEndCallBusy
The remote participant was already in a call.

2.1.49.5 OclEndCallDeclined
The remote participant declined the call.

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

2.1.49.7 OclEndCallShuttingDown
The call was disconnected because the OCL is shutting down.

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

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

2.1.49.10 OclEndCallNetworkError
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.

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

2.1.50.1 OclNotLoggedIn
The OclNotLoggedIn 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.

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

2.1.50.3 OclLoggedIn
The OclLoggedIn 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.

2.1.51 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 OclNotLoggedIn to OclLoggingIn. Then if the login fails, the login state moves from OclLoggingIn back to OclNotLoggedIn. Accompanying this login state notification is a 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.

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

2.1.52.1 OclBasDisabled
The OclBasDisabled value indicates that BAS is disabled.

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

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

2.1.52.4 OclBasUseDefault
The OclBasUseDefault 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.

2.1.53 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 roll when the saveImagesToCameraAlbum property is set to ON.

The enum contains the following values:

2.1.53.1 OclCaptureStart
The OclCaptureStart value indicates that an image capture has been started.

2.1.53.2 OclCaptureComplete
The OclCaptureComplete value indicates that an image capture into the app's sandbox has successfully completed.

2.1.53.3 OclCaptureError
The OclCaptureError value indicates that an image capture has failed.

2.1.53.4 OclCaptureCameraRollError
The OclCaptureCameraRollError value indicates that, after the capture completed, the attempt to write the image to the camera roll failed.

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

2.1.54.1 OclCaptureNone
The OclCaptureNone value indicates that there is no additional information.

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

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

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

2.1.54.5 OclCaptureErrorSandboxCameraFailure
The OclCaptureErrorSandboxWriteFailure 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.

2.1.54.6 OclCaptureErrorSandboxWriteFailure
The OclCaptureErrorSandboxWriteFailure value indicates the 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.

2.1.54.7 OclCaptureErrorCameraRollUnknown
The OclCaptureErrorCameraRollUnknown value indicates that the captured still image could not be written to the camera roll for an unknown reason.

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

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

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

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

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

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

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

2.1.55.4 OclSystemHealthBad
The OclSystemHealthBad 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.

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

2.1.56.1 OclInviteInternalError
The OclInviteInternalError value indicates that there is an internal error.

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

2.1.56.3 OclInviteEmailAlreadyExists
The OclInviteEmailAlreadyExists value indicates that the email address is already registered.

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

2.1.56.5 OclInviteSipAccountPoolEmpty
The OclInviteSipAccountPoolEmpty value indicates that SIP account pool is empty.

2.1.56.6 OclInviteInvalidExpiryTime
The OclInviteInvalidExpiryTime value indicates that the expiry time is invalid.

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

2.1.56.8 OclInviteSipServerTemporarilyUnavailable
The OclInviteSipServerTemporarilyUnavailable value indicates that the SIP server is temporarily unavailable.

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

2.1.56.10 OclInviteInvalidPhoneNumber
The OclInviteInvalidPhoneNumber value indicates the phone number is invalid.

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

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

2.1.57.1 OclChangePasswordInternalError
The OclChangePasswordInternalError value indicates that there is an internal error.

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

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

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

2.1.57.5 OclChangePasswordInvalidResetPasswordCode
The OclChangePasswordInvalidResetPasswordCode value indicates that the reset password code is invalid.

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

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

2.1.57.8 OclChangePasswordAccountLocked
The OclChangePasswordAccountLocked value indicates that the OAM account is locked.

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

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

2.1.60 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:

- OclImageShareStopped – no image sharing going on, default state
- Someone requests image sharing:
  - If locally (via the shareImage method): OclImageShareRequestedLocal
  - If remotely: OclImageShareRequestedRemote
- OclImageShareStarted – the image is transferred, and both participants are now viewing the same image

- OclImageShareStopped – either participant has ended image sharing (locally via the stopImageSharing method)

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

2.1.60.2 OclImageShareRequestedLocal
The OclImageShareRequestedLocal 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 OclImageShareStarted; otherwise, the state changes to OclImageShareStopped.

2.1.60.3 OclImageShareRequestedRemote
The OclImageShareRequestedRemote 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 OclImageShareStarted; otherwise, the state will change to OclImageShareStopped.

2.1.60.4 OclImageShareStarted
The OclImageShareStarted 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.

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

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

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

2.1.61.2 OclImageShareErrorCopyFailed
The OclImageShareErrorCopyFailed 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.

2.1.62 EOclButtonId
The EOclButtonId enum is used as part of the OclButtonPressedNotification event to indicate which OclViewController button was pressed. The HostApp is responsible for handling the button press.

2.1.62.1 OclButtonId_HangUp
The OclButtonId_HangUp enum indicates that the hang up button was tapped.

2.2 OclInterface Methods

The OclInterface class contains the following methods.

Note: All OCL methods calls must be made on the HostApp's main thread. The iOS function performSelectorOnMainThread can be used if needed. The OCL is not architected to support simultaneous method invocations from different threads.

2.2.1 init
The init: method is called to initialize a new OclInterface object immediately after memory for the object has been allocated.

- (id) init

2.2.1.1 Parameters
- None

2.2.1.2 Return Value

- An initialized object.

2.2.1.3 Discussion

The init: method is used along with the alloc: method to create an instance of the OclInterface class as follows:

OclInterface *ocl = [[OclInterface alloc] init];

2.2.2 initialize:loginServer

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:(OclViewController *)oclViewController loginServer:(NSString *) loginServer

2.2.2.1 Parameters

- oclViewController:
    - The HostApp has previously instantiated an OclViewController object which must be passed as a parameter to this selector.
- loginServer
    - [optional] The Onsight Account Manager (OAM) server to login to when making a call to loginWithUsername:. 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.

2.2.2.2 Return Value

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

2.2.2.3 Discussion

The initialize: method is asynchronous. The OclInitializeNotification::OclInitId_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 OclNotInitialized.

This method allocates resources needed by the OCL. 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.

NOTE: The initialize: method appears to be taking less time than expected so option #2 is recommended because it minimizes resource use when the OCL is not in use.

2.2.3 deinitialize

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

- (EOclReturnValue) deinitialize

2.2.3.1 Parameters

- None

2.2.3.2 Return Value

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

2.2.3.3 Discussion
This function frees resources that were allocated by the initialize: method or accumulated during normal library use. Once this function is called, the OCL can no longer be used. The deinitialize: method is asynchronous. The OclInitializeNotification::OclInitId_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.

2.2.4 makeCallWithUri:reconnect
The makeCallWithUri: method initiates a SIP call from the local device to a remote endpoint.

```
- (EOclReturnValue) makeCallWithUri:(NSString *)remoteUri reconnect:(bool)reconnect
```

2.2.4.1 Parameters
- remoteUri:
  - The SIP URI of the remote endpoint.
- 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. If the remoteUri parameter does not match with the previous call, this parameter has no effect. If reconnect is not present, the default is FALSE.

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

2.2.4.3 Discussion
The makeCallWithUri: method kicks off the asynchronous process of making a call. The HostApp is notified of changes in the call state via the OclCallStateNotification notification. Once the OclCallStateNotification indicates that the OclCallConnected state is reached, the OclViewController should be displayed by the HostApp. See the Notifications and OclViewController sections below for details.

2.2.5 makeCallWithContact:reconnect
The makeCallWithContact: method initiates a SIP call from the local device to a remote endpoint.

```
- (EOclReturnValue) makeCallWithContact:(OclContact *)callee reconnect:(bool)reconnect
```

2.2.5.1 Parameters
- 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 address property of the callee parameter. If the address property does not match with the previous call, this parameter has no effect. If reconnect is not present, the default value is FALSE.

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

2.2.5.3 Discussion
The makeCallWithContact: method kicks off the asynchronous process of making a call. The HostApp is notified of changes in the call state via the OclCallStateNotification event. Once the OclCallStateNotification indicates that the OclCallConnected state is reached, the OclViewController should be displayed by the HostApp. See the Notifications and OclViewController sections below for details.

2.2.6 acceptCallWithHandle
The acceptCallWithHandle: method is used to accept an incoming call.

```
- (EOclReturnValue) acceptCallWithHandle:(UInt32)callHandle
```

2.2.6.1 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.

2.2.6.2 Return Value

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

2.2.6.3 Discussion

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

2.2.7 endCallWithHandle

The endCallWithHandle: method ends an existing call, or declines an incoming call.

- (EOclReturnValue) endCallWithHandle:(UInt32)callHandle

2.2.7.1 Parameters

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

2.2.7.2 Return Value

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

2.2.7.3 Discussion

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

2.2.8 appStateTransition

The appStateTransition: method is used by the HostApp to inform the OCL of changes to the application's state.

- (EOclReturnValue) appStateTransition:(EOclAppStateChange)change

2.2.8.1 Parameters

- change:
    - The type of application state transition that HostApp is going through represented by the enum EOclAppStateChange.

2.2.8.2 Return Value

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

2.2.8.3 Discussion

The HostApp must watch for application state changes with the AppDelegate life cycle methods (applicationWillResignActive, applicationDidEnterBackground, etc.) and inform the OCL of these changes via the appStateTransition: method.

2.2.9 loginWithUsername:password

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

-(EOclReturnValue) loginWithUsername:(NSString *)username password:(NSString *)password

2.2.9.1 Parameters

- username:

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

2.2.9.2 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 OclNotLoggedIn.

2.2.9.3 Discussion

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

When the login succeeds, the HostApp receives an OclLoginStateNotification with the OclLoggedIn 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 EOclSipRegistrationState::OclSipRegistered state is reached, the OCL is in a state in which a call can be attempted.

2.2.10 loginWithUsername:token

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

```
-(EOclReturnValue) loginWithUsername:(NSString *)username token:(NSString *)token
```

2.2.10.1 Parameters

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

2.2.10.2 Return Value

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

2.2.10.3 Discussion

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

When the login succeeds, the HostApp receives a OclLoginStateNotification notification with the OclLoggedIn 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 EOclSipRegistrationState::OclSipRegistered state is reached, the OCL is in a state in which a call can be attempted.

2.2.11 logout

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

```
-(EOclReturnValue) logout
```

2.2.11.1 Parameters

- none

2.2.11.2 Return Value

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

2.2.11.3 Discussion

The method is asynchronous and if it returns OclSuccess, the HostApp needs to listen to the OclLoginStateNotification for the result of the logout attempt. If the method does not return OclSuccess, 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 OclNotLoggedIn state is reached. The HostApp monitors the SIP registration state with the OclSipRegistrationStateNotification, and the login state with the OclLoginStateNotification.

2.2.12 overrideUserNames:first:last

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

- (EOclReturnValue) overrideUserNames:(NSString *)first last:(NSString *)last

2.2.12.1 Parameters

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

2.2.12.2 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 YES.

2.2.13 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

2.2.13.1 Parameters

- none

2.2.13.2 Return Value

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

2.2.13.3 Discussion

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

When the imageCapture method is successfully invoked, the OCL initiates an image capture and reports its progress via OclCaptureStateNotification. The HostApp will receive EOclCaptureState::OclCaptureStart when the image capture starts, EOclCaptureState::OclCaptureComplete when it ends, and EOclCaptureState::OclCaptureError if an error occurs. Additionally, if the saveImageToCameraAlbum property is ON, the image is saved to the camera roll. If this process fails, there is an EOclCaptureState::OclCaptureCameraRollError notification indicating the error cause.

2.2.14 inviteGuestWithEmail

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

-(EOclReturnValue) inviteGuestWithEmail:(NSString *)firstName lastName:(NSString *)lastName address:(NSString *)emailAddress message:(NSString *)message expiryHours:(UInt32)expiryHours ignoreDuplicateEmail:(BOOL)ignoreDuplicateEmail callInviterAutomatically:BOOL(callInviterAutomatically)

2.2.14.1 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

2.2.14.2 Return Value

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

2.2.14.3 Discussion

The method sends an invite guest request to the OAM server. If it returns OclSuccess, 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 OclInviteGuestNotification. If the result is OclInviteSuccess, 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 value is an error code indicating what went wrong.

2.2.15 inviteGuestWithSms

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

-(EOclReturnValue) inviteGuestWithSms:(NSString *)firstName lastName:(NSString *)lastName address:(NSString *)phoneNumber message:(NSString *)message expiryHours:(UInt32)expiryHours callInviterAutomatically:BOOL(callInviterAutomatically)

2.2.15.1 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

2.2.15.2 Return Value

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

2.2.15.3 Discussion

The method sends an invite guest request to the OAM server. If it returns OclSuccess, 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 OclInviteGuestNotification. If the result is OclInviteSuccess, 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 value is an error code indicating what went wrong.

2.2.16 changePassword

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

-(EOclReturnValue) changePassword:(NSString *)userName oldPassword:(NSString *)oldPassword newPassword:(NSString *)newPassword

2.2.16.1 Parameters

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

2.2.16.2 Return Value

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

2.2.16.3 Discussion

The method sends a change password request to the OAM server. If it returns OclSuccess, 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 OclChangePasswordNotification. If the result is OclChangePasswordSuccess, the OAM server has successfully changed the password for OAM user. Any other result value 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 if 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 OclLoginPasswordExpired. 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 OclChangePasswordSuccess, after which the guest user is allowed to login again with the new password.

2.2.17 dumpLogsIntoDirectory

The dumpLogsIntoDirectory method is used by the HostApp to create a directory and place the OCLib Log files into the specified directory.

-(EOclReturnValue) dumpLogsIntoDirectory:(NSString *)directory

2.2.17.1 Parameters

- directory:
    - the name of the directory

2.2.17.2 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.
    - OclInternalError – an internal error has occurred.

2.2.17.3 Discussion
The method creates the directory and places the log files into it. If it returns OclSuccess, the directory was created and log files were placed within it. Any other return value is an error code indicating what went wrong. Log files are needed only if you request troubleshooting assistance from Librestream otherwise they are not necessary to access.

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

- (EOclReturnValue) shareImage:(NSString *)filename;

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

2.2.18.2 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

2.2.18.3 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 OclImageShareStarted 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.

2.2.19 stopImageSharing
The stopImageSharing method is used to end image sharing.

- (EOclReturnValue) stopImageSharing

2.2.19.1 Parameters
- None

2.2.19.2 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 OclImageShareStopped state.

2.2.19.3 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 acn be drawn on the image if the image is shared again.

2.3 OclInterface Properties

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

2.3.1 initialized
@property (readonly) BOOL initialized

2.3.1.1 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.

2.3.2 sipRegistrationState
@property (readonly) EOclSipRegistrationState sipRegistrationState

2.3.2.1 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 OclSipUnregistered.

2.3.3 callState
@property (readonly) EOclCallState callState

2.3.3.1 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 OclCallDisconnected.

2.3.4 videoState
@property (readonly) EOclVideoState videoState

2.3.4.1 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 OclVideoStopped.

2.3.5 version
@property (readonly) NSString * version

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

2.3.6 deleteImagesAfterTransfer
@property (readwrite) BOOL deleteImagesAfterTransfer

2.3.6.1 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.

2.3.7 useFirstLastNameInImageName
@property (readwrite) BOOL useFirstLastNameInImageName

2.3.7.1 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.

2.3.8 saveImagesToCameraAlbum

@property (readwrite) BOOL saveImagesToCameraAlbum

2.3.8.1 Discusson

The saveImagesToCameraAlbum property controls whether captured still images are saved to the Camera Roll Album.

The initial value is FALSE.

2.3.9 burnTelestrationOnCameraAlbumImages

@property (readwrite) BOOL burnTelestrationOnCameraAlbumImages

2.3.9.1 Discussion

The burnTelestrationOnCameraAlbumImages property controls whether the captured image's telestration is burned onto the image before they are saved into the Camera Roll Album. 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. The Camera Roll images do not support telestration save in the Exif tag so if this property is set, the telestration is burned onto the image. If the property is set to FALSE, the telestration does not appear on the Camera Roll image and can no longer be retrieved.

The initial value is TRUE.

2.3.10 loginState

@property (readonly) EOclLoginState loginState

2.3.10.1 Discusson

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

The initial value is OclNotLoggedIn.

2.3.11 basConfiguration

@property (readwrite) EOclBandwidthAdaptiveStreaming basConfiguration

2.3.11.1 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 OclBasUseDefault, 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 OclBasEnabledForCellularOnly.

2.3.12 callInfo

@property (readonly) OclCallInfo callInfo

2.3.12.1 Discussion

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

- NSString *callerName – name of the caller
- NSString *callerAddress – address of the caller
- BOOL incoming – TRUE for incoming calls, FALSE for outgoing calls
- BOOL encrypted – TRUE for encrypted calls, FALSE otherwise
- BOOL remoteRecording – TRUE if the caller is recording, FALSE if not
- NSString *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.

2.3.13 systemHealth

@property (readonly) OclSystemHealth systemHealth

2.3.13.1 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 remove management service

2.3.14 enableGpsLocationUpdates
@property (readwrite) BOOL enableGpsLocationUpdates

2.3.14.1 Discussion
The enableGpsLocationUpdates property controls whether to save location information (GPS coordinates) into the Exif tags of captured images.

The default value is FALSE.

2.3.15 personalContacts
@property (readonly) NSArray *personalContacts

2.3.15.1 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 the personalContacts property has a new array of OclContact objects with the changes included.

2.3.16 imageShareState
@property (readonly) EOclImageShareState imageShareState

2.3.16.1 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.

2.4 OclInterface Notifications

The OCL uses the iOS NSNotificationCenter to notify the HostApp about interesting events, status and errors that occur in the library. The userInfo NSDictionary contains parameters that are specific for each notification.

Note: All OCL notifications are posted on the HostApp's main thread.

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

The userInfo parameter contains the following keys:
- kOclSipRegistrationStateKey: This key identifies the new SIP registration state, and 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 EOclSipRegistrationState::OclSipRegistered state is reached, the OCL is ready to accept incoming or make outgoing SIP calls.

2.4.2 OclCallStateNotification

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

The userInfo parameter contains the following keys:

- kOclCallStateKey: This key identifies the new call state, and can take one of the values of the enum EOclCallState.
- kOclCallHandleKey: This key identifies the call handle for the specific call to which this notification refers. The call handle is used as a parameter for acceptCallWithHandle: and endCallWithHandle: methods. The call handle is first identified to the HostApp in the OclCallIncoming notification for incoming calls, and in the OclCallInProgress notification for outgoing calls.
- kOclCallEndReasonKey: This key is present in the OclCallStateNotification if the kOclCallStateKey is set to EOclCallState::OclCallDisconnected, which signifies the end of a call. It gives a reason why the call ended and takes one of the values of EOclEndCallReason.
- kOclCallerNameKey: This key identifies the name of the caller and is only present if the kOclCallStateKey is set to EOclCallState::Incoming.
- kOclCallerAddressKey: This key identifies the address of the caller and is only present if the kOclCallStateKey is set to EOclCallState::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.

- EOclCallState::OclCallConnected: When this notification arrives, the HostApp should display the OclViewController. See the OclViewController section below.
- EOclCallState::OclCallIncoming: When the OclCallIncoming 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 acceptCallWithHandle: or endCallWithHandle: methods to accept or decline the incoming call. This is the first time that the HostApp sees the call handle (via the kOclCallHandleKey) for incoming calls. It should be saved at this point so that the HostApp can work with the call (like declining or accepting it).
- EOclCallState::OclCallInprogress: When the OclCallInprogress notification arrives, if the HostApp has made the outgoing call, this is the first time that the HostApp sees the call handle (via the kOclCallHandleKey). It should be saved at this point so that the HostApp can work with the call (like cancelling it).

2.4.3 OclVideoStateNotification

When the OclViewController starts streaming video to the remote endpoint, the HostApp is notified using OclVideoStateNotification notification. The kOclVideoStateKey key in the userInfo parameter identifies the new video stream state, and can take one of the values of the enum EOclVideoState.

2.4.4 OclButtonPressedNotification

The OclViewController UI contains a hang-up button. When the hang-up button is pressed and released, the HostApp is notified with the OclButtonPressedNotification notification.

The kOclButtonIdKey key in the userInfo parameter identifies which button was pressed and released. The value can be one of the following:

- OclButtonId_HangUp – The hang-up button was pressed and released. The HostApp handles the hang up button press. It may want to put up a confirmation or it may want to disable the hang-up button for periods of time. Whatever the case, when the time is right, HostApp must initiate the hang-up action by calling the endCallWithHandle: method. If there is no extra processing required, the OclButtonId_HangUp button would simply call the endCallWithHandle: method.

2.4.5 OclErrorNotification

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

The kOclErrorIdKey key in the userInfo parameter identifies the specific error. The value can be one of the following:

- OclErrorId_NetworkNotFound - The OCL posts this error to inform the HostApp that there are no WiFi or cellular networks available for the OCL to use.

- OclErrorId_NetworkError – The OCL posts this error to indicate that a call has ended because the network that the call had been using has been lost.

2.4.6 OclInitializationNotification

The OclInitializationNotification notification identifies the initialization state of the OCL.

The kOclInitIdKey key in the userInfo parameter identifies the specific state. The value can be on of the following:

- OclInitId_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.
- OclInitId_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.
- OclInitId_Failure – The OCL posts this notification to indicate that the initialization of the OCL has failed.

2.4.7 OclLoginStateNotification

The OclLoginStateNotification identifies the login state of the OCL.

The userInfo parameter is a dictionary of values identified by the following keys:

- The kOclLoginStateKey identifies the new login state that was just reached and takes one of the values of the EOclLoginState enum.
- The kOclLoginReasonKey identifies further information about the login state change, and takes one of the values of the EOclLoginReason enum.
- The kOclLoginStatusCodeKey identifies a standard HTTP status code if the kOclLoginReasonKey is equal to EOclLoginReason::OclLoginHttpCode or an internal server status code if the kOclLoginReasonKey is equal to EOclLoginReason::OclLoginServerCode. For all other values in kOclLoginReasonKey, this parameter is irrelevant.

2.4.8 OclCaptureStateNotification

The OclCaptureStateNotification identifies the image capture state of the OCL as identified by the capture state key, which takes one of the values of EOclCaptureState enum.

The userInfo parameter is a dictionary of values identified by the following keys:

- The kCaptureStateKey identifies the new image capture state that was just reached and takes one of the values of the EOclCaptureState enum.
- The kCaptureFileNameKey identifies the file name of the captured image if the kCaptureState is equal to EOclCaptureState::Complete. For all other values in kCaptureStateKey, this parameter is irrelevant.
- The kCaptureStatusCodeKey identifies the initiator (local or remote) of the image capture if the kCaptureStateKey is equal to EOclCaptureState::Start, or an error code if it's equal to EOclCaptureState::Error. For all other values in kCaptureStateKey, 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 ON, the OCL will also save the image into the camera roll. If this process gets an error, the app will receive an OclCaptureStateNotification event with the capture state set to OclCaptureCameraRollError.

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 roll.

2.4.9 OclSystemHealthNotification

The OclSystemHealthNotification occurs when some part of the system health has changed. The kOclSystemHealthkey key in the userInfo parameter identifies the current system health. You can also query the systemHealth property and compare it to the previous value to see what has changed.

2.4.10 OclRemoteVersionNotification

The OclRemoteVersionNotification occurs when the remote app version number has been obtained. The kOclRemoteVersionKey key in the userInfo parameter identifies the version number of the remote endpoint. You can also query the appVersion field of the callInfo property.

2.4.11 OclRemoteRecordNotification

The OclRemoteRecordNotification occurs when the remote participant's recording state has changed. The kOclRemoteRecordKey key in the userInfo parameter identifies the state of remote record. You can also query the remoteRecording field of the callInfo property and compare it to the previous value to see if the state has changed.

2.4.12 OclInviteGuestNotification

The OclInviteGuestNotification occurs when the OAM server has processed a guest invitation request. The kOclInviteGuestResultKey key in the userInfo parameter identifies the result of a guest invitation request and takes one of the values of the EOclInviteGuestResult enum.

2.4.13 OclChangePasswordNotification

The OclChangePasswordNotification occurs when the OAM server has processed a change password request. The kOclChangePasswordResultKey key in the userInfo parameter identifies the result of a guest invitation request and takes one of the values of the EOclChangePasswordResult enum.

2.4.14 OclImageShareStateNotification

The OclImageShareStateNotification event occurs when there is a change in the image share state. The kOclImageShareStateKey in the userInfo parameter identifies the new image sharing state (EOclImageShareState enum). The image sharing state can always be checked with the imageShareState property.

2.4.15 OclImageShareNewImageNotification

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 file name of the new shared image is retrieved from the userInfo parameter using the kOclImageShareFileNameKey.

2.4.16 OclImageShareErrorNotification

The OclImageShareErrorNotification event occurs when there is an error somewhere in the image sharing process. The error code is available from the userInfo parameter using the kOclImageShareErrorCodeKey.

2.4.17 OclPersonalContactsUpdatedNotification

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 the personalContacts property.

2.4.18 OclSettingChangedNotification

The OclSettingChangedNotification event informs the HostApp that one of the settings type properties have changed internally within OCL. The kOclSettingChangedKey within the userInfo parameter indicates which setting has changed.

2.5 Permissions

The OCL requires the following permissions and/or features:

- Required background modes:
  - Voip
  - Audio
- Required device capabilities:
  - Armv7
  - Forward-facing camera – only in an effort to eliminate having to support under-powered iDevices
- Application uses WiFi
- Location services – if GPS or access to Camera Roll is required (Production requirements)

- NSLocationWhenInUseUsageDescription or NSLocationAlwaysUsageDescription – in iOS8 and later, if GPS is required, include one of these keys in Info.plist to describe the reason for using GPS information. The first key is for accessing GPS info only when the app is in the foreground, and the latter is for accessing GPS info when the app is in the foreground or background.
  - If the HostApp requires GPS location information, the HostApp also needs to call CLLocationManager::requestWhenInUseAuthorization or CLLocationManager::requestAlwaysAuthorization, depending on which key is used in the Info.plist file. These selectors are only available in iOS8 and later so a call to respondsToSelector is necessary to avoid crashing in older OS versions.
- In iOS8 and later, an application must first ask for permission before using notifications to inform the user of some event when the application is in the background. Therefore, if the HostApp is using OCLib and wants to inform the user of incoming calls when the HostApp is in the background, the HostApp must ask the user for notification permission by making a call to [[UIApplication sharedApplication] registerUserNotificationSettings:] during startup. This selector is only supported in iOS8 and later so a call to respondsToSelector is necessary to avoid crashing in older OS versions.
- In iOS8, the user must give the HostApp permission to use the camera. If OCLib does not have access, the initialize selector returns OclMissingPermission. The HostApp can check permission using [AVCaptureDevice authorizationStatusForMediaType:AVMediaTypeVideo], and request permission using [AVCaptureDevice requestAccessForMediaType: AVMediaTypeVideo]. The first time requestAccessForMediaType is called (or the camera hardware is accessed in OCLib), the user will be presented with an OS dialog asking for permission. Subsequent calls will not result is this dialog being presented. If the user says "no", the HostApp can see this by calling the authorizationStatusForMedia selector and looking for a AVAuthorizationStatusDenied status. Since OCLib is nearly useless without camera access, the HostApp may want to ask the user to go to the **HostApp's** privacy settings and enable Camera access. A few more notes:
  - Changing any privacy setting for HostApp will cause the OS to immediately terminate HostApp.
  - For testing purposes, privacy settings can be reset in Settings->General->Reset->Reset Location & Privacy.

3 OclViewController

The OclViewController is a full screen view controller that presents a minimalist Onsight Connect collaboration screen. When the OclCallConnected state is reached, the HostApp should make the view controller active, such that it appears on the display.

The OclViewController has the following visual components:

- When the view controller 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 view controller 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 view controller displays a set of status icons that indicate the following:
  - 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 iOS 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 OclViewController showing progress as the still image is captured and transferred from the iOS device to the remote participant.

3.1 OclViewController Methods

The OclViewController class contains the following methods.

3.1.1 init

The init: method is called to initialize a new OclViewController object immediately after memory for the object has been allocated.

- (id) init:(OclInterface *)ocl

3.1.1.1 Parameters
- ocl:
    - An instantiated OclInterface object through which the OclViewController talks to the rest of the OCL

3.1.1.2 Return Value
- An initialized object.

3.1.1.3 Discussion
The init: method is used along with the alloc: method to create an instance of the OclViewController class as follows:

```
OclInterface *ocl = [[OclInterface alloc] init];
OclViewController *vc = [[OclViewController alloc] init:ocl];
```

3.1.2 setInForeground
The setInForeground: method is called to inform the OclViewController when the application goes into and out of foreground execution mode.

- (void) setInForeground:(BOOL)foreground

3.1.2.1 Parameters
- foreground:
    - TRUE if the application is entering the foreground, FALSE if the application is leaving the foreground.

3.1.2.2 Return Value
- None

3.1.2.3 Discussion
The setInForeground: method is used to inform the OclViewController that the application has entered or left foreground mode. This allows the controller to enable and disable its internal OpenGL drawing engine and to conserve resources when not needed. This method should be called from the AppDelegate's applicationWillResignActive: method with FALSE as the parameter. It should also be called from the AppDelegate's applicationDidBecomeActive: method with TRUE as the parameter.

3.1.3 configureControls
The configureControls method is called to inform the OclViewController that it should configure the state of its internal controls based on the current state of the OCL interface.

- (void) configureControls

3.1.3.1 Parameters
- None

3.1.3.2 Return Value
- None

3.1.3.3 Discussion
The configureControls method is primarily used by the OCL interface to tell the OclViewController that it should update its internal state based on the interface's current state. It is called whenever significant state events happen within the interface. While not really intended for external use, it won't do any harm to call this method at any time during the execution of the client program.

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

- (void) showModal:(UIView *)view

3.1.4.1 Parameters

- view:
  - The view to be displayed modally.

3.1.4.2 Return Value

- None

3.1.4.3 Discussion

The showModal: method is used to display a popup modal view over top of the OclViewController. While the view is displayed, all other controls on the screen are grayed out and disabled. Tapping the screen anywhere outside the view will cause it go away.

3.1.5 hideModal

The hideModal: method is called to hide a specific modal view. If that view is NOT the current modal view as displayed by the showModal: method, no action is taken.

- (void) hideModal:(UIView *)view

3.1.5.1 Parameters

- view:
  - The modal view to be hidden.

3.1.5.2 Return Value

- None

3.1.5.3 Discussion

The hideModal: method is used to hide a specific modal popup view as displayed by the showModal: method. If the view passed into the method is the current modal view, it is hidden. If it isn’t the current modal view, no action is taken.

3.1.6 hideModal

The hideModal method is called to hide the current modal view.

- (void) hideModal

3.1.6.1 Parameters

- None

3.1.6.2 Return Value

- None

3.1.6.3 Discussion

The hideModal: method is used to hide the current modal popup view as displayed by the showModal: method. If there is no current modal view, no action is taken.

3.1.7 showMessage:title:buttons:timeout:defaultResult

The showMessage: method is called to display a popup alert. It provides several standard sets of buttons that can be displayed and it returns a value indicating which button was pressed to dismiss the alert. A timeout can also be used to dismiss the alert after a certain amount of time, regardless of whether or not a button has been pressed. This method does not return until one of the buttons is pressed or the timeout occurs.

- (EAlertResult) showMessage:(NSString *)message title:(NSString *)title buttons:(EAlertButtons)buttons timeout:(double)timeout defaultResult:(EAlertResult)defaultResult

3.1.7.1 Parameters

- message:
  - The message to be displayed within the alert box.
- title:

  - The title to be displayed at the top of the alert box.
- buttons:
  - Specifies which set of buttons should be displayed within the alert box and must be one of the following:
    - AlertButtonsOK
    - AlertButtonsOKCancel
    - AlertButtonsAbortRetryIgnore
    - AlertButtonsYesNoCancel
    - AlertButtonsYesNo
    - AlertButtonsRetryCancel
- timeout:
  - The number of seconds that the method should wait for the user to press one of the buttons. If this time period expires, the defaultResult value is returned. If this value is zero, no timeout will occur.
- defaultResult:
  - The value to be returned if a timeout occurs.

3.1.7.2 Return Value

- The return value indicates which button was pressed to dismiss the alert. If the alert was dismissed by a timeout, the defaultResult value is returned.

3.1.7.3 Discussion

The showMessage: method displays a popup alert and waits for the user to press one of the buttons. An optional timeout value can be used to return a default result after waiting the specified period without the user pressing a button.

If showMessage: (or showError:) is called while a previous call to either is still displaying its alert, the first alert will be dismissed before showing the new one. In other words, you can only have one alert on the screen at a time.

3.1.8 showMessage:buttons:timeout:defaultResult

The showMessage: method is called to display a popup alert. This version of the method uses the title as specified by the messageTitle property rather than passing a specific title to this method. See the documentation for the full version of showMessage: for further specifics and discussion.

- (EAlertResult) showMessage:(NSString *)message buttons:(EAlertButtons)buttons timeout:(double)timeout defaultResult:(EAlertResult)defaultResult

3.1.9 showMessage:buttons:defaultResult

The showMessage: method is called to display a popup alert. This version of the method uses the title as specified by the messageTitle property and a timeout value of zero (thus disabling the timeout functionality). See the documentation for the full version of showMessage: for further specifics and discussion.

- (EAlertResult) showMessage:(NSString *)message buttons:(EAlertButtons)buttons defaultResult:(EAlertResult)defaultResult

3.1.10 showMessage

The showMessage: method is called to display a popup alert. This version of the method uses the default title (as specified by the messageTitle property), a simple OK button (as specified by AlertButtonsOK), and a timeout value of zero (thus disabling the timeout functionality), . See the documentation for the full version of showMessage: for further specifics and discussion.

- (EAlertResult) showMessage:(NSString *)message

3.1.11 showError:buttons:timeout:defaultResult

The showError: method is called to display a popup alert. This version of the method uses the title as specified by the errorTitle property rather than passing a specific title to this method. See the documentation for the full version of showMessage: for further specifics and discussion.

- (EAlertResult) showError:(NSString *)message buttons:(EAlertButtons)buttons timeout:(double)timeout defaultResult:(EAlertResult)defaultResult

3.1.12 showError:buttons:defaultResult
The showError: method is called to display a popup alert. This version of the method uses the title as specified by the errorTitle property and a timeout value of zero (thus disabling the timeout functionality). See the documentation for the full version of showMessage: for further specifics and discussion.

- (EAlertResult) showError:(NSString *)message buttons:(EAlertButtons)buttons defaultResult:(EAlertResult)defaultResult

3.1.13 showError
The showError: method is called to display a popup alert. This version of the method uses the default title (as specified by the errorTitle property), a simple OK button (as specified by AlertButtonsOK), and a timeout value of zero (thus disabling the timeout functionality), . See the documentation for the full version of showMessage: for further specifics and discussion.

- (EAlertResult) showError:(NSString *)message

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

- (int) addWarning:(NSString *)text

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

3.1.14.2 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.

3.1.14.3 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.

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

- (void) removeWarning:(int)warningId

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

3.1.15.2 Return Value
- None

3.1.15.3 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.

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

- (void) removeAllWarnings

3.1.16.1 Parameters
- None

3.1.16.2 Return Value

- None

3.1.16.3 Discussion

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

3.2 OclViewController Properties

The OclViewController class has the following properties giving easy access to its internal subviews.

3.2.1 oclInterface

@property (retain) OclInterface *oclInterface

3.2.1.1 Discussion

The oclInterface property gives the HostApp access to the OclInterface object that was passed into the init: method when the OclViewController was created.

3.2.2 renderer

@property (retain) OclRenderer *renderer

3.2.2.1 Discussion

The renderer property gives the HostApp access to the video renderer that is displayed within the OclViewController. The renderer is an OpenGL view that's used to display all video and images. It fills the OclViewController's view and supports the following gestures:

- pinch to zoom,
- double-tap to return the zoom to normal,
- panning a zoomed image using two fingers,
- drawing telestration using one finger.

All other views appear on top of the renderer.

3.2.3 hangupButton

@property (retain) UIButton *hangupButton

3.2.3.1 Discussion

The hangupButton property gives the HostApp access to the hang-up button that is displayed in the top-right corner of the OclViewController. This allows the HostApp to do anything it wants with the hang-up button, from changing the button graphics, to hiding the button, to overriding the event handlers for the button.

3.2.4 statusBar

@property (retain) OclStatusBar *statusBar

3.2.4.1 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 OclViewController. 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:(UIImage *)sharingImage

3.2.4.2 Parameters

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

3.2.4.3 Discussion

In order for the image to fit properly within the status bar, it should have dimensions of 24x24 for non-retina screens and 32x32 for retina screens.

3.2.5 messageTitle

@property (retain) NSString *messageTitle

3.2.5.1 Discussion

The messageTitle property specifies the title to be used with the showMessage: methods. This is a convenience property that simplifies calling showMessage:. It should be set once at the start of your program.

3.2.6 errorTitle

@property (retain) NSString *errorTitle

3.2.6.1 Discussion

The errorTitle property specifies the title to be used with the showError: methods. This is a convenience property that simplifies calling showError:. It should be set once at the start of your program.

3.2.7 modalView

@property (retain) UIView *modalView

3.2.7.1 Discussion

The modalView property is the current modal popup view as displayed by the showModal: method. If there is no current modal view, this property is nil.

3.2.8 enableControls

@property (retain) BOOL enableControls

3.2.8.1 Discussion

The enableControls property allows the host app to disable all of the controls displayed within the OclViewController, regardless of the current state of the OCL interface. If this property is set to TRUE or YES, the state of the controls is determined by the OCL interface state. If that property is set to FALSE or NO, all of the controls are disabled.

3.2.9 pinchAndZoomEnabled

@property (readwrite) BOOL pinchAndZoomEnabled

3.2.9.1 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.

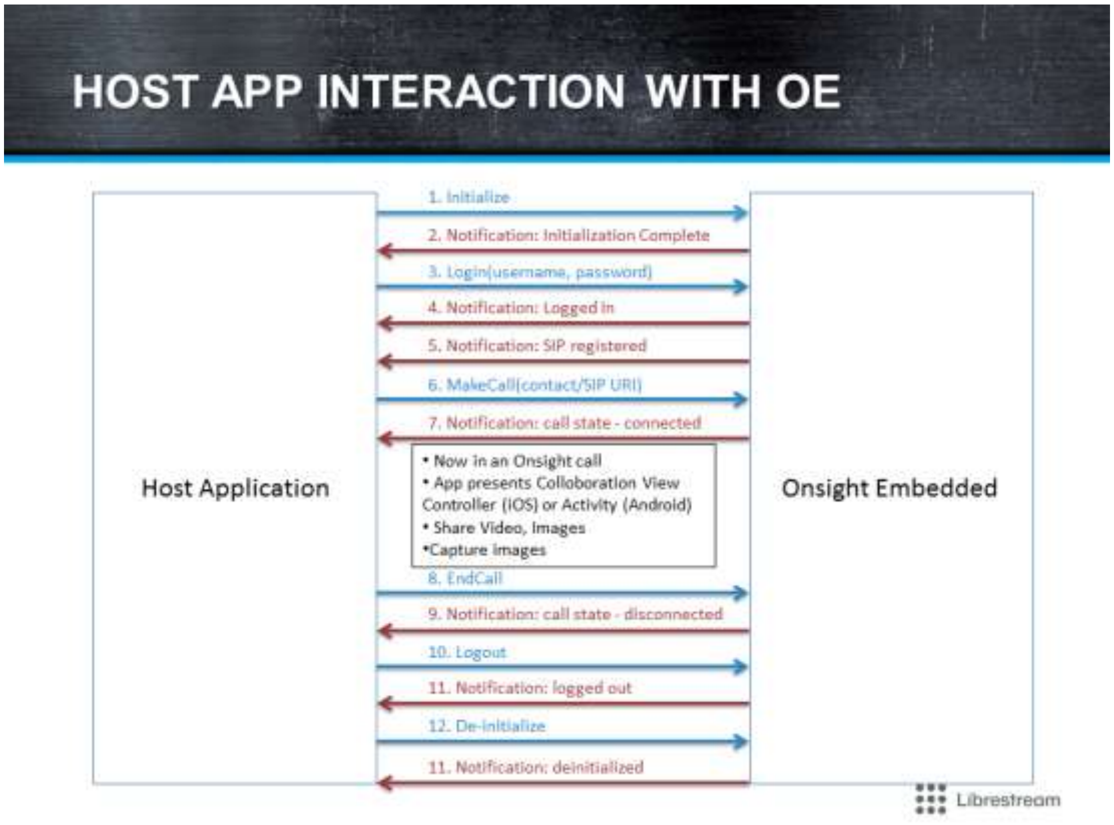
3.3 Overlaying UI Controls

The OclViewController allows the HostApp to overlay other UI controls on top of the viewfinder OpenGL view. This can be accomplished by creating a view controller that inherits from OclViewController. Within the derived view controller, overlayed UI controls can be added and their events handled. The OESample app that comes with the OCL contains an OclTestViewController 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
Librestream

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.

   i. The HostApp instantiates the OclInterface object.
   ii. The HostApp instantiates an OclViewController object, passing it the OclInterface object created in the previous step.
   iii. The HostApp calls the OclInterface::initialize: method to initialize the OCL.
   iv. The HostApp waits for the OclInitializeNotification::OclInitId_Initialized notification to indicate that the OCL is ready to go.
3. Enter the Server login information, (the login information is provided by the OAM Administrator):
   a. Server: oam.test.librestream
   b. User Name: username
   c. Password: password or time limited Token (received from OAM Token Service)
4. Press Login.
   i. The HostApp calls the OclInterface::loginWithUsername:token: method using the OAM Username and OAM Password or Token as parameters.
   ii. The HostApp presents status UI indicating that the OAM Login is progressing (using OclLoginStateNotifcation)
   iii. 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.
   iv. The HostApp waits for the OclSipRegistrationStateNotification:: OclSipRegistered notification which indicates that the SIP registration process is complete.
5. 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.
6. Enter a Contact: Manual SIP URI as a parameter e.g. john.smith@sip.librestream.com.
7. Press Call.
   i. TheOclCallStateNotification notification can be used to track changes in call state for either situation, including OclCallIncoming state which notifies the HostApp of incoming calls.
   ii. The HostApp watches for the OclCallStateNotification:: OclCallInProgress notification and takes note of the call handle.
   iii. The HostApp watches for the OclCallStateNotification:: OclCallConnected notification and displays the OclViewController when it is received.
8. Call Established
   i. The remote participant initiates video sharing from the OCL endpoint using the UI of the Onsight Connect for Windows software.
   ii. The remote participant captures still images using the UI controls of the Onsight Connect for Windows software.
9. End Call
   i. The OCL participant presses the Hang Up call button on the OclViewController UI,
   ii. The HostApp receives the OclButtonPressedNotification notification with the OclButtonId_HangUp parameter indicating that the Hang Up button was pressed.
   iii. The HostApp responds by calling the OclInterface::endCallWithHandle: method.
   iv. The remote participant presses the Hang Up button in the Onsight Connect for Windows UI.
   v. The HostApp receives the OclCallStateNotification:: OclCallDisconnected notification at which point it removes the OclViewController from the display.
10. Logout
   i. The HostApp calls the OclInterface::logout: method to initiate the process of SIP unregistration and OAM logout.
   ii. The HostApp waits for the OclLoginStateNotification::OclNotLoggedIn notification to indicate that the logout is complete.
11. Deinit
   i. The HostApp calls the OclInterface::deinitialize: method to de-initialize the OCL.
   ii. The HostApp waits for the OclInitializeNotification::OclInitId_Deinitialized notification, at which point the use of the OCL is complete.

4.3 Onsight Features

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.

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:
    i. Guest First Name
    ii. Guest Last Name
    iii. Invite Method: Email or SMS
    iv. Guest Email
    v. Guest Phone
    vi. Call me After Login
    vii. Account Expiry
    viii. Message to Guest
    ix. Send Invitation
9. Change Password
    i. Old Password
    ii. New Password
    iii. Confirm Password
10. Change the Sharing Image
11. Status Messages
12. Dump Logs into Documents – creates a folder which contain the OCLib log files. The HostApp can make this folder Public so that it is accessible through iTunes or provide another mechanism such as ftp or email for access. *These logs can be provided to Librestream to help with troubleshooting.*