C&C Android Change List

5.0.1

- Refactored the HLS parser used within the SDK to make it better support fastplay HLS assets with any internal structure of sub-manifests.
- Changed the behavior of the fastplay flag in the Asset builders so that an asset created for fastplay will support only fastplay, not download, but with improved performance. Assets created for fastplay cannot be added to the download queue.
- The SDK now requires Kotlin code support and has added this to the gradle dependencies.

5.0.0

- Added support for API31 (Android 12). This is a breaking change for the SDK integration. Android 12 requires background download to be implemented using WorkManager while a foreground service approach will still be used when available.
- This version of the SDK must be built with API31 and WorkManager version 2.7+.
- The new meta-data key “com.penthera.virtuososdk.notification.provider.impl” must be added to the Android manifest to provide the name of a class implementing the IForegroundNotificationProvider interface. This class will be used within both the client application and service process to provide UI notifications for the service during download. For more details, please refer to the documentation.
- Updates to support Exoplayer version 2.15

4.0.3

- Fix potential race condition is SDK startup on slower devices.
- Fix reported background error in SDK startup when file system not yet reported as available.
- Add support for Exoplayer 2.15.*. There is a new support library (cnc-exoplayer-2_15-support) available for using Exoplayer 2.15.*.
- Deprecated support for Exoplayer versions 2.11.* and 2.12.* as those versions are no longer available via maven repository.
- Added asset information to the manifest rendition selection callbacks for use in determining correct rendition for the specific asset.
- Analytics fixes.

4.0.2

- Added an interface for advanced selection of video renditions from HLS and DASH manifests. The

interface for both manifest types can be accessed via the IBackgroundProcessingManager, which the SDK instantiates within the download service process. If the interface is registered with the SDK then all selection of video renditions is passed to the client.

- Added settings via the ISettings interface to enable clients to set string properties on the MediaDrm instances within the SDK, or specify the security level when opening MediaDrm sessions (session security available on API28+).
- Fixed an issue where download could generate a file size mismatch error upon retrying a download, based upon zero length sent in the headers of chunked transfer.
- Added support for Android WorkManager2.5 Configuration.Provider initialization.
- Fixed an issue where the Expire After Download settings were not being properly applied in some instances.
- Update SDK to latest Firebase Cloud Messaging version and prevent issues when a push provider is not present in implementing app.
- Add ability to select video stream by resolution.
- Fix an issue where HLS sub-manifest with 302 redirect urls were not processed properly.
- Add handling for exceptions reported by VirtuosoClock on device shutdown.
- Analytics fixes.

4.0.1

- Additional support for exoplayer versions and features
    - ‘cnc-android-exoplayer-2_12-support now includes support for the Exoplayer playlist and MediaItem api.
    - 'cnc-android-exoplayer-2_13-support' provides support for the Exoplayer 2.13 SDK version. It includes helper utility methods that can create a player pre-configured with DRM and client side advertising support for an SDK asset or list of assets in a playlist.
    - 'cnc-android-exoplayer-2_14-support' provides support for the Exoplayer 2.14 SDK version. It includes helper utility methods that can create a player pre-configured with DRM and client side advertising support for an SDK asset or list of assets in a playlist.
- Analytics fixes

4.0.0

- To better support the incompatible changes within exoplayer versions, player support has been removed from the main gradle distribution library and included within separate extension libraries:
    - 'cnc-android-exoplayer-2_11-support' provides legacy support for Exoplayer versions 2.11

and below. This provides components to interface with the SDK DRM solution that match those in 3.17, but requires a change in the package name of included SDK components.

    - 'cnc-android-exoplayer-2_11-support' also provides some new helper methods that will create a Player object pre-configured to play an SDK Asset, including DRM configuration.
    - 'cnc-android-exoplayer-2_12-support' provides support for integrating the Exoplayer 2.12+ SDK versions. It includes helper utility methods that can create a player pre-configured with DRM and client side advertising support for an SDK asset.
    - The 2.12 support library also includes support for using Exoplayer playlists containing SDK assets and offline Widevine DRM.
    - A support library has also been added for BitMovin player to simplify integrations.
- Updated the SDK Android Manifest to encapsulate more of the SDK components and permissions so they do not need declaring in the application manifest. The download service (VirtuosoService), HTTP proxy service, and service broadcast receivers should not be declared in the application AndroidManifest unless the app wishes to override the default process names.
- Provided a gradle plugin that can optionally be used to automatically generate the SDK content provider for an application and include it in the AndroidManifest. This option requires adding a gradle plugin and removing any legacy SDK component from the manifest.
- Altered the internal structure of the SDK so that the Service Starter component no longer needs to be created within a separate process. It can share the main UI process.
- A new interface has been provided to simplify FCM based push notification integration for apps which already implement their own push notifications. The legacy implementation relies upon deriving from an SDK class whereas the new interface can be called from within an existing FirebaseMessagingService instance.
- A new support library has been created to help simplify the SDK integration for applications utilizing LiveData from Android Jetpack. The support library provides LiveData derived classes to deliver lists of assets, engine status, and autodownload playlists. This provides an alternative to cursors for populating lists of SDK assets.
- Updated the API behavior of the ISettings interface such that the save method no longer needs to be called after making changes.
- Updated to WorkManager v2.5.0 and implemented RemoteWorkManager-based cross-process support for scheduling of timed tasks from the download service.
- Improved progress estimation of HLS manifest based assets containing a single video segment by pre-fetching the segment sizes prior to download.
- Added an isPlayable method to IAsset to match the equivalent method of the iOS SDK.
- Added setters for expiry (including Expiry After Play / Expiry After Download) to the asset builders for creation of new assets.
- Minor behavioral change to autodownload behavior to ensure a playlist asset is only considered

complete if download finishes prior to the asset being deleted.

- Upgraded an old encryption algorithm that was used internally for protecting HLS encryption keys that were saved for offline playback. This prevents warnings from the Play store submission process about using unsafe encryption. This is only relevant when using HLS content protected by encryption keys and not DRM.
- Fixed an upgrade issue that could occur when restoring asset permissions for an asset downloaded in an earlier version of the SDK.
- Fixed an issue where ampersand characters in DASH SegmentTemplate manifests were incorrectly escaped when creating the paths for storage and playback.
- Updated DRM key extraction code to safely ignore some non-android stream types that may appear in streams delivered for multiple platforms. Also improved performance to minimize the amount of the stream which is downloaded to find the DRM keys.

3.17.2

- Fix an issue where multiple adaptation sets within an MPEG DASH manifest which contained audio tracks for the same language and codec were reduced to a single set to save download space. The previous implementation did not check if the sets differed in other aspects such as Accessibility or Role parameters, which would result in valid cases of audio tracks in the same language. In this case a representation from each set will now be downloaded.

3.17.1

- Provide a mechanism to allow a client to update segment URLs directly before download, so that if a URL contains session based parameters then the client can update the segments if the original session has expired. The new functionality requires the creation of a class implementing the IPrepareURLObserver interface, which is returned from a IBackgroundProcessingManager instance registered in the Android manifest.
- Update the permissions flow, so that an asset in an error state will keep the permission assigned to it until either the error is fixed or the asset is deleted. Previously, the SDK would return any permission allocations from such an asset to be used by another asset or device.
- Update SDK startup so that any change of backplane URL, in addition to user name and keys, causes the SDK to reset previously stored downloads. Previously, the backplane URL could be changed upon startup and if the other details were unchanged then the SDK would erroneously keep any existing assets from the previous account.
- Fix an issue where query parameters which use the forward slash (/) character contained within URLs extracted from MPEG DASH manifests could cause local paths to be generated incorrectly for storage of the segments.
- Fix some issues with backplane permission requests for business rules that could cause a device to

miscount allocated permissions and temporarily permit more or less asset downloads than the setting until the next scheduled sync.

- Fix a number of internal issues relating to correct generation of analytics events.

3.17.0

- API Change: IAsset#getPlaylist() has been replaced by IAsset#getPlaybackURL() to bring the API inline with the iOS platform and prevent any confusion with the new autodownload feature.
- Added the method IAssetManager#deleteAssets(List<IAsset>) to make it easier to delete multiple assets in a single call.
- Added the AutoDownload feature to the SDK. This provides interfaces for IPlaylist, IPlaylistItem and IPlaylistManager which is used to create and maintain playlists.
- Updated the android WorkManager dependency to version 2.4.0
- Added the resolution of the downloaded video to each asset.
- Added some protection for database rollback moving forward from version 3.17.0, such that if an older SDK version is installed over a newer one the SDK will intelligently try to recover and downgrade the database. In many cases this may require deleting all current assets depending on the downgrade. This will not take effect if an SDK version prior to 3.17.0 is installed over a newer version.
- Updated IAsset#getFractionComplete() to limit the resolution to 3 decimal places, or 0.1% intervals. This prevents the reported fraction appearing noisy during very early downloading.
- Fixed a crash which occurred due to a synchronization issue on concurrent modification of SDK settings.
- Fixed a crash which occurs in the downloader if an internal error is thrown during integrity checking of downloaded assets before they are marked as complete. The crash resulted from a null pointer exception where the integrity result was unset.
- Fixed an issue that occurs intermittently on old Android OS versions when cleaning up the secure clock. The previous implementation could result in an exception thrown within the GC.
- Removed deprecated Subscriptions feature from SDK.

3.16.7

- Fixed an issue where playback was not working correctly for HLS manifests that used absolute URLs in the EXT-X-MAP tag data.

3.16.6

- Fixed an issue that occurred when an asset which had been placed in the external lock denied state was reset or added back to the download queue and was subsequently permitted to download. This erroneously resulted in the state being transitioned to early downloading, and not the full downloading state which subsequently resulted in the download never being able to finish.
- Fixed an issue where an asset which was denied permission to download due to a DRM failure at the start of download was being placed into a different queue state to other denied assets. A DRM related failure (upon setting ISetting#setErrorAssetOnDRMFail()) will remain in the queue, with retry count set to max retries. It can be reset by refreshing the license.
- Fixed an issue that occurred when an asset which had been placed in the external policy denied state was reset or added back to the download queue and was subsequently permitted to download. This erroneously resulted in the state being transitioned to early downloading, and not the full downloading state which subsequently resulted in the download never being able to finish.
- Fixed an issue where the DASH parser did not recognize codec definitions in the AdaptationSet, and only accepted them from the Representation. This prevented correct filtering by codec for some DASH manifests.
- Fixed an issue where adding an asset with the synchronous methods in IAssetManager would not correctly add ancillary files that were defined in the asset params.
- Fixed an issue with the generation of proxy manifests for DASH where a tagged ancillary file in the asset could get incorrectly included as an entry in the manifest.
- Fixed an issue with handling iframes in HLS manifests where the generated manifest would include a reference to the original external URL, preventing offline playback.

3.16.5

- Removed uses of androidx LocalBroadcastManager from the SDK to remove that dependency as it has been deprecated for some time. If you used the broadcast demonstrated in the player demo to detect changes to the proxy port upon re-entering the app during playback, then this needs to be replaced with the new Observer callback as now demonstrated in that example.
- Deprecated manual sync request from backplane API.
- Improved logic for the timing of automated sync requests to ensure consistent minimum (15 minutes) and maximum (12 hours) intervals between sync.
- Removed issue where the SDK can report sync errors if the first sync occurs before the internal secure clock has set the time.
- Fixed an issue where a leading ../ on the relative URLs for DASH segmented template caused playback issues in the local playback proxy.
- Fixed a crash that occurred if a manifest registered receiver for time changes was run when the application was not previously in memory.
- Fixed a concurrency related crash that occurred when multiple assets were added in quick succession.

- Fixed an issue that could occur when checking the authentication status of the SDK using a non-blocking call very early in the SDK startup before the database cursor was fully loaded. This has been resolved but may result in the authentication status temporarily being reported as unauthenticated during early access concurrency cases.
- Fixed an issue that occurred when parsing an HLS manifest with relative URLs that had been received following an HTTP redirect. The parser was not detecting the redirect and applying the new base URI to the relative URLs.

3.16.4

- Added an Observer (IConnectivityObserver) to register for the same connectivity notifications that the download service uses to control download over WiFi and cell networks.
- Added an extra update to the engine observer so the client is informed if the download state transitions from being blocked on one network state to another network state, such as from being blocked on no available network to blocked on cell network with no quota.
- Changed the downloader behavior upon encountering unexpected MIME types during segment downloads. The debug library still behaves as before, and will the stop the download and report the error upon encountering an issue. The release library reports error events for MIME type errors, but only blocks the download if the reported type is a text type such as HTML or XML, as these types are more likely to indicate that the request has been intercepted by a proxy.
- Changed the SDK behavior upon encountering a situation where the Penthera cloud servers report the device is not authenticated, but the device has credentials and believes it should be authenticated. The SDK will now automatically re-register, whereas previously it would report this as an error.
- Fixed an issue that could result in an ANR on occasions where the download service was woken by a connectivity change.
- Fixed an issue which affected reporting of estimated size for assets with a low segment count if a large segment failed mid-download and restarted.
- Improved reporting on some analytics events for when business rules were applied to prevent downloads.
- Removed the deprecated HSS manifest type support.
- Improved JavaDoc to link correctly to Android OS classes and cleaned up some comments.

3.16.3

- Removed a dependency on java unit test classes which was caused by a long running dependency on some support classes from OkHttp mock web server.
- Added new methods isExpired() and isWithinWindow() to the IAsset interface to simplify checking for the asset status.

- Fixed an issue in download progress reporting that caused progress to jump backwards if a download containing a low segment count was stopped and restarted. The change in progress would be negligible in assets with large segment counts.
- Fixed an issue where permissions from external policy servers were not applied correctly to non-segmented asset downloads.
- Fixed an issue which was introduced in 3.16.2 which caused expected size calculations to be reported incorrectly to the client API for non-segmented assets, or assets with very low segment counts (less than 5).

3.16.2

- Fixed an issue where device localization could affect the reporting of asset fraction complete via the cross process database Cursor.

3.16.1

- Changed DRM key management to optionally bypass license remaining duration checks on Android 8.x devices where a platform bug reports 0 duration. This feature can be turned on with a manifest meta-data declaration setting “com.penthera.virtuososdk.client.drm.android_o_license_allowed” to true.

3.16.0

- Changed Cookie Handler implementation to be entirely within the SDK service process so that it should not interfere with any cookie handling within the client application.
- Make the modular features for Subscriptions and Advertising support disabled by default, enabled via a manifest meta-data declaration.
- Renamed error count to retry count in the public API for the Asset interfaces and introduced a public constant for the maximum retries to make the operation of the SDK in retry and failure conditions clearer to the client.
- Prevent OS foreground notifications appearing on SDK resume where the service is already known to be configured and running.
- Fixed an issue with a missing resource close from a database cursor, which was introduced in 3.15.14.
- Fixed an issue where first play times were not recorded if the client did not implement the playStart event, which was introduced in 3.15.14.

3.15.15

- Added new analytics events for better recording of blocked download reasons.

- Removed a custom hostname verifier from the HTTPS connections on manifest and segment downloads. This relaxed verification on hostnames and was originally introduced for technical issues with the Android OS on a much earlier version and should no longer be necessary. This will not cause any issues providing SSL certificates are properly configured so the default Android verified accepts the certificate.
- Fixed an issue with HLS parsing where HTTP get parameters on the master manifest could cause incorrect formatting of request URLs on sub manifests.
- Fixed an issue with MPEG-DASH where HTTP get parameters on segment urls cause incorrect formatting of the manifest delivered by the playback proxy.
- Updated the dependency to the latest stable version of WorkManager.
- Fixed a number of crashes identified from custom reports in 3.15.9 which can occur during shutdown of the service and release of observers on connectivity.
- Fixed an issue introduced in 3.15.14 where the completed segment count may exceed the total segment count during download of any segmented asset.
- Protected against a crash that could occur during startup if the filesystem was unavailable.
- Improved fraction complete reporting to use the same size estimates as were fixed in 3.15.14 and not segments counts, which are not a good representation of progress.
- Fixed an issue where DRM refresh intervals are not calculated correctly following a failed refresh.
- Made a change to ensure integrity checks are run at the earliest opportunity after an asset has expired while the device is sleeping, to ensure old files are not on the device for longer than necessary.
- Fixed an issue where downloads can get permanently blocked if an initial permission request fails for network reasons and succeeds on a later attempt.
- Fixed an issue where the download service may cycle through the start/complete cycle more than once when blocked on storage.
- Fixed some logging levels to prevent unnecessary warning logs for unexceptional conditions.
- Fixed an issue which can cause crashes in error logging on optimized builds.
- Fixed issue with IAsset.getFractionComplete returning incorrect values

3.15.14

- Refactored manifest parsing such that the download engine can begin to start downloading segments before the manifest parsing has completed, resulting in the download appearing to start sooner.
- Introduced new asset states to track the asset while it is being parsed and if download has started while parsing continues. The new states are MANIFEST PARSE PENDING, MANIFEST PARSING, and EARLY DOWNLOADING. Early downloading indicates that downloading has started but that the manifest parse is incomplete; during this time the asset could still transition to an error state if the manifest parsing fails to complete, and the download will stop. During early downloading the estimated asset size cannot be trusted to be accurate as the initial estimated sizes are based upon the stream bitrates and not on the sizes of real downloaded segments. Once the downloader reaches the DOWNLOADING state it will transition to using the sizes of downloaded segments to estimate the final asset size.

- Changed SDK behavior such that assets which fail parsing are no longer removed from the asset list, and remain visible in one of two new states. The new states are MANIFEST REACHABILITY ERROR, if a manifest (or HLS sub-manifest) cannot be downloaded from the server, and MANIFEST_PARSING_ERROR if the parser cannot successfully parse the delivered manifest and determine the required segments for download.
- Refactored manifest parsing to run within the download service, in order to facilitate the features described above.
- Introduced a new interface IManifestParserObserver to replace the parser observer which could previously be passed as an argument when creating a new asset. An implementation of IManifestParserObserver must be instantiated by an IBackgroundProcessingManager instance, which in turn is registered with the SDK via the manifest. The methods defined by this interface will run within the download service process and not the UI.
- Improved asset estimated size calculations during download.
- Transitioned to AndroidX dependencies, android.arch and support libraries are no longer supported.
- Transitioned androidx.work.WorkManager to version 2.2.0
- Fixed a bug where internal URIs may not resolve during startup due to the late registration of content provider URIs in some android versions.
- Improved the secure clock implementation to fall back to alternative time sources when NTP is unavailable.
- Added an option to the SDK logging to block all logging output.

3.15.12

- Refactored some code in time parsing for HLS manifest lines to prevent proguard transformations causing errors in Java verification on Android 4.4.2.
- Removed all dontwarn entries from the customer proguard file so that customers can manage these themselves.
- Refactored some code inVirtuosoHTTPProxyService to prevent proguard issues in transforming.
- Refactored some code which was causing transformation issues when used with Firbase performance.
- Fixed some issues related to SDK startup where overuse of binder threads could result in delays to the main thread and subsequently ANRs.
- Ensure immediate retry on connectivity for backplane calls for asset deletion to minimize delay in business rule updates.
- Improve performance of HTTP proxy in playback of Dash SegmentBase streams.
- Ensure audio bitrate selection matches video bitrate selection in HLS manifests with multiple bitrates.
- Fix issue where queue order can jump to the next item when individual assets are paused and resumed.

3.15.11.1

- Added extra try/catch handlers inside VirtuosoHttpProxyService to prevent java verification issues on Android 5.1.1.

3.15.11

- Fix playback issue for very simple HLS files.
- Fix for validation failure not reporting correctly during register callback.
- Expose processing state in IAsset during deletion and expiry.
- Fix crash in blocking DASH asset creation methods.
- Remove unneeded debug logging.

3.15.10

- Fix issue with bit rate selection with certain DASH manifest formats.
- Remove SSL dependencies that could case conflicts with newer versions of OkHttp.
- Additional documentation for LanguageSettings.
- Add settings option to fail asset download hen DRM license is unavailable.
- Added default notification URL to cursors returned by IAssetProvider. This fixes an issue where the cursor was not updating as expected.
- Fix playback issue with DASH SegmentBase manifests.
- Analytics enhancements.

3.15.9

- Additional performance fixes for large manifest file playback
- Fix for CastLabs DRM request URL formatting
- Fix for crash parsing HLS EXT-X-TIMESTAMP entries on older devices
- Fix for DASH manifest parsing where filtering by codec would result in an invalid playback manifest.

3.15.8

- Improved memory footprint on app for parsing and playback to allow extremely large manifest files to download properly.
- Fixed NPE crash with createAdMediaRoot.
- Fixed crash caused ty adRefreshWorker.reschedule.
- Fix NPE in VirtuosoService.run.
- Fix NPE in RegistryInstance.set.

3.15.7

- Fixed an issue in the HTTP proxy, introduced in 3.15.4 which affected HSS playback.

- Fixed an issue in HLS playback manifest construction for manifests containing AES encryption keys, introduced in 3.15.5.
- Deprecated mime type filter settings for CC and SUBTITLE types in DASH manifests and replaced with a general TEXT type to represent both.
- Fixed an issue in the distributed proguard/R8 rules with the gradle distribution which was preventing shrinking and obfuscating of client and framework classes.

3.15.6.1

- Added mechanism to disable injection of a new global CookieHandler upon SDK initialization.

3.15.6

- Filter async notifications from the download service for assets that have been deleted.

3.15.5

- Corrected an issue upgrading to 3.15.3 and higher from early 3.15.0 builds.
- Fixed issue where Widevine license fetching would fail in some circumstances.
- Fixed issue where HLS default audio renditions were not included in the playback manifest.
- Added Widevine support for HLS manifests.
- Added API to allow updates to external ID without unregistering the device.

3.15.4

- Corrected issue in HLS parsing for media streams.
- Added support for HLS EXT-X-MAP tags.
- Added support for HLS inline encryption tags.
- Added additional error handling in content provider to address background crashes in RegistryInstance that occur due to race condition in content provider initialization.
- Correct issue with codec filtering for MPEG-DASH files.

3.15.1 -> 3.15.2 -> 3.15.3

- Corrected an issue in HLS parsing causing unexpected NPE for some types of assets.

3.15.0 -> 3.15.1

- Fix issue when adding multiple HLS assets in close sequence where manifest parsing would become deadlocked. (Pulled forward from 3.14.29)

3.14.28 -> 3.15.0

- Fastplay.
- Language filtering for audio, CC, and subtitle content streams.
- Filtering audio streams by codec type.
- New, easy to use, Builder classes streamline asset creation.
- Ability to pause/resume individual asset downloads.
- Ability to add ancillary files to asset download and retrieve them from SDK.
- Improved DRM refresh handled in background tasks including new SDK setting to enable/disable automatic DRM refresh.
- Ability to manually refresh DRM for individual assets.
- Modernize SSL handling in SDK download connections.
- Enhancements to SDK analytics to provide deeper insight into usage.
- Added api methods to access the SDK secure clock information.
- Improved error reporting for asset permissions and manifest parsing.
- Improved asset expiration handled in background tasks.
- Support for large images in SDK notifications.
- Stability fixes around service restarts.
- Added ability to specify content destination path in SDK settings.
- Added ability to configure valid mime types for segments in SDK settings.
- Clarified documentation on SDK settings.

3.14.29 -> 3.14.30

- Added additional error handling in content provider to address background crashes in RegistryInstance that occur due to race condition in content provider initialization.

3.14.28 -> 3.14.29

- Fix issue when adding multiple HLS assets in close sequence where manifest parsing would become deadlocked.

3.14.27 -> 3.14.28

- Catch a crash that can occur if http proxy status is checked during service binding.
- Remove mime type checks for initialisation blocks in dash assets.
- Fix issue with default audio streams without separate URI being declared at top of manifest in HLS manifest parsing

3.14.26 -> 3.14.27

- Fixed an issue in backplane permission checking where registering an observer for Queue time permission checks did not ensure the permissions were checked.
- Fixed a timing issue that could cause crashes in the download engine if it was restarted and shutdown by the OS simultaneously. This only occurred when changing the setting for Maximum Download Threads.
- Cleaned up some missing resource releases for the ContentProviderClient.
- Corrected service restart processing to allow a smoother restart flow under conditions where the OS started the process following a force close or crash.

3.14.25 -> 3.14.26

- Fixed an issue where some cookie persistence to transfer cookies between http requests in different processes was causing crashing upon reloading.

3.14.24 -> 3.14.25

- Fixed issue where initial backplane validation on first startup fails if secure clock time has not been established. This was blocking registration from completing successfully in some circumstances.

3.14.23 -> 3.14.24

- Update MPEG DASH parsing to work with multi-period manifests where only start or duration elements are available for each period.

3.14.22 -> 3.14.23

- Update of feature which detects specific HTTP authentication error responses to only be activated upon a setting.
- Added a backup method for fetching time when NTP is blocked by the network or the server is unavailable. This connects to a separate backplane API to fetch the server time.

3.14.21 -> 3.14.22

Features:

- Added a minor feature to detect specific HTTP custom error responses and report a separate error code back to the API.
- Fixed an issue where the downloader can be marked as blocked on cell quota when cell quota is unlimited and the download reaches the storage quota.

3.14.20 -> 3.14.21

Features:

- Fixed a timing issue which can intermittently cause assets being deleted or expired while downloading to show download stopped reasons other than the true cause.
- Fixed the queue observer interface to make the error callback engineEncounteredErrorDownloadingAsset() for all permissions related error cases.
- Added the Max copies permission rule to the list of download stopped reasons so that it will be shown in the queue observer callback.

3.14.19 -> 3.14.20

Features:

- Fixed a number of threading and timing issues related to the content provider and settings which could result in intermittent ANRs during startup or background operations.
- Changed backplane sync scheduling logic to ensure a sync occurs at least once with every 24 hour period when the app is opened.
- Fixed issue which occurred if backplane comms were completing during sdk shutdown, which could result in an intermittent crash.

3.14.18 -> 3.14.19

Features:

- Fixed an edge case introduced in 3.14.18 that can cause long parsing times in certain types of manifests.

3.14.17 -> 3.14.18

Features:

- Fixed a number of small errors relating to correctly reporting failure if parsing errors occur while assets are being queued or if an asset is deleted while it is being queued. Also ensured that no artifacts are left in the database in these failure cases.

3.14.16 -> 3.14.17

Features:

- Fixed an error when saving very large HSS or HLS manifests to the database.

3.14.15 -> 3.14.16

Features:

- Corrected an issue where expires don’t properly complete and the expiry handler begins to hard-loop attempting to expire the asset.

3.14.14 -> 3.14.15

Features:

- Corrected an issue where some internal error states for network errors could be reported as the max error state in the notifications for the asset.

3.14.13 -> 3.14.14

Features:

- Corrected an issue where some internal error states for network errors could be treated as external policy server permission errors.
- Reduced the restrictions on CnCForegroundServiceHandler::updateNotification() so that the service will accept updates to the cached notification when the service is not currently in foreground.

3.14.12 -> 3.14.13

Features:

- Corrected an issue where subtitles in some DASH manifests caused a parsing exception.

3.14.11 -> 3.14.12

Features:

- Corrected an issue where a security exception was not properly handled. The exception could occur in unusual circumstances when the SDK is starting its own services. In the event of the download service not starting, a method is called on any registered IEngineObserver to inform the main application.
- Corrected an issue where deleting the final item in the download queue whilst the item is blocked was not correctly resetting the download engine state to idle and clearing the notification.
- Added support for ttml+xml being recognized as a valid mime type for download in MPEG DASH manifests.

3.14.10 -> 3.14.11

Features:

- No new features, Corrected an issue where parsing of top level HLS manifests containing audio definitions with no url caused a crash.

3.14.9 -> 3.14.10

Features:

- Corrected additional configuration issues where invalid SSL configuration could be used on older Android devices.

3.14.8 -> 3.14.9

Features:

- Corrected an issue where invalid SSL configuration could be used on older Android devices.
- Corrected an issue where foreground service notifications could still be shown for particular use cases.
- Fixed an NPE error happening on some devices when the playback proxy service took too long starting up.

3.14.7 -> 3.14.8

Features:

- Corrected an issue where closed captions in DASH assets wouldn't download

3.14.6 -> 3.14.7

Features:

- Corrected an issue where service notification may be shown when app restarts
- Corrected an issue where SDK may crash if multiple assets parse simultaneously

3.14.5 -> 3.14.6

Features:

- Adding setting to configure download service to exit foreground mode and remove system notification when the download engine is paused, which defaults to disabled.

3.14.4 -> 3.14.5

Features:

- Added codec filtering support and multiple codec support for MPEG-DASH assets.

3.14.3 -> 3.14.4

Features:

- No SDK Changes. Corrections were made to SDK Demo build configuration.

3.14.2 -> 3.14.3

Features:

- Added ability to select which audio codecs are downloaded when multiple audio codecs are included in the same master HLS manifest.

- Added ability to support multiple simultaneous audio codecs in a master HLS manifest.
- Added new permissions model “max copies of asset per account”.

3.14.1 -> 3.14.2

Features:

- Added API/workflow to properly handle asset creation failures due to app suspension

3.14.0 -> 3.14.1

Features:

- Improved release packaging to support gradle/maven based builds
- Insure setting battery level to 0 completely disables battery checks
- Updates to packaged proguard rules to insure clean building with minify enabled without external rules

3.13.6 -> 3.14.0

Features:

- Improved download engine performance
- Added API for manually reporting play events
- Updates to support latest ExoPlayer versions for DRM support and exposing MediaDrm.OnEventListener via API
- Added support for external policy service permissions model
- Added setting to allow limiting maximum concurrent connections
- Added support for creating downloads directly from Uplynk asset data URLs
- Fix to insure notifications are updated when changes occur while downloads are paused

3.13.1 -> 3.13.6

Android 8 Support:

- Update service to use Foreground service when downloading in the background. Download service switches in and out of foreground mode when necessary. Application needs to provide the notification to display for progress updates.
- Update to use explicit broadcasts throughout SDK to support Android 8.
- Replaced Apache HTTP stack with current default Android URLConnection for main downloader and OkHttp for REST server calls.
- Update monitoring of disk space / memory / CPU usage to support Android 8.

Features:

- Support for subtitles in HSS
- Maximum Asset Download feature to limit maximum number of downloads per asset.
- Increase progress reporting to UI observers for large segments during download.
- Efficiently handle HTTP 416 responses for partial file requests
- Improve local file management to handle HLS files containing multiple folders containing files of identical names.
- Expiry of content during download if necessary.
- Added adaptive multi threading framework for downloader to optimise number of threads used for files with large segment counts and prevent file and network access from throttling download speed. Makes use of additional cores in newer devices. Greatly improved download throughput.

Bug Fixes:

- Improved calculations for fraction complete, available through asset manager.
- Fixed issue with support for really large manifests.
- Improve logging levels throughout SDK to reduce size of log files required for issue reporting.
- Prevent partial download and range requests on key file downloads to prevent encryption issues.
- Handle { and } characters in urls within manifests

3.13 -> 3.13.1

New Features:

The SDK now allows checking of the Max Account Downloads and Max Asset Downloads at the time of adding an Asset to the Queue. The “REQUIRE_PERMISSION_ON_QUEUED” Setting is governed by the backplane and comes down in the Sync.

In order to check the permission at time of addition to queue there is a new Observer Interface defined in IQueue:

```
interface IQueuedAssetPermissionObserver {
/**
 * Callback to report on Asset's permission check when added to the queue.
 * If the interface is provided to one of the Queue's Add methods then
 * it will be called regardless of the RequirePermissionWhenQueued Setting.
 * If an asset was not granted permission then the Asset will have its download status
 * set to the relevant error. Additional details can be retrieved from the Asset; it
 * holds the Backplane's permission response.
 *
 * @param aQueued True if asset was added to the queue.
 * @param aDownloadPermitted True if Asset was granted download permission. False
otherwise.
 * @param aAsset The asset which was granted / denied download permission.
 * @param aAssetPermissionError one of the IAssetPermission.PermissionCode values
 * @see IAsset#getLastPermissionResponse()
 * @see IAssetPermission
 * @see com.penthera.virtuososdk.client.IAssetPermission.PermissionCode
 * @category 003_queue
 */
void onQueuedWithAssetPermission(boolean aQueued,
                                 boolean aDownloadPermitted,
                                 IAsset aAsset,
                                 int aAssetPermissionError);
}
```

Overloaded Add methods have also been added to the IQueue interface to allow passing in the IQueuedAssetPermissionObserver. To get the details of the backplane response and a listing of devices which have registered a download for the asset use the IAsset interface and call the getLastPermissionResponse Method.

The IAssetPermisson contains an interface that details the account devices and number of Downloads registered against them (IDownloadsPerDevice and IAssetDownloadsPerDevice).

3.13 -

New Features:

The SDK now uses Firebase Cloud Messaging (FCM) instead of Google Cloud Messaging (GCM). The SDK now supports Amazon Device Messaging (ADM). Please see the Setting up Push Notifications section in the CCClientDevelopersGuideAndroid.pdf , that section provides details on what needs to be specified in the manifest. The SDK Demo application also shows how to implement and subclass the SDK components.

Support for the “Maximum Permitted Asset Downloads per Account” (MAD) rule. This rule is governed by the backplane. It can be read from the com.penthera.virtuososdk.client.IBackplaneSettings using the method getMaxDownloadsPerAsset. This rule controls the maximum number of times an asset can be downloaded across all devices associated with a user.

Support for the “Maximum Permitted Downloads per Account” (MDA) rule. This rule is governed by the backplane. It can be read from the com.penthera.virtuososdk.client.IBackplaneSettings using the method getMaxDownloadsPerAccount. This rule controls the maximum number of assets a user can have downloaded across all associated devices.

The SDK now allows Segmented Assets to complete with Errored segments. You can access and modify the number of segments which are permitted to have errors through the ISettings interface. Look for the methods: setMaxPermittedSegmentErrors and getMaxPermittedSegmentErrors

The Client HTTP proxy for playing out Segmented files will by default return a HTTP OKAY (200) and a zero length file for segments which have errored out during download. Depending on your player you may need to return a different HTTP code for playout to continue; You can modify the code through the ISettings interface. Look for the methods: setSegmentErrorHttpCode and getSegmentErrorHttpCode.

API Changes:

Class com.penthera.virtuososdk.client.Virtuoso The startup method’s signature has changed. The GCMSenderId parameter has been removed.

Interface com.penthera.virtuososdk.client.IPushRegistration This interface is passed as a parameter into the startup method. The onRegisterSuccess method has been removed. The onRegisterFailed method has been removed. The onServiceAvailabilityResponse method’s signature has been modified. It now takes 2 integer parameters that specify the type of push service and an error code. Please look at the SDK Demo and java docs for full details.

Interface com.penthera.virtuososdk.client.IEngineObserver There are now 2 types of setting change callbacks: One for settings provided by the backplane backplaneSettingChanged and the other for Settings configurable by the client settingChanged. A new set of flags com.penthera.virtuososdk.Common.BackplaneSettingFlag has been added for usage with the backplaneSettingChanged callback.

Interface com.penthera.virtuososdk.client.IAssetManager Signature of the createFileAsset method has been modified: the expected size parameter has been removed.

Class com.penthera.virtuososdk.Common.AssetStatus

Three new constants have been added to describe the status of an Asset:

- DOWNLOAD_DENIED_ACCOUNT: Asset can not be downloaded due to the Max Downloads Per Account rule.
- DOWNLOAD_DENIED_ASSET: Asset can not be downloaded due to the Max Downloads Per Asset rule.
- DOWNLOAD_BLOCKED_AWAITING_PERMISSION: Asset can not be downloaded until permission has been verified with Backplane.