Kaltura Online Video Security Capabilities

Overview of Security Capabilities

Kaltura's online video platform offers advanced video publishing, management, syndication and monetization solutions suitable for many verticals, including education, enterprise, government, media and entertainment, advertising, and many others. Kaltura's flexible platform and APIs allow publishers and organizations to rapidly, and cost-effectively, build video applications, widgets and plug-ins, as well as add core video services to their existing offerings.

Whether you publish video on the web or use it only for internal audiences, it is important for you to address the security aspects of your online video strategy. Kaltura offers various effective ways to implement the right level of security for your needs and has built in physical, architectural and applicative security measures that provide end-to-end protection for your assets and information.

Whether you choose a Software as a Service (SaaS) implementation using our scalable infrastructure and trusted CDN partner, or opt for a self-hosted and self-operated platform on your own premises (Kaltura On-Prem™), Kaltura will help you implement the security measures you need to have in place, while assuring an intuitive and smooth experience for your end users.

This guide provides an overview of the security capabilities that Kaltura offers and includes a high level description of the Kaltura security capabilities for video based content – including ingestion, storage and delivery. The methods that Kaltura uses to protect user information are described, in addition to Kaltura's business continuity and disaster recovery policies, Kaltura's secure platform architecture, and the physical security Kaltura maintains in its hosting facilities. If you require a more in depth description of Kaltura's security capabilities, please contact a Kaltura representative.

The options can be used as standalone capabilities – or used together to provide a full security package.

Kaltura Access Control

In many cases, organizations are interested in restricting access to content. You may want or need to employ broad controls such as allowing access only from a specific geographic location, domains, or you may want to restrict access to specific assets to certain authorized individuals only.

Kaltura offers several features that are designed to help you achieve the right level of access control to your media.

The options are summarized in the following table:

Feature	Description	When to use it?	How to apply it?

Copyright © 2019 Kaltura Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this document constitutes acceptance of the Kaltura Terms of Use and Privacy Policy.

Authentication on entry to the web page	Restricts access to the web page in which the media is hosted. Only authorized users will be able to access the web page using a password or any other secret.	In case access needs to be granted to specific people.	For Kaltura's Video Portal, MediaSpace, Kaltura offers multiple authorization options – manage users through our system or integrate with external authorization systems (LDAP, Shibboleth, CAS) as well as custom databases for single sign-on (SSO), or use a hybrid approach where authentication is done by your organization and authorization is handled by Kaltura. For more information about the MediaSpace permissions, refer to the article Kaltura Entitlement Infrastructure. For integrations with LMSs (Learning Management Systems) and CMSs (Content Management Systems), Kaltura operates within the context of the LMS or CMS. The authentication method used in the organization for access to the LMS or CMS is in effect and the media file can only be viewed by those with access permissions to the page that hosts it per the LMS/CMS configuration. If the video is embedded in the customer's website, password protection needs to be set up by the customer. on their web page.

Copyright © 2019 Kaltura Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this document constitutes acceptance of the Kaltura Terms of Use and Privacy Policy.

Geo Restriction	Restricts access to media based on the viewer's IP address geo location. This is set using the browser IP address received within the HTTP requests and the use of IP to location lookup services. For example, a Spanish client can deny access to their media to all users outside of Spain, allowing users with Spanish IPs only to access the site's media.	Geo restriction is a good way to help enforce licensing agreements, which often limit viewership to a list of approved countries.	Geo restriction can be applied using the Kaltura Management Console (KMC) on each entry or in bulk. For more information about creation of access profiles via the KMC, refer to the article How to create an access profile.
Authorized domains	Restricts access to media based on a predefined list of approved domains.	Domain restriction is useful, for example, in case you want to make sure content can only be viewed from within your domains. For example – an internal training video can only be viewed from within the enterprise domain, or a course video can only be viewed from within the university domain.	Domain restriction can be applied using the Kaltura Management Console (KMC) on each entry or in bulk. For more information about creation of access profiles via the KMC refer to the article How to create an access profile.
Authorized IP addresses	Restricts access to media based on a predefined list of approved IP addresses	In case domain authorization is not granular enough (for example if there is a large organization comprising several networks serving different divisions, and there is a desire to limit access to a specific division only).	IP address restriction can be applied using the Kaltura Management Console (KMC) on each entry or in bulk. For more information about creation of access profiles via the KMC, refer to the article How to create an access profile.

Copyright © 2019 Kaltura Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this document constitutes acceptance of the Kaltura Terms of Use and Privacy Policy.

Make your content playable only within your Kaltura account and by your player	Restrict content playback to Kaltura players from your account only	To prevent video theft, this feature also enhances brand awareness by showing your videos only on your branded player.	This setting can be turned on from within the Kaltura Management Console (KMC). If you already have content that was not secured this way, Kaltura support can assist in applying this security measure to existing entries as well.
Make your content available only in specific time windows	Configure a schedule for media entries, defining a start and end date. Playback will be allowed only within the defined schedule.	When you want to limit the time in which a media entry is available for viewing.	Scheduling can be applied using the Kaltura Management Console (KMC) on each entry or in bulk. For more information about scheduling, refer to the article More Actions Menu.
Kaltura Session Authentication for embed codes	Any published embed code of media requires a valid Kaltura Session to be passed to the embed code before the content is played. A Kaltura Session has a time expiration.	If you would like to restrict embed codes to play in authorized applications only.	Kaltura Session authentication for embed codes can be applied using the Kaltura Management Console (KMC) on each entry or in bulk. For more information about creation of access profiles via the KMC, refer to the article How to create an access profile.

Copyright © 2019 Kaltura Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this document constitutes acceptance of the Kaltura Terms of Use and Privacy Policy.

URL tokenization	URL tokenization is a content protection offered at the CDN level. The token includes a TTL (time-to-live), so that if an end user tampers with the URL, their request for CDN content is denied. If a URL has an expired TTL, end-user requests for CDN content are denied.	All the access control solutions described in this table are enforced on the Kaltura Player level. If a sophisticated user spoofs the content URL, these solutions will not be effective. URL tokenization helps prevent attempts to play the content outside of the Kaltura player. It is best to combine this feature with one of the other described access control features	URL tokenization is enabled on the CDN level, Kaltura's support team is available to set this up on your behalf.

Note: This form of protection does not encrypt the content itself, but only restricts access to the content – according to the specifications listed.

Protecting Content in Transit

Some organizations are concerned about the security of their media as it is being streamed. Content can be hijacked using man-in-the-middle attacks or other stream capture methods, and then the content can be stored locally or published illegally.

To protect your content in transit, you can use a secured streaming protocol. You can configure the CDN to stream over an encrypted protocol (HTTPS/RTPME) to prevent exposure of the content in transit. The Kaltura platform allows you to easily implement secured streaming.

AES Encryption

To support content protection on delivery, Kaltura supports AES standard encryption of content delivery for HLS delivery. Content is encrypted on the fly utilizing the Kaltura on the fly packager, and the Kaltura player can access the decryption key on the Kaltura servers to decrypt content as it is being played back.

Encryption at Rest

To support secure storage of content on the Kaltura servers, Kaltura employs encryption at rest of content. Encryption is on a per rendition level, with the encryption done as part of the transcoding process. Content is securely transitioned and stored thought the whole ingest/transcoding process.

Encryption at rest is especially beneficial for customers utilizing the Kaltura uDRM module. Since Kaltura utilizes on the fly packaging and encryption for DRM content, customers can enjoy the benefits of storing only the original content renditions – without the need of storing pre encrypted DRM flavors, and still make sure content is stored securely utilizing encryption at rest.

See Digital Rights Management for more information.

Digital Rights Management

DRM offers another layer of content protection, by adding a license policy to the content encryption.

Copyright © 2019 Kaltura Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this document constitutes acceptance of the Kaltura Terms of Use and Privacy Policy.

By adding a license, content owners can make sure that only authorized users can have access to decryption keys – and can tie their content to their business modules and protection policies.

The Kaltura uDRM module is fully integrated with Kaltura business modules definitions, making it possible for content owners to define complex business scenarios – supporting AVOD, TVOD and SVOD configurations.

Kaltura offers a full multi DRM solution – supporting all major DRM schemas including

- Microsoft PlayReady
- Google Widevine
- Apple Fairplay

By supporting all DRM schemas – content owners can ensure their content is fully DRM protected across all devices, browsers and OS, as Kaltura delivers the most natively supported and security enhanced schema on playback – utilizing the Kaltura on the fly packager. This also ensures a minimal storage foot-print, by enabling the content owners to store only the original transcoded renditions – instead of pre encrypted renditions for all DRM schemas.

DRM protection is usually required when using premium content on a monetized service and is usually a content owner/studio requirement.

The Kaltura uDRM module is integrated with the Kaltura player and the Kaltura on the fly packager, to offer a complete, easy to setup DRM eco system. In addition, since the uDRM module is API-driven – it is easily integrated with external video head ends and players is needed.

Supported Desktop Browsers for DRM

Browser	Delivery Format	DRM
IE < 11	Smooth Stream	PlayReady
IE >= 11, Edge	Dash	PlayReady
Chrome	Dash	Widevine
Safari	HLS	Fairplay
Firefox	Smooth Stream Dash	PlayReady Widevine

Mobile Device Support for DRM

Mobile Device/OS	Delivery Format	DRM
Android 4.1	WVM	Widevine Classic
Android >= 4.2	Dash	Widevine Modular
iOS	HLS	Fairplay

Copyright © 2019 Kaltura Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this document constitutes acceptance of the Kaltura Terms of Use and Privacy Policy.

Connected Devices Support for DRM

Note: * marks devices that are not supported by Kaltura player SDK and DRM plugin. Support is in the form of uDRM licensing API with integration to external players.

Device	Delivery Type	DRM
Chromecast	Dash Dash	Widevine PlayReady
XBox*	Smooth Stream	PlayReady
AppleTV*	HLS	Fairplay
GoogleTV*	WVM Smooth Stream	Widevine Classic PlayReady
FireTV*	Smooth Stream	PlayReady
SmartTV Alliance (LG, Phillips, Panasonic, Toshiba)*	Smooth Stream WVM	PlayReady Widevine Classic
Samsung TV*	Smooth Stream WVM	PlayReady Widevine Classic
HBBTV (1.5+)*	DVB Dash	PlayReady

Kaltura's API Authentication and Security

Kaltura's API is a REST-based web service accessed over HTTP. REST APIs provide a simple and easy interface for communication between applications and the Kaltura server. However, this can also be a door for weaknesses in your applications if you overlook proper security and authentication when designing your applications.

Kaltura was designed with privacy and security standards in mind, while at the same time providing openness of Kaltura’s technology as an open source platform and providing flexible integration models for open and free applications as well as highly secured and limitted applications.

The following overview describes the authntication and security model of Kaltura’s API, and how to put it to practice when implementing Kaltura applications.

Authentication and Security

To establish communication with the Kaltura servers, a client app must have a secret (one of 2 types) coupled with a unique account ID and a set of permissions.

A valid Kaltura Session (aka KS) is required to interact with the Kaltura API; displaying content, upload media, delete, update or list.

The KS expiry can be set at session initiation to range from 1 second to 10 years.

Copyright © 2019 Kaltura Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this document constitutes acceptance of the Kaltura Terms of Use and Privacy Policy.

Once the KS is acquired, it can be used to interact with content by users for specific pre-set actions, such as uploading, deletion, updating and listing.

Securing apps content is done by leveraging one or more of the following methods -

Kaltura Session version 2:

Since October 14, 2012 - Kaltura introduced a second version to the KS format that includes encryption of the fields for protecting the user privacy.

Version 1 (the original format) will continue to be maintained for backward compatibility - the Kaltura server accepts both version 1 and 2. The Kaltura server generates version 2 by default for publisher accounts created after Oct 2012.
Implementations that generate a KS locally are encouraged to use KS version 2 as well.

Since the new KS format requires encryption of the fields, performing base64 decode on the KS will not reveal its fields (as was the case with KS version 1).
To decode a KS v2, IT admins and developers who operate self hosted Kaltura servers can use the admin console developer tools page: https://[KalturaServerURL]/admin_console/index.php/plugin/KalturaInternalToolsPluginSystemHelperAction

The steps for generating a KSv2 are:

1. Gather all the different KS fields and their values:
   a. _e – expiry (unix timestamp)
   b. _u – user
   c. _t – type (KalturaSessionType)
   d. Privileges (edit, download, sview, etc.)
2. Compile all fields and URL encode the parameters as a query string. e.g.
   _u=userId&_e=12345678&_t=2&Privileges=sview:1_0xada32as;edit:*
3. Prepend 16 random binary bytes to the fields
4. Prepend the binary SHA1 hash of the string (20 string)
5. Encrypt the string with the SHA1 hash of the account's API secret using AES128/CBC/Zero bytes padding
6. Prepend the KS version and partner ID separated by pipes (e.g. v2|1234|..)
7. Encode the result using Base64
8. Replace + with – and / with _ to make the KS URL-safe

To see an implementation of the KS generation algorithm, refer to the GenerateSession function in the client library of your choice.

Methods for generating a valid Kaltura Session:

- **Generate Session Locally** - Combine all the above details, and sign them using the shared secret key. This method is great for reducing callbacks to the server and enhanced security, since the session is generated locally and the secret key is kept private.
- **Call session.start** - Calling the Kaltura Session.start API to generate a session on the server.
  Note: Using the session.start API is discouraged unless secure connection (SSL) is enabled on the account and there are specific reasons to generate the KS on the server side, using short expiry time that requires synchronizing to the server time.
- **Call user.loginByLoginId -** This method is using Kaltura Users and their Password instead of partner id and secret key.
  NOTE: This method is should be preferred in most cases.
  1. It is easier to remember user name and password.
  2. Users can be limited to specific roles and permissions (e.g. enabling only upload).
  3. Users can be deleted, password changed or demoted in permissions, while the secret keys can't be easily modified.

KS Types

User KS (Non-Authenticated User Session)

- A User KS is generated using the USER SECRET.
- USER type can only use a subset of the available services that are relevant for a user in the system.
- USER KS can invoke services on his entries and his user-data. (e.g. list actions will result in a filtered list according to the user KS)
- Attempting to manipulate other users' data will fail.

Copyright © 2019 Kaltura Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this document constitutes acceptance of the Kaltura Terms of Use and Privacy Policy.

Admin KS

- ADMIN KS is generated using the ADMIN SECRET.
- ADMIN Type is an absolute administrator and can call / perform all actions in the system. Services that use this type of session are:
  - Services that expose list of entries / users that belong to different users
  - Services that allow to update other user's data
  - Services that delete data.
- An admin KS should never reach the browser. By letting users access an admin KS they will be able to cause changes not limited to their own content.
- An admin KS ignores any privilage restirctions.

User Roles & Permissions (Authenticated User Session)

- Allow more advanced configuration of the access and permissions based on the defined Kaltura User permissions.

How May Session Type Affect API Behavior?

The session type may affect the way that some API calls behave.

Examples:

- A *media.list* call:
  - With a *user* session – lists videos owned by the user specified in the KS
  - With an *admin* session – lists all entries in the account that match your filter criteria. The list is not filtered for a specific user (unless you specifically filter by *userId*).
- An *update* call: If the user specified in a user session is not the owner of content item, the user does not have permission to update the item. You can override this restriction by specifying special session privileges.

KS Validation on the Server

The Kaltura API servers will validate the KS for:

- Check the signature against the secret of the specific publisher account to verify the authenticity of the KS.
- Check whether the KS has elapsed or the action limit has been reached.
- Check whether the KS was explicitly revoked (by issuing a Kaltura API call to expire a KS).

Once all the KS validations pass, the server will use the KS for:

- Determining the account on which an API call should be performed.
- Checking which Kaltura API services / actions the user is authorized to perform, and which API objects / properties he's allowed to view / modify. Based on the Kaltura User permissions.
- Choosing the content entities visible to the specific user.
- Setting the owning user for the API actions, e.g. any uploaded content will have the user specified in the KS as its owner.

KS Privileges

Session privileges allows applications to limit the user to perform only specific actions.

The privileges in the KS, in general, do not block actions but instead limit some actions to a smaller scope.

For example, passing "sview:{entry ID}" enables the KS to be usable for playing a specific entry.

Any attempt to use that specific KS to play another entry ID will fail, as long as the entry is protected with KS-restriction access control.

To be certain that the KS passed to player cannot be used for any update actions you can either:

- Add "setrole:PLAYBACK_BASE_ROLE" privilege to it, so it will not be allowed to perform any action other than a white-list of actions needed for the player (such as baseEntry.get, flavorAsset.list etc.).

or

- Add "widget:1" privilege to the KS to tell the server that this KS was generated for player use only, which will tell the server

Copyright © 2019 Kaltura Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this document constitutes acceptance of the Kaltura Terms of Use and Privacy Policy.

to make a distinction between a regular USER session and a "PLAYER" session.

You define privileges using a comma-separated list of key-value pairs.

Each key-value pair is a specific privilege:

- The key is the name of the privilege.
- The value is the object ID to which the privilege applies.

The key-value pair format is the key followed by the value, separated by a colon: *key:value*

Multiple key-value pairs are separated by commas with no spaces: *key:1_value,key:0_value*

Multiple parameters in a single value are separated by a slash: *key:1_value/0_value,key2:another_value*

Some privileges support a wildcard (*) value (for example, *edit:**). A wildcard permits the action for any object.

The available privileges (source reference)

Privilege	Description	Use Case	Arguments
edit	Allows editing (updating) an entry. For example, *edit:0_zsadqv3e*	Allow a specific user to edit a specific entry that does not belong to the user.	Expects entry id or * for wildcard
sview	Allows viewing and downloading an entry asset	When implementing pay-per-view with the KS Protected Access Control, allow access to the blocked video asset after purchase.	Expects entry id or * for wildcard
list	Enables the session to list for entries that are not owned by the user. By default, only admin session can list all entries, this privilege enables it for user sessions.	Performing entry search on client side, for example a contribution wizard that allows reuse of entries uploaded by other users.	Only list:* is supported (list with other parameters will be ingored)
download	Allows downloading an entry asset	Similar to *sview*. Allow actions that are meant for downloading, as opposed to streaming for playback. For example, raw action (www.kaltura.com/p/1/sp/100/raw/entryId/0_XXXYYYZZ), or download action (www.kaltura.com/p/1/sp/100/download/entryId/0_XXXYYYZZ)	Expects entry id or * for wildcard
downloadasset	enables the download of a specific asset / all assets	Used internally by the server when flavorAsset.getUrl is called.	asset id or *
editplaylist	Allows editing an entry in a specific manual playlist	Allow a user to edit a dynamic list of content for a list that is managed in a manual playlist.	Expects the id of the playlist
sviewplaylist	Allows viewing an entry in a manual playlist	Similar to *sview*. Allow a user to view a dynamic list of content.	Expects the id of the playlist

Copyright © 2019 Kaltura Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this document constitutes acceptance of the Kaltura Terms of Use and Privacy Policy.

Privilege	Description	Use Case	Arguments
edituser	Provides a USER KS the privilege to change the owner of an Entry	Allow a user to change the owner of content to another user. Allow an API-based integration to upload content on behalf of other users.	* to allow changing ownership to any user. Or specify a list of allowed usernames. Separate multiple usernames using a slash (/)
actionslimit	Allows a specific session to be used for a defined number of API calls	Allow a session with an exposed KS to be used for a restricted period. The purpose is to minimize the risk of a malicious user using the session for prohibited actions.	Expects an integer indicating number of actions
setrole	Allows a specific session to be used only for a specific role	Temporarily allow a user to perform an action that is not normally permitted, without changing the user role.	Expects the id of the role to apply on the ks
iprestrict	Limits the use of the KS to a certain IP address	Tighter security for content protection (prevent a user from being able to send the KS to other parties)	Only a single address is allowed
urirestrict	Limits the URI of the API call that the KS can call, e.g.,urirestrict:/api_v3/* will be able to call only api v3 URIs	Used internally by the server in several API calls that return a URL to the client containing a KS.	A URI (starting with /), a trailing * indicates it should be treated as a prefix
enableentitlement	Forces entitlement checks. Note: there is a setting on account level (configured in the admin console) that determines the default entitlement enforcement	Applications like MediaSpace rely on the server to perform the entitlement checks, so it uses this flag.	Doesn't have any additional attributes

Copyright © 2019 Kaltura Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this document constitutes acceptance of the Kaltura Terms of Use and Privacy Policy.

Privilege	Description	Use Case	Arguments
disableentitlement	Bypasses any entitlement checks, for example, a session with this privilege will be able to access entries in private categories that the user is not a member of Note: there is a setting on account level (configured in the admin console) that determines the default entitlement enforcement	Admin applications (e.g. KMC) that work on accounts that have entitlement enabled by default.	Doesn't have any additional attributes
disableentitlementforentry	Bypasses entitlement checks for a given entry ID. In other words, access to the given entry will be allowed even if it belongs to a private category that the user is not a member of	Sharing an entitlement protected entry.	Only a single entry id is allowed (if more are needed multiple privileges of this type can be chained)
privacycontext	Sets the privacy context for entitlement checks.	See Kaltura's Entitlement Infrastructure Information Guide.	
enablecategorymoderation	When set, new category entries that are created on categories that have moderation=true will be created in PENDING status. Otherwise, they will be created in ACTIVE status.	Supports the category moderation flow when entitlement is not enforced.	No additional attributes
reftime	A Unix timestamp that is used as the reference of relative date fields. For example, if the API gets a value of 300 for some date field, it will be translated to <reftime> + 300 (5 minutes). When this privilege is not supplied, the server uses the current time.	Tests the result of some API call in some timestamp in the future, can be used to validate the effect of scheduled tasks' filters.	A Unix timestamp
preview	A limit (in bytes) on the size of the file that is returned from the flavor download action	Used internally by the server when flavorAsset.getUrl is called on an entry whose access control has preview restrictions.	size in bytes

Copyright © 2019 Kaltura Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this document constitutes acceptance of the Kaltura Terms of Use and Privacy Policy.

Privilege	Description	Use Case	Arguments
sessionid	Can be used to group a set of KS's together for invalidation purposes - when session.end is called. With a ks that has sessionid=X, all other KS's that have sessionId=X become invalid as well.	Applications that create multiple KS's for different uses can use this privilege to terminate all KS's upon user logoff, without the need to keep track of them.	An arbitrary string identifying the session
apptoken	For a KS that was created with appToken.startSession, this privilege will contain the app token through which the KS was created.	Used mainly for investigation/tracking purposes.	The apptoken id

Examples are in PHP using the PHP5 Kaltura Client Library:

Never use KalturaSessionType::ADMIN in ks generated for end users.

Allow access to a specific entry Id (limitation is set via Access Control):
Example: allow access to entry id 0_iuasd7 (Read this blog post for use-case):

```
$ks = $client->session->start ( $userSecret, "myUser", KalturaSessionType::USER, $partnerID , null, "sview:0_iuasd7");
```

Limit number of action For KS:
Example: limit number of actions to 4:

```
$ks = $client->session->start ( $userSecret, "myUser", KalturaSessionType::USER, $partnerID , null, "actionslimit:4");
```

Set Role on the KS:
Example: set role id 2345 on a ks:

```
$ks = $client->session->start ( $userSecret, "myUser", KalturaSessionType::USER, $partnerID , null, "setrole:2345");
```

Secured Delivery

Kaltura supports various methods of securing delivery of video streams, as follows:

- Progressive download over HTTPS
- RTMPE / RTMPTE
- Akamai HD Network (chunked/throttled HTTPS)
- SWF Verification
- IP-linked token authentication

The table below shows the Stream security techniques as these apply differently across devices -

Delivery	Device	Player Security	Entitlement	Encryption
Akamai HD	Flash - PC, Android	SWF verification	IP based token	HTTPS
RTMP	Flash - PC, Android	SWF verification	IP based token	RTMPE
Progressive	All – iOS, Bberry, Flash, etc.		IP based token	HTTPS
IOS Streaming (HLS)	iPhone, iPad		IP based token	HTTPS

Kaltura’s integrated DRM solutions seamlessly plug in to its existing infrastructure and workflows, protecting customers from vendor lock-in.

Copyright © 2019 Kaltura Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this document constitutes acceptance of the Kaltura Terms of Use and Privacy Policy.

DRM Support

Encrypted video files are generated as additional “flavors” of original asset using Kaltura’s transcoding farm and based on selected vendor and license policy.

NOTE: Due to licensing requirements, DRM solutions are only available for commercial Kaltura editions (SaaS and On Prem) and are at additional cost. For more information about DRM and the available DRM solutions, please contact us or contact your Kaltura Account Manager.

Important Considerations For Application Developers

When not applications are not developer with security in mind, a malicious user can use:

- A compromised secret to create a KS at will
- A compromised admin KS to cause irreversible harm to your account (such as deleting all content)

In this section, we highlight a number of common and important practices to consider when creating applications that interact with the Kaltura API.

Authenticated User Privileges override the User Type KS

When you generate a user session KS and specify an ID of a Kaltura Admin User, the KS will allow all the actions included in the user’s role.

Always Protect your API Secret Keys

Your API Secret Keys (ADMIN and USER) are generated when you create am account. These keys hold global access permissions to your account and thus should always be kept in secret.

- Always prefer local session generation over server session.start.
- Prefer User Login over session.start when local KS generation is not possible.
- When calling the session.start API request - Make sure the connection between your client and the Kaltura server is encryoted and secured.
- NEVER keep your secret keys in a front-end application (such as Flash or JavaScript). A KS should always be generated on the server side and then passed to the front-end.
- Keep the secret keys in a seperated file with strict file permissions.

Use Admin KS with care

A compromised Admin KS will allow a malicious user to gain full access to the publisher account, leading way to harm.

Use Admin KS in between servers and with secured communication channel.

Prefer Login of Users with Defined Roles and Permissions over Generic Admin KS

Kaltura Users can be assigned a fine grained level of permissions. This allows applications developers to provide a stronger login and authentication mechanism while not exposing the account secret keys.

Use user.loginByLoginId providing user credentials and your account Id.

Use Widget KS for Anonynous Public Content Playback

The session.startWidgetSession provides an anonymous simple and light KS generation mechanism that does not require a secret. This type of session can be used to perform READ operations only and only on content that is defined as publicly available with no Access Control or special permissions.

The Widget KS is perfect for cases where public content needs to be accessed freely and without secured authentication.

[template("cat-subscribe")]

Copyright © 2019 Kaltura Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this document constitutes acceptance of the Kaltura Terms of Use and Privacy Policy.