Users, Roles, and Permissions Publisher Guide

Version: Eagle

Kaltura Business Headquarters

200 Park Avenue South, New York, NY. 10003, USA

Tel.: +1 800 871 5224

Contents

Preface

This preface contains the following topics:

- About this Guide
- Audience
- Document Conventions
- Related Documentation

About this Guide

This document describes the roles and permissions infrastructure and how to use Kaltura server APIs to define and modify roles and permissions in Kaltura applications

NOTE: Please refer to the official and latest product release notes for last-minute updates Technical support may be obtained directly from: Kaltura Support.

Contact Us:

Please send your documentation-related comments and feedback or report mistakes to the Knowledge Management Feedback group.

We are committed to improving our documentation and your feedback is important to us.

Audience

This guide is intended for Kaltura employees, partners, community members, and customers.

To understand this document, you need to be familiar with:

- Kaltura terminology
- Kaltura server API architecture, including services, actions, and objects
- PHP programming language

Document Conventions

Kaltura uses the following admonitions:

- Note
- Workflow

NOTE: Identifies important information that contains helpful suggestions.

Workflow: Provides workflow information.

1. Step 1
2. Step 2

Related Documentation

In addition to this guide, product documentation is available on the Kaltura Knowledge Center.

NOTE: Please remember to review all product release notes for known issues and limitations.

CHAPTER 1

Understanding Users, Roles, and Permissions

This chapter describes:

Roles and Permissions: Overview

Roles and permissions enable organizations to define a user's ability to perform actions based on the user's responsibilities.

A publisher uses the roles and permissions infrastructure to specify actions that a user is allowed to perform.

Server Infrastructure for Roles and Permissions

This section describes the Kaltura server infrastructure for roles and permissions.

API actions and API object properties

- An API action is not allowed unless a user has specific permission to execute it.
- All API actions have permission items.
- Only some API object properties have permission items.
- An API object parameter is allowed by default.

  When an object parameter requires a permission, a code comment specifies the permission item settings. The comment is included in the code of the class that defines the object, in the parameter's comment section. The comment format is `@requiresPermission`, followed by the applicable permission item settings. For example: `* @requiresPermission insert,update`

Terminology

permission item— Enables granular settings for accessing a specific API and object property.

permission— Defines a functional flow by grouping granular permission items.

role—Groups functional flows into usable user roles.

A permission item consists of an API action or an API object property and defines a specific API action, such as:

- baseEntry->list
- category->add
- liveStream->delete

or defines a specific API object property, such as:

- KalturaBaseEntry::startDate
- KalturaBaseEntry::accessControlProfileId

A permission item is internal to the Kaltura server. You can read, update or insert a permission item, when applied to an API object property:

A permission consists of a set of permission items and may have different meanings in different applications. A permission may be used by an application to enable access to functionality, such as creating a player or uploading content.

A *role* is a set of permissions. A user may be associated with a role.

NOTE: A user currently can have only one role. Although the server supports multiple roles per user, the API currently blocks multiple role functionality.

Default System Roles

The Kaltura system provides the following types of default roles:

- Partner (0) Roles

  Every partner may use the roles but cannot edit them.
- Template Partner (99) Roles

  The roles are copied to every partner when the partner is created. Every partner may use and edit the roles.
- Admin Console Partner (-2) Roles

  Only an Admin Console partner may use and edit the roles.

Definition of a Kaltura User

A user is an individual who logs on to a Kaltura account. A user typically accesses the Kaltura server, a Kaltura application or widget, or a Kaltura plugin. Roles and permissions apply to users of the Kaltura system.

Server Infrastructure for Users

This section describes the Kaltura server infrastructure for users.

Partner

A partner is an individual or organization with a Kaltura system account. A partner defines that roles and permissions that apply to its users. A user may be associated with multiple partner accounts.

Account Owner

Each Kaltura account must have a user who is defined as the account owner.

An account owner:

- Cannot be deleted
- Receives account administrator emails
- Has full control of account permissions and roles
- May assign full control of account permissions and roles to additional users

- May be changed to a different user after the user is assigned full control of account permissions and roles

NOTE: An account may have only one account owner.

Kaltura Session (KS)

The Kaltura system uses a Kaltura Session (KS) identifier to identify and authenticate a user. The KS is the string identifier generated by Kaltura or the client application using a shared secret for web session authentication The KS is generated using elements such as:

- Partner ID
- User ID
- Session type

The session type may be one of the following:

- ADMIN – Can access all the entries of the partner
- USER – Can access only entries created by the user

Source for Roles and Permissions

The user's role determines permissions when the KS contains a user ID and a role is assigned to the user.

The KS session type determines permissions when one of the following occurs:

- The KS does not contain a user ID.
- The user is not assigned a role.

Partner User ID (puser_id)

A puser ID uniquely identifies the user for a specific partner.

Kaltura User (kuser)

A kuser:

- Is an object that represents a person who uses the Kaltura system
- Contains metadata about the user, such as name, email, and location
- Is identified by a partner user ID (puser_id), which must be unique among one partner's users
- May be associated with more than one partner.

NOTE: When a kuser is associated with multiple partners, the user has multiple kuser objects, one for each partner. All of the multiple kuser objects use the same `user_login_data` record for system logins.

User Login Data

Information about Kaltura user (kuser) logons is stored in the `user_login_data` database table. The user login data includes: Email, Name, Password, Last partner whose account the user logged on to, Number of bad login retries, and the Password expiry date.

One data record may relate to multiple kusers, allowing a user to use a single ID and password for multiple partner accounts.

Permission Types

The Kaltura system provides the following types of permission:

- Normal Permissions
- Special and Plugin Permissions
- Partner Group Permissions

A permission type may be one of the following:

- A user-level permission, which is associated with an individual user through a user role
- A partner-level permission, which applies only to a partner and is not associated with an individual user. A user role does not contain partner-level permissions.

Normal Permissions

Normal permissions, described in this document, are user-level permissions. Normal permissions group permission items, which may be included in a user role. The user role is assigned to an individual user.

A user role may only contain normal permissions.

Special and Plugin Permissions

Special and *plugin* permissions are partner-level permissions. Special and plugin permissions may define a feature, such as access to a specific plugin (for example, virus scan) or a special service, such as analytics.

NOTE: A normal permission that specifies access to an API action or object property that relates to a plugin usually applies only if the plugin is included in a partner-level special permission.

Partner Group Permissions

Partner group permissions are partner-level permissions and include permission items. Partner group permissions have a special `partnerGroup` parameter that is a comma-separated list of partner IDs.

NOTE: When the value of `partnerGroup` is an asterisk (*), the parameter applies to all partners.

When a user is allowed to perform an action based on the user's role and the user's partner is included in a partner group permission, the user is allowed to perform the action on all the partners listed in `partnerGroup`.

CHAPTER 2

APIs for Users, Roles, and Permissions

This chapter describes the Kaltura APIs that you can use to define and modify users, roles, and permissions.

UserService

A service API class that manages partner users.

Remarks

`userId` is the unique identifier in the partner's system.

The `partnerId,userId` couple constitutes a unique key in Kaltura's database.

Extends

`KalturaBaseUserService`

NOTE: `AdminUserService` also extends `KalturaBaseUserService` (for backward compatibility). Do not use `AdminUserService`, which is deprecated.

Actions

Name	Description
addAction	Adds a new user to an existing account in the Kaltura database.
updateAction	Updates an existing user object.
getAction	Retrieves a user object for a specified user ID.
getByLoginIdAction	Retrieves a user object for a user's login ID and partner ID.
deleteAction	Deletes a user from a partner account.
listAction	Lists user objects that are associated with an account.
notifyBan	Notifies that a user is banned from an account.
loginAction	Logs a user into a partner account with a partner ID, a partner user ID (puser), and a user password.
loginByLoginIdAction	Logs a user into a partner account with a user login ID and a user password.
updateLoginDataAction	Updates a user's login data: • Email • Password • Name
resetPasswordAction	Resets a user's password and sends the user an email containing

Name	Description
	a link for setting up the new password.
setInitialPasswordAction	Sets a user's password.
enableLoginAction	Enables a user to log into a partner account using an email address and a password.
disableLoginAction	Disables a user’s ability to log into a partner account using an email address and a password.

addAction

Adds a new user to an existing account in the Kaltura database.

```
function addAction(KalturaUser $user)
```

Parameters

Name	Input/Output	Type	Description
`user`	Input	`KalturaUser`	The new user

Return Value

Type	Description
`KalturaUser`	The added user object

Errors

Name	Description
DUPLICATE_USER_BY_ID	A user with the ID [user->id] already exists in system.
PROPERTY_VALIDATION_CANNOT_BE_NULL	The property [property_name] cannot be NULL.
INVALID_FIELD_VALUE	The value in field [field_name] is not valid.
UNKNOWN_PARTNER_ID	The partner ID [partner_id] is not recognized.
ADMIN_LOGIN_USERS_QUOTA_EXCEEDED	The permitted number of user logins has been reached. No more logins are permitted for this partner account.
PASSWORD_STRUCTURE_INVALID	The password you entered is not valid. Passwords must: • Contain between 8 and 14 characters • Not contain your name • Contain at least one lowercase letter (a-z) • Contain at least one digit (0-9) • Contain at least one of the following symbols: %~!@#$^*=+?[]{} • Not contain the following characters: < or >
DUPLICATE_USER_BY_LOGIN_ID	A loginable user with the email [login_email] already exists in system.

updateAction

Updates an existing user object.

```
public function updateAction($userId, KalturaUser $user)
```

Remarks

You also can use this action to update the `userId`.

Parameters

Name	Input/Output	Type	Description
`userId`	Input	String	The user's unique identifier in the partner's system
`user`	Input	`KalturaUser`	The updated user

Return Value

Type	Description
`KalturaUser`	The updated user object

Errors

Name	Description
INVALID_USER_ID	The user ID is not valid.
CANNOT_DELETE_OR_BLOCK_ROOT_ADMIN_USER	An account owner user cannot be deleted.
USER_ROLE_NOT_FOUND	The user role cannot be located.
ACCOUNT_OWNER_NEEDS_PARTNER_ADMIN_ROLE	The account owner must have a partner administrator role.

getAction

Retrieves a user object for a specified user ID.

```
public function getAction($userId)
```

Parameters

Name	Input/Output	Type	Description
`userId`	Input	String	The user's unique identifier in the partner's system

Return Value

Type	Description
`KalturaUser`	The specified user object

Errors

Name	Description
INVALID_USER_ID	The user ID is not valid.

getByLoginIdAction

Retrieves a user object for a user's login ID and partner ID.

```
public function getByLoginIdAction($loginId)
```

Remarks

A login ID is the email address used by a user to log into the system.

Parameters

Name	Input/Output	Type	Description
`loginId`	Input	String	The user's email address that identifies the user for login

Return Value

Type	Description
`KalturaUser`	The user object represented by the login and partner IDs

Errors

Name	Description
LOGIN_DATA_NOT_FOUND	The login ID cannot be located.
USER_NOT_FOUND	The user cannot be located.

deleteAction

Deletes a user from a partner account.

```
public function deleteAction($userId)
```

Parameters

Name	Input/Output	Type	Description
`userId`	Input	String	The user's unique identifier in the partner's system

Return Value

Type	Description
`KalturaUser`	The deleted user object

Errors

Name	Description
INVALID_USER_ID	The user ID is not valid.

listAction

Lists user objects that are associated with an account.

```
public function listAction(KalturaUserFilter $filter = null, KalturaFilterPager
   $pager = null)
```

Remarks

Blocked users are listed unless you use a filter to exclude them.

Deleted users are not listed unless you use a filter to include them.

Parameters

Name	Input/Output	Type	Description
`filter`	Input (optional)	`KalturaUserFilter`	A filter used to exclude specific types of users
`pager`	Input (optional)	`KalturaFilterPager`	A limit for the number of records to display on a page

Return Value

Type	Description
`KalturaUserListResponse`	The list of user objects

notifyBan

Notifies that a user is banned from an account.

```
public function notifyBan($userId)
```

Parameters

Name	Input/Output	Type	Description
`userId`	Input	String	The user's unique identifier in the partner's system

Errors

Name	Description
INVALID_USER_ID	The user ID is not valid.

loginAction

Logs a user into a partner account with a partner ID, a partner user ID (puser), and a user password.

```
public function loginAction($partnerId, $userId, $password, $expiry = 86400,
   $privileges = '*')
```

Remarks

The action enables a user to generate a KS. The KS enables the user to execute API actions.

Parameters

Name	Input/Output	Type	Description
`partnerId`	Input	Integer	The identifier of the partner account
`userId`	Input	String	The user's unique identifier in the partner's system
`password`	Input	String	The user's password
`expiry`	Input	Integer	The requested time (in seconds) before the generated KS expires. By default, a KS expires after 24

Name	Input/Output	Type	Description
			hours.
`privileges`	Input	String	Special privileges

Return Value

Type	Description
String	The user's KS

Errors

Name	Description
USER_NOT_FOUND	The user cannot be located.
USER_WRONG_PASSWORD	The user password is incorrect.
INVALID_PARTNER_ID	The partner ID [partner_id] is not valid.
LOGIN_RETRIES_EXCEEDED	You tried logging in too many times. Your account is locked and will not be available for 24 hours.
LOGIN_BLOCKED	Your account is locked.
PASSWORD_EXPIRED	Your password expired.
USER_IS_BLOCKED	The user is blocked.

loginByLoginIdAction

Logs a user into a partner account with a user login ID and a user password.

```
public function loginByLoginIdAction($loginId, $password, $partnerId = null,
   $expiry = 86400, $privileges = '*')
```

Remarks

The action enables a user to generate a KS. The KS enables the user to execute API actions.

Parameters

Name	Input/Output	Type	Description
`loginId`	Input	String	The user's email address that identifies the user for login
`password`	Input	String	The user's password
`partnerId`	Input (Optional)	Integer	The identifier of the partner account
`expiry`	Input	Integer	The requested time (in seconds) before the generated KS expires. By default, a KS expires after 24 hours.
`privileges`	Input	String	Special privileges

Return Value

Type	Description
String	The user's KS

Errors

Name	Description
USER_NOT_FOUND	The user cannot be located.
USER_WRONG_PASSWORD	The user password is incorrect.
INVALID_PARTNER_ID	The partner ID [name] is not valid.
LOGIN_RETRIES_EXCEEDED	You tried logging in too many times. Your account is locked and will not be available for 24 hours.
LOGIN_BLOCKED	Your account is locked.
PASSWORD_EXPIRED	Your password expired.
USER_IS_BLOCKED	The user is blocked.

updateLoginDataAction

Updates a user's login data:

- Email
- Password
- Name

```
public function updateLoginDataAction($oldLoginId, $password, $newLoginId = "",
  $newPassword = "", $newFirstName = null, $newLastName = null)
```

Parameters

Name	Input/Output	Type	Description
`oldLoginId`	Input	String	The user's current email address that identified the user for login
`password`	Input	String	The user's existing password
`newLoginId`	Input (Optional)	String	The user's email address that will identify the user for login
`newPassword`	Input (Optional)	String	The user's new password
`newFirstName`	Input (Optional)	String	The user's new first name
`newLastName`	Input (Optional)	String	The user's new last name

Errors

Name	Description
INVALID_FIELD_VALUE	The value in field [name] is not valid.
LOGIN_DATA_NOT_FOUND	The login ID cannot be located.
WRONG_OLD_PASSWORD	The existing password is incorrect.
PASSWORD_STRUCTURE_INVALID	The password you entered is not valid. Passwords must: • Contain between 8 and 14 characters • Not contain your name

Name	Description
	• Contain at least one lowercase letter (a-z) • Contain at least one digit (0-9) • Contain at least one of the following symbols: %~!@#$^*=+?[]{} • Not contain the following characters: < or >
PASSWORD_ALREADY_USED	The password you chose has already been used.
LOGIN_ID_ALREADY_USED	The login ID is already in use.

resetPasswordAction

Resets a user's password and sends the user an email containing a link for setting up the new password.

```
public function resetPasswordAction($email)
```

Remarks

This action is used in the "Forgot password" feature.

Parameters

Name	Input/Output	Type	Description
`email`	Input	String	The user's email address (login email)

Errors

Name	Description
LOGIN_DATA_NOT_FOUND	The login ID (login email) cannot be located.
PASSWORD_STRUCTURE_INVALID	The password you entered is not valid. Passwords must: • Contain between 8 and 14 characters • Not contain your name • Contain at least one lowercase letter (a-z) • Contain at least one digit (0-9) • Contain at least one of the following symbols: %~!@#$^*=+?[]{} • Not contain the following characters: < or >
PASSWORD_ALREADY_USED	The password you chose has already been used.
INVALID_FIELD_VALUE	The value in field [field_name] is not valid.
LOGIN_ID_ALREADY_USED	The login ID is already in use.

setInitialPasswordAction

Sets a user's password.

```
public function setInitialPasswordAction($hashKey, $newPassword)
```

Remarks

The application uses this action internally.

Context

Called to set a user's password after a `resetPasswordAction` request.

Parameters

Name	Input/Output	Type	Description
`hashKey`	Input	String	The hash key used to identify the user
`newPassword`	Input	String	The new password to set for the user

Errors

Name	Description
LOGIN_DATA_NOT_FOUND	The login ID cannot be located.
PASSWORD_STRUCTURE_INVALID	The password you entered is not valid. Passwords must: • Contain between 8 and 14 characters • Not contain your name • Contain at least one lowercase letter (a-z) • Contain at least one digit (0-9) • Contain at least one of the following symbols: %~!@#$^*=+?[]{} • Not contain the following characters: < or >
NEW_PASSWORD_HASH_KEY_EXPIRED	The specified hash key is expired.
NEW_PASSWORD_HASH_KEY_INVALID	The specified hash key is not valid.
PASSWORD_ALREADY_USED	The password you chose has already been used.
INTERNAL_SERVERL_ERROR	An internal server error occurred.

enableLoginAction

Enables a user to log into a partner account using an email address and a password.

```
public function enableLoginAction($userId, $loginId, $password = null)
```

Parameters

Name	Input/Output	Type	Description
`userId`	Input	String	The user's unique identifier in the partner's system
`loginId`	Input	String	The user's email address that identifies the user for login
`password`	Input (Optional)	String	The user's password

Return Value

Type	Description

Type	Description
`KalturaUser`	The user object represented by the user and login IDs

Errors

Name	Description
USER_LOGIN_ALREADY_ENABLED	The user already is allowed to login.
USER_NOT_FOUND	The user cannot be located.
ADMIN_LOGIN_USERS_QUOTA_EXCEEDED	The permitted number of user logins has been reached. No more logins are permitted for this partner account.
PASSWORD_STRUCTURE_INVALID	The password you entered is not valid. Passwords must: • Contain between 8 and 14 characters • Not contain your name • Contain at least one lowercase letter (a-z) • Contain at least one digit (0-9) • Contain at least one of the following symbols: %~!@#$^*=+?[]{} • Not contain the following characters: < or >
LOGIN_ID_ALREADY_USED	The login ID is already in use.

disableLoginAction

Disables a user’s ability to log into a partner account using an email address and a password.

```
public function disableLoginAction($userId = null, $loginId = null)
```

Remarks

You may use either a `userId` or a `loginId` parameter for this action.

Parameters

Name	Input/Output	Type	Description
`userId`	Input (Optional)	String	The user's unique identifier in the partner's system
`loginId`	Input (Optional)	String	The user's email address that identifies the user for login

Return Value

Type	Description
`KalturaUser`	The user object represented by the user and login IDs

Errors

Name	Description
USER_LOGIN_ALREADY_DISABLED	The user already is not allowed to log in.
PROPERTY_VALIDATION_CANNOT_BE_NULL	The property [property_name] cannot be NULL.

Name	Description
USER_NOT_FOUND	The user cannot be located.
CANNOT_DISABLE_LOGIN_FOR_ADMIN_USER	A login cannot be disabled for an ADMIN user.

UserRoleService

A service API class that creates and manages user roles.

Extends

`KalturaBaseService`

Actions

Name	Description
addAction	Adds a new user role object to the account.
getAction	Retrieves a user role object using its ID.
updateAction	Updates an existing user role object.
deleteAction	Deletes an existing user role object.
listAction	Lists user role objects that are associated with an account.
cloneAction	Creates a new user role object that is a duplicate of an existing role.

addAction

Adds a new user role object to the account.

```
public function addAction(KalturaUserRole $userRole)
```

Parameters

Name	Input/Output	Type	Description
`userRole`	Input	`KalturaUserRole`	A new role

Return Value

Type	Description
`KalturaUserRole`	The added user role object

Errors

Name	Description
PROPERTY_VALIDATION_CANNOT_BE_NULL	The property [name] cannot be NULL.
PROPERTY_VALIDATION_NOT_UPDATABLE	The property [name] cannot be updated.
PERMISSION_NOT_FOUND	A permission associated with the role cannot be located.

getAction

Retrieves a user role object using its ID.

```
public function getAction($userRoleId)
```

Parameters

Name	Input/Output	Type	Description
`userRoleId`	Input	Integer	The user role's unique identifier

Return Value

Type	Description
`KalturaUserRole`	The retrieved user role object

Errors

Name	Description
INVALID_OBJECT_ID	The object identifier is not valid.

updateAction

Updates an existing user role object.

```
public function updateAction($userRoleId, KalturaUserRole $userRole)
```

Parameters

Name	Input/Output	Type	Description
`userRoleId`	Input	Integer	The user role's unique identifier
`userRole`	Input	`KalturaUserRole`	The role object that contains parameters to update

Return Value

Type	Description
`KalturaUserRole`	The updated user role object

Errors

Name	Description
INVALID_OBJECT_ID	The object identifier is not valid.
PERMISSION_NOT_FOUND	A permission associated with the role cannot be located.

deleteAction

Deletes an existing user role object.

```
public function deleteAction($userRoleId)
```

Parameters

Name	Input/Output	Type	Description
`userRoleId`	Input	Integer	The user role's unique identifier

Return Value

Type	Description
`KalturaUserRole`	The deleted user role object

Errors

Name	Description
INVALID_OBJECT_ID	The object identifier is not valid.
ROLE_IS_BEING_USED	The role is in use. The action cannot be completed.

listAction

Lists user role objects that are associated with an account.

```
public function listAction(KalturaUserRoleFilter $filter = null,
   KalturaFilterPager $pager = null)
```

Remarks

Blocked user roles are listed unless you use a filter to exclude them. Deleted user roles are not listed unless you use a filter to include them.

Parameters

Name	Input/Output	Type	Description
`filter`	Input (Optional)	`KalturaUserRoleFilter`	A filter used to exclude specific types of user roles
`pager`	Input (Optional)	`KalturaFilterPager`	A limit for the number of records to display on a page

Return Value

Type	Description
`KalturaUserRoleListResponse`	The list of user role objects

cloneAction

Creates a new user role object that is a duplicate of an existing role.

```
public function cloneAction($userRoleId)
```

Parameters

Name	Input/Output	Type	Description
`userRoleId`	Input	Integer	The user role's unique identifier

Return Value

Type	Description

Type	Description
`KalturaUserRole`	The duplicate user role object

Errors

Name	Description
INVALID_OBJECT_ID	The object identifier is not valid.

PermissionService

A service API class that creates and manages user permissions.

Extends

`KalturaBaseService`

Actions

Name	Description
addAction	Adds a new permission object to the account.
getAction	Retrieves a permission object using its ID.
updateAction	Updates an existing permission object.
deleteAction	Deletes an existing permission object.
listAction	Lists permission objects that are associated with an account.
getCurrentPermissions	Retrieves a list of permissions that apply to the current KS.

addAction

Adds a new permission object to the account.

```
public function addAction(KalturaPermission $permission)
```

Parameters

Name	Input/Output	Type	Description
`permission`	Input	`KalturaPermission`	The new permission

Return Value

Type	Description
`KalturaPermission`	The added permission object

Errors

Name	Description
PROPERTY_VALIDATION_CANNOT_BE_NULL	The property [name] cannot be NULL.
PROPERTY_VALIDATION_NOT_UPDATABLE	The property [name] cannot be updated.

getAction

Retrieves a permission object using its ID.

```
public function getAction($permissionName)
```

Parameters

Name	Input/Output	Type	Description
`permissionName`	Input	String	The name assigned to the permission

Return Value

Type	Description
`KalturaPermission`	The retrieved permission object

Errors

Name	Description
INVALID_OBJECT_ID	The object identifier is not valid.

updateAction

Updates an existing permission object.

```
public function updateAction($permissionName, KalturaPermission $permission)
```

Parameters

Name	Input/Output	Type	Description
`permissionName`	Input	String	The name assigned to the permission
`permission`	Input	`KalturaPermission`	The updated permission

Return Value

Type	Description
`KalturaPermission`	The updated permission object

Errors

Name	Description
INVALID_OBJECT_ID	The object identifier is not valid.

deleteAction

Deletes an existing permission object.

```
public function deleteAction($permissionName)
```

Parameters

Name	Input/Output	Type	Description

Name	Input/Output	Type	Description
`permissionName`	Input	String	The name assigned to the permission

Return Value

Type	Description
`KalturaPermission`	The deleted permission object

Errors

Name	Description
INVALID_OBJECT_ID	The object identifier is not valid.

listAction

Lists permission objects that are associated with an account.

```
public function listAction(KalturaPermissionFilter $filter = null,
  KalturaFilterPager $pager = null)
```

Remarks

Blocked permissions are listed unless you use a filter to exclude them. Deleted permissions are not listed unless you use a filter to include them.

Parameters

Name	Input/Output	Type	Description
`filter`	Input (Optional)	`KalturaPermissionFilter`	A filter used to exclude specific types of permissions
`pager`	Input (Optional)	`KalturaFilterPager`	A limit for the number of records to display on a page

Return Value

Type	Description
`KalturaPermissionListResponse`	The list of permission objects

getCurrentPermissions

Retrieves a list of permissions that apply to the current KS.

```
public function getCurrentPermissions()
```

Remarks

The data is extracted from the KS as follows:

- The data is taken from the user’s role when there is a `userId`.
- The data is taken according to the `sessionType` in the following cases:
  - There is no `userID`.
  - The user is not associated with a role.

Return Value

Type	Description
String	A comma-separated list of current permission names

PermissionItemService

A service API class that creates and manages permission items.

Extends

`KalturaBaseService`

Actions

Name	Description
addAction	Adds a new permission item object to the account.
getAction	Retrieves a permission item object using its ID.
updateAction	Lists permission item objects that are associated with an account.
deleteAction	Deletes an existing permission item object.
listAction	Lists permission item objects that are associated with an account.

addAction

Adds a new permission item object to the account.

```
public function addAction(KalturaPermissionItem $permissionItem)
```

Remarks

This action is available only to Kaltura system administrators.

Parameters

Name	Input/Output	Type	Description
`permissionItem`	Input	`KalturaPermissionItem`	The new permission item

Return Value

Type	Description
`KalturaPermissionItem`	The added permission item object

Errors

Name	Description
PROPERTY_VALIDATION_CANNOT_BE_NULL	The property [name] cannot be NULL.
PROPERTY_VALIDATION_NOT_UPDATABLE	The property [name] cannot be updated.

getAction

Retrieves a permission item object using its ID.

```
public function getAction($permissionItemId)
```

Parameters

Name	Input/Output	Type	Description
`permissionItemId`	Input	Integer	The permission item's unique identifier

Return Value

Type	Description
`KalturaPermissionItem`	The retrieved permission item object

Errors

Name	Description
INVALID_OBJECT_ID	The object identifier is not valid.

updateAction

Updates an existing permission item object.

```
public function updateAction($permissionItemId, KalturaPermissionItem
   $permissionItem)
```

Remarks

This action is available only to Kaltura system administrators.

Parameters

Name	Input/Output	Type	Description
`permissionItemId`	Input	Integer	The permission item's unique identifier
`permissionItem`	Input	`KalturaPermissionItem`	The updated permission item

Return Value

Type	Description
`KalturaPermissionItem`	The updated permission item object

Errors

Name	Description
INVALID_OBJECT_ID	The object identifier is not valid.

deleteAction

Deletes an existing permission item object.

```
public function deleteAction($permissionItemId)
```

Remarks

This action is available only to Kaltura system administrators.

Parameters

Name	Input/Output	Type	Description
`permissionItemId`	Input	Integer	The permission item's unique identifier

Return Value

Type	Description
`KalturaPermissionItem`	The deleted permission item object

Errors

Name	Description
INVALID_OBJECT_ID	The object identifier is not valid.

listAction

Lists permission item objects that are associated with an account.

```
public function listAction(KalturaPermissionItemFilter $filter = null,
   KalturaFilterPager $pager = null)
```

Parameters

Name	Input/Output	Type	Description
`filter`	Input (Optional)	`KalturaPermissionItemFilter`	A filter used to exclude specific types of permission items
`pager`	Input (Optional)	`KalturaFilterPager`	A limit for the number of records to display on a page

Return Value

Type	Description
`KalturaPremissionItemListResponse`	The list of permission item objects

CHAPTER 3

Use Cases

This chapter describes use cases for:

- Adding a User
- Creating a New Role Based on an Existing Role
- Modifying a Permission
- Listing an Account's Permission Items
- Listing Permission Items for a Specific User Role
- Applying a Role to a User

Adding a User

To add a user to a partner account:

1. Log on to the Kaltura system.
   A KS is generated to identify and authenticate the user.
2. Send the following API request to the server: `UserService->addAction`
   Specify the user parameters as a `KalturaUser` object.
   The server processes the API request and returns the new user object.

Sample Code

```
/** Include the KalturaClient class */
require_once('KalturaClient.php');
```

```
/** Initiate a new KalturaClient object with the partner details */
$partnerId = <PARTNER_ID_HERE>; // replace with your partner ID
$config = new KalturaConfiguration($partnerId);
$config->serviceUrl = 'http://www.kaltura.com/'; // set serviceUrl if different
$client = new KalturaClient($config);

/** Set user login variables and call user->loginByLoginId action */
$loginId = '<USER_LOGIN_EMAIL_HERE>';     // replace with user's login email
$password = '<USER_LOGIN_PASSWORD_HERE>'; // replace with user's login password
$ks = $client->user->loginByLoginId($loginId, $password, $partnerId);

/** Set the client to use the KS that is returned */
$client->setKs($ks);

/** Create a new KalturaUser object to be added */
$newUser = new KalturaUser();
$newUser->id = 'NEW_USER_ID';
$newUser->firstName = 'first name';
$newUser->lastName = 'last name';
$newUser->email = 'new_user@kaltura.com';
$newUser->isAdmin = true;
$newUser->roleIds = 30; // must match an existing user role ID

/** Call the user->add action with the new user object */
$addedUser = $client->user->add($newUser);
```

Creating a New Role Based on an Existing Role

To create a new role from an existing role:

1. Log on to the Kaltura system.
   A KS is generated to identify and authenticate the user.
2. Send the following API request to the server: UserRoleService->cloneAction
   Specify the user role to copy.
   The server processes the API request and returns the duplicate user role object.

Sample Code

```
/** Include the KalturaClient class */
require_once('KalturaClient.php');

/** Initiate a new KalturaClient object with the partner details */
$partnerId = <PARTNER_ID_HERE>; // replace with your partner ID
$config = new KalturaConfiguration($partnerId);
$config->serviceUrl = 'http://www.kaltura.com/'; // set serviceUrl if different
$client = new KalturaClient($config);

/** Set user login variables and call user->loginByLoginId action */
$loginId = '<USER_LOGIN_EMAIL_HERE>';     // replace with user's login email
$password = '<USER_LOGIN_PASSWORD_HERE>'; // replace with user's login password
$ks = $client->user->loginByLoginId($loginId, $password, $partnerId);

/** Set the client to use the KS that is returned */
$client->setKs($ks);

/** Set the original user role ID*/
$originalRoleId = <ORIGINAL_USER_ROLE_ID>; // must match an existing user role ID

/** Call the userRole->clone action */
$duplicatedRole = $client->userRole->cloneAction($originalRoleId);
```

Modifying a Permission

To modify a permission:

1. Log on to the Kaltura system.
   A KS is generated to identify and authenticate the user.
2. Send the following API request to the server: PermissionService->updateAction
   Specify the following:
   - The current permission name
   - The properties to update as a `KalturaPermission` object

   The server processes the API request and returns the updated permission object.

Sample Code

```
/** Include the KalturaClient class */
require_once('KalturaClient.php');

/** Initiate a new KalturaClient object with the partner details */
$partnerId = <PARTNER_ID_HERE>; // replace with your partner ID
$config = new KalturaConfiguration($partnerId);
$config->serviceUrl = 'http://www.kaltura.com/'; // set serviceUrl if different
$client = new KalturaClient($config);

/** Set user login variables and call user->loginByLoginId action */
$loginId = '<USER_LOGIN_EMAIL_HERE>';     // replace with user's login email
$password = '<USER_LOGIN_PASSWORD_HERE>'; // replace with user's login password
$ks = $client->user->loginByLoginId($loginId, $password, $partnerId);

/** Set the client to use the KS that is returned */
$client->setKs($ks);

/** Set the existing permission’s name */
$permissionName = '<PERMISSION_NAME_HERE>'; // must match an existing permission name

/** Set a KalturaPermission object with the required parameters to update */
$updatePermission = new KalturaPermission();
$updatePermission->name = 'NEW_PERMISSION_NAME';
$updatePermission->permissionItemsIds = '<LIST_OF_PERMISSION_ITEM_IDS>'; // must match existing permission item IDs

/** Call the permission->update action */
$updatedPermission = $client->permission->update($permissionName, $updatePermission);
```

Listing an Account's Permission Items

To list the permission items that are associated with an account:

1. Log on to the Kaltura system.
   A KS is generated to identify and authenticate the user.
2. Send the following API request to the server: PermissionItemService->listAction
   You may specify:

- o A filter
- o A limit on the number of records to display on a page

The server processes the API request and returns the list of permission item objects.

Sample Code

```
/** Include the KalturaClient class */
require_once('KalturaClient.php');

/** Initiate a new KalturaClient object with the partner details */
$partnerId = <PARTNER_ID_HERE>; // replace with your partner ID
$config = new KalturaConfiguration($partnerId);
$config->serviceUrl = 'http://www.kaltura.com/'; // set serviceUrl if different
$client = new KalturaClient($config);

/** Set user login variables and call user->loginByLoginId action */
$loginId = '<USER_LOGIN_EMAIL_HERE>';     // replace with user's login email
$password = '<USER_LOGIN_PASSWORD_HERE>'; // replace with user's login password
$ks = $client->user->loginByLoginId($loginId, $password, $partnerId);

/** Set the client to use the KS that is returned */
$client->setKs($ks);

/** Set a KalturaPermissionItemFilter object - optional */
$filter = new KalturaPermissionItemFilter();
$filter->typeIn = KalturaPermissionItemType::API_ACTION_ITEM; // example of listing only action permission items

/** Set a KalturaFilterPager object - optional */
$pager = new KalturaFilterPager();
$pager->pageIndex = 1; // get only page 1
$pager->pageSize = 20; // page size must be 20

/** Call the permissionItem->list action */
$permissionItemList = $client->permissionItem->listAction($filter, $pager);
```

Listing Permission Items for a Specific User Role

To list the permission items that are associated with a specific user role:

1. Log on to the Kaltura system.
   A KS is generated to identify and authenticate the user.
2. Send the following API request to the server: UserRoleService->getAction
   You must specify the user role ID.
   The server processes the API request and returns the requested user role object.
3. From the returned user role object, copy the value of the `permissionNames` parameter. The `permissionNames` parameter is a comma-separated list of all permissions associated with a role.
4. Send the following API request to the server: PermissionService->listAction
   Specify a filter and set the `nameIn` filter parameter value as the list of permission names copied from the user role service.
   The server processes the API request and returns the list of permission objects with given names.
5. From each permission object returned in the list, copy the `permissionItemsIds` parameter value. The `permissionItemsIds` parameter is a comma-separated list of permission item IDs associated with a permission.
6. Send the following API request to the server: PermissionItemService->listAction
   Specify a filter and set the `idIn` value as a comma-separated list of permission item IDs copied from the permission item service (a concatenated list of all permission item IDs from all permission objects).
   The server processes the API request and returns the list of permission item objects.

Sample Code

```
/** Include the KalturaClient class */
require_once('KalturaClient.php');

/** Initiate a new KalturaClient object with the partner details */
$partnerId = <PARTNER_ID_HERE>; // replace with your partner ID
$config = new KalturaConfiguration($partnerId);
$config->serviceUrl = 'http://www.kaltura.com/'; // set serviceUrl if different
$client = new KalturaClient($config);

/** Set user login variables and call user->loginByLoginId action */
$loginId = '<USER_LOGIN_EMAIL_HERE>';      // replace with user's login email
$password = '<USER_LOGIN_PASSWORD_HERE>'; // replace with user's login password
```

```
$ks = $client->user->loginByLoginId($loginId, $password, $partnerId);

/** Set the client to use the KS that is returned */
$client->setKs($ks);

/** Call the userRole->get action with the desired user role id **/
$userRoleId = <USER_ROLE_ID_HERE>;
$userRole = $client->userRole->get($userRoleId);

/** Set a KalturaPermissionFilter object  */
$permissionFilter = new KalturaPermissionFilter();
$permissionFilter->nameIn = $userRole->permissionNames; // list only permission with the given names from the user role object

/** Call the permission->list action with the created filter **/
$permissionList = $client->permission->listAction($permissionFilter);

$permissionItemIds = '';
foreach ($permissionList->objects as $permission)
{
   if ($permission->permissionItemsIds)
   {
       $permissionItemIds .= ','.$permission->permissionItemsIds;
   }
}
$permissionItemIds = trim($permissionItemIds, ',');

/** Set a KalturaPermissionItemFilter object  */
$permissionItemFilter = new KalturaPermissionItemFilter();
$permissionItemFilter->idIn = $permissionItemIds; // list only permission items with the given ids from all permission objects

/** Call the permissionItem->list action */
$permissionItemList = $client->permissionItem->listAction($permissionItemFilter);
```

Applying a Role to a User

To apply a role to a user:

1. Log on to the Kaltura system.
   A KS is generated to identify and authenticate the user.
2. Send the following API request to the server: UserService->updateAction

Specify the following:

- The user role ID
- A `KalturaUser` object with a parameter that specifies the new user role identifier

The server processes the API request and returns the user role object.

Sample Code

```
/** Include the KalturaClient class */
require_once('KalturaClient.php');

/** Initiate a new KalturaClient object with the partner details */
$partnerId = <PARTNER_ID_HERE>; // replace with your partner ID
$config = new KalturaConfiguration($partnerId);
$config->serviceUrl = 'http://www.kaltura.com/'; // set serviceUrl if different
$client = new KalturaClient($config);

/** Set user login variables and call user->loginByLoginId action */
$loginId = '<USER_LOGIN_EMAIL_HERE>';     // replace with user's login email
$password = '<USER_LOGIN_PASSWORD_HERE>'; // replace with user's login password
$ks = $client->user->loginByLoginId($loginId, $password, $partnerId);

/** Set the client to use the KS that is returned */
$client->setKs($ks);

/** Set the existing user’s ID */
$userId = 'user_id'; // must match an existing user ID

/** Create a new KalturaUser object and set the roleIds parameter to the required user role ID */
$updateUser = new KalturaUser();
$updateUser->roleIds = '<NEW_ROLE_ID_HERE>'; // must match an existing user role ID

/** Call the user->update action */
$updatedUser = $client->user->update($userId, $updateUser);
```

CHAPTER 4

Data Flows

This chapter describes server and application data flows for users, roles, and permissions.

Understanding Server Data Flows

This section describes the basic data flow for users, roles, and permissions in the Kaltura server.

Workflow:

1. A user logs into the system using a unique email and password or the account's user/admin secret.
2. A successful login request generates a KS string that is returned to the user. The KS holds elements that identify the user, such as `partnerId`, `userId`, and `sessionType`.
3. The user uses the KS to issue an API request.
4. The server:
   a. Receives the API request for a specific service and action.
   b. Decodes the KS, identifies the user, and checks for the role associated with the user.
   c. Checks for the permissions contained in the user's role.
   d. Checks for the permission items contained in the user's permissions.
   e. Grants access to the requested service and action only if a suitable permission item is found for the user.
5. A request to insert or update a new object property that requires special permission succeeds only if the user has the insert or update permission for the property. The request fails if the user does not have the permission.
6. The response includes an object property that requires special permission only if the user has the permission item that allows reading the property.

Understanding Application Data Flows

This section describes data flows for users, roles, and permissions in the KMC and the Admin Console.

KMC Data Flow

This section describes the basic data flow for users, roles, and permissions in the KMC.

Workflow:

1. A user logs into the KMC.
2. Using an API request, the KMC requests a list of permissions associated with the user from the server.
3. The KMC displays and hides specific UI objects according to the user's permissions.

Admin Console Data Flow

This section describes the basic data flow for users, roles, and permissions in the Admin Console.

Workflow:

1. A user logs into the Admin Console.
2. Using an API request, the Admin Console requests a list of permissions associated with the user from the server.
3. According to the user's permissions, the Admin Console:
   - Displays and hides specific UI objects
   - Enables and blocks Admin Console actions

GLOSSARY

Term	Definition
Kaltura Administration Console	An application for administering the Kaltura system, including administration of multiple Kaltura accounts. The Admin Console typically is accessed by Kaltura system administrators and the IT team.
KMC	Kaltura Management Console. An application for content management, application creation and configuration, content monetization, distribution and syndication, and account management and reporting. The KMC is accessed by Kaltura partner administrators and the various users of a Kaltura account.
KS	Kaltura session. A unique string that identifies the session creator. The KS is used to authenticate Kaltura API calls.
kuser	Kaltura user
Partner	An individual or organization with a Kaltura system account
Partner ID	A numeric identifier that uniquely identifies a partner in the Kaltura database
Publisher	See Partner.
puser ID	Partner user identifier