Creating a Custom Distribution Destination Plugin
Developer Guide

Version: Eagle

Kaltura
open source video

Kaltura Business Headquarters

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

Tel.: +1 800 871 5224

Contents

Preface 6
About this Guide 6
Audience 6
Document Conventions 6
Related Documentation 7
Chapter 1 Understanding Content Distribution 8
Media Entries and Distribution Destinations 8
Implementation 8
Key Features 8
Chapter 2 Understanding Kaltura Server Plugin Creation 9
Chapter 3 Understanding Key Data Flows 10
Submit 11
Update 12
Delete 13
Chapter 4 Implementing Objects 14
IDistributionProvider 14
getType 15
getName 15
isDeleteEnabled 15
isUpdateEnabled 16
isReportsEnabled 16
isScheduleUpdateEnabled 16
useDeleteInsteadOfUpdate 17
getJobIntervalBeforeSunrise 17
getJobIntervalBeforeSunset 18
getUpdateRequiredEntryFields 18
getUpdateRequiredMetadataXPaths 19
IDistributionEngine 19
IDistributionEngineSubmit 20
IDistributionEngineCloseSubmit 20
IDistributionEngineUpdate 21
IDistributionEngineCloseUpdate 21
IDistributionEngineDelete 22
IDistributionEngineCloseDelete 23
IDistributionEngineReport 23
IDistributionEngineCloseReport 24
DistributionProfile 24
partner_id 24
provider_type 25
Form_ProviderProfileConfiguration 25

__construct 25
saveProviderAdditionalObjects 26
addProviderElements 26
resetUnUpdatebleAttributes 26
addProfileAction 27
KalturaDistributionJobProviderData 27
__construct 28
kDistributionJobProviderData 28
__construct 28
KalturaDistributionProfile 29
getMapBetweenObjects 30
toObject 30
fromObject 31
Chapter 5 Implementing Kaltura Plugins 32
IKalturaContentDistributionProvider 32
getProvider 32
getKalturaProvider 33
contributeMRSS 33
IKalturaEnumerator 34
getEnums 34
IKalturaEventConsumers 34
getEventConsumers 34
IKalturaObjectLoader 35
loadObject 35
getObjectClass 36
IKalturaPending 37
dependsOn 37
IKalturaPermissions 37
isAllowedPartner 37
Chapter 6 Database Schema 39
Distribution Profile 39
Entry Distribution 42
Chapter 7 Sample Code 47
ExampleDistributionPlugin.php 47
lib 52
ExampleDistributionEngine.php 52
ExampleDistributionProfile.php 58
ExampleDistributionProvider.php 59
ExampleDistributionProviderType.php 62
kExampleDistributionJobProviderData.php 62
api 64
ExampleAPI 69
xml 70
update.template.xml 70

Glossary 71

Preface

This preface contains the following topics:

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

About this Guide

This document explains how to use the Kaltura infrastructure to create a new Kaltura server plugin for one or more content distribution destinations.

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, customers, and media providers that want to integrate with Kaltura.

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

- Kaltura terminology
- Kaltura server plugin architecture
- Media concepts
- 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 Content Distribution

Kaltura's content distribution feature enables a partner to send content to multiple destinations from a centralized location.

Each external destination is a media provider.

Only media entries for video and audio media types, excluding images, are currently supported.

Media Entries and Distribution Destinations

For each media entry, you can specify multiple distribution destinations and define entry scheduling.

You can schedule a media entry differently for each distribution destination.

For each distribution destination, you can define media entry packages. A media package represents a Kaltura entry in the destination.

A media package may include, for example:

- Multiple video or audio files
- Multiple thumbnail images
- Metadata

Implementation

You implement the content distribution feature with a Kaltura server plugin.

This document explains how to create a Kaltura server plugin using the Kaltura infrastructure.

Key Features

Key features of a Kaltura server plugin for distribution destinations include:

- Submit a media package to a destination.
- Update media and metadata of a package that has been submitted to a distribution destination.
- Delete a media package from a distribution destination.
- Define automated sunrise and sunset times when the distribution destination does not support specifying these times.

CHAPTER 2

Understanding Kaltura Server Plugin Creation

To create a Kaltura server plugin for distribution destinations, you:

- Implement interfaces:
    - IDistributionProvider
    - IDistributionEngine
- Extend objects:
    - DistributionProfile
    - Form_ProviderProfileConfiguration
    - KalturaDistributionJobProviderData
    - kDistributionJobProviderData
    - KalturaDistributionProfile
- Implement Kaltura Plugins (see Implementing Kaltura Plugins)
    - IKalturaContentDistributionProvider
    - IKalturaEnumerator
    - IKalturaEventConsumers
    - IKalturaObjectLoader
    - IKalturaPending
    - IKalturaPermissions

Database Schema describes the database tables that store configuration data.

Sample Code provides sample code.

CHAPTER 3

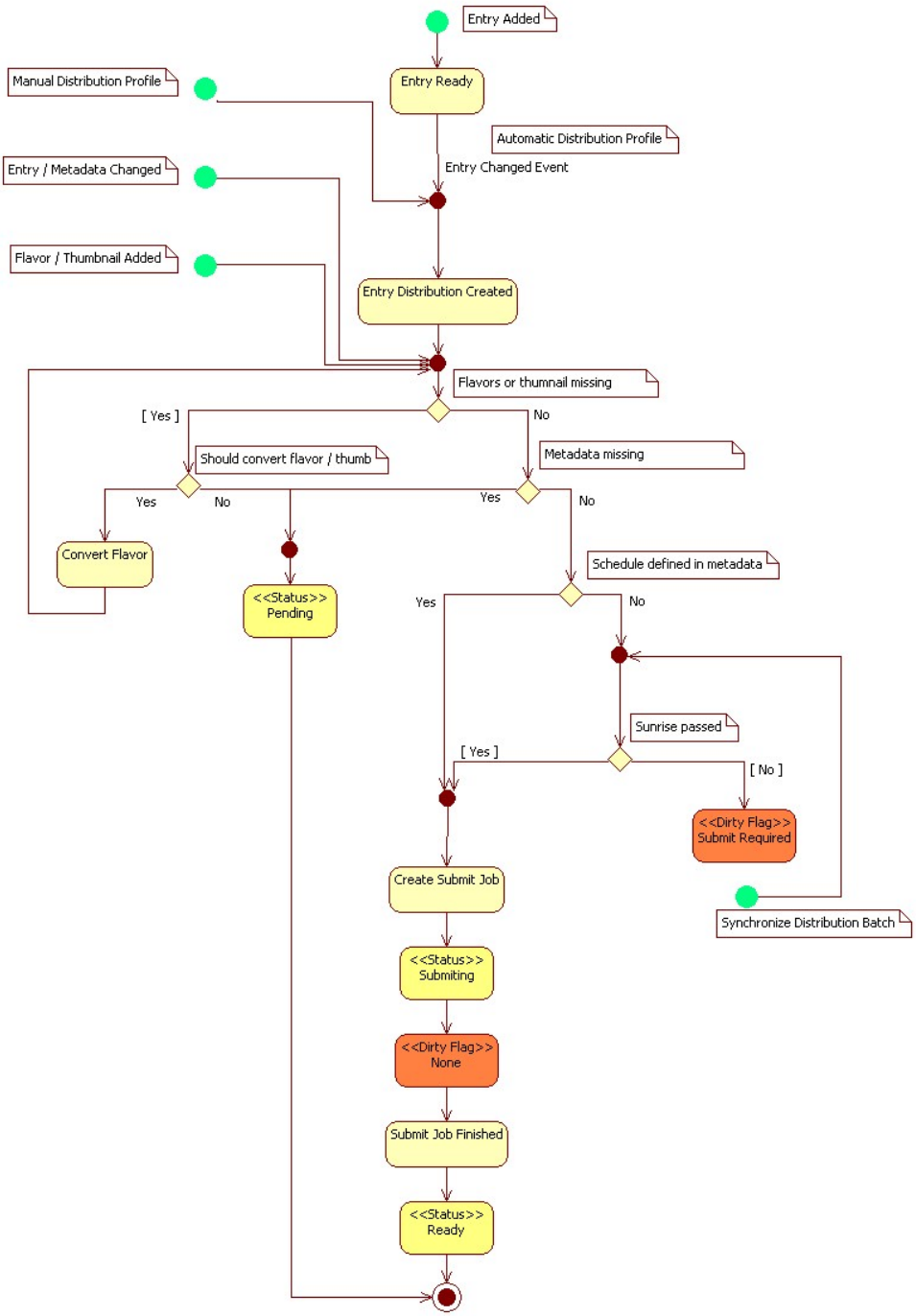
Understanding Key Data Flows

The key data flows of `IDistributionEngine,` which implements the distribution protocol, are:

- Submit
- Update
- Delete

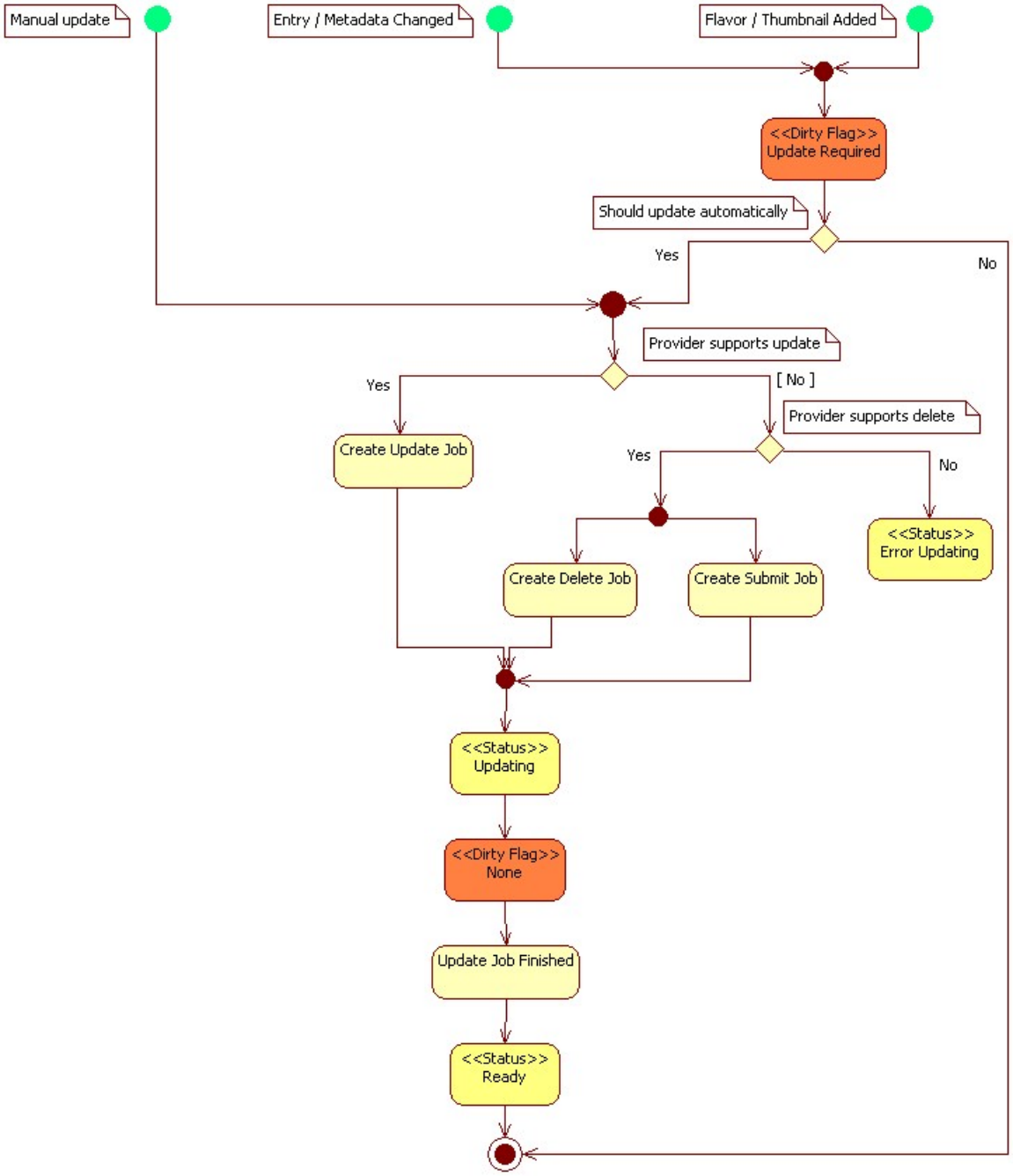
Submit

The submit action sends a media package to a destination.



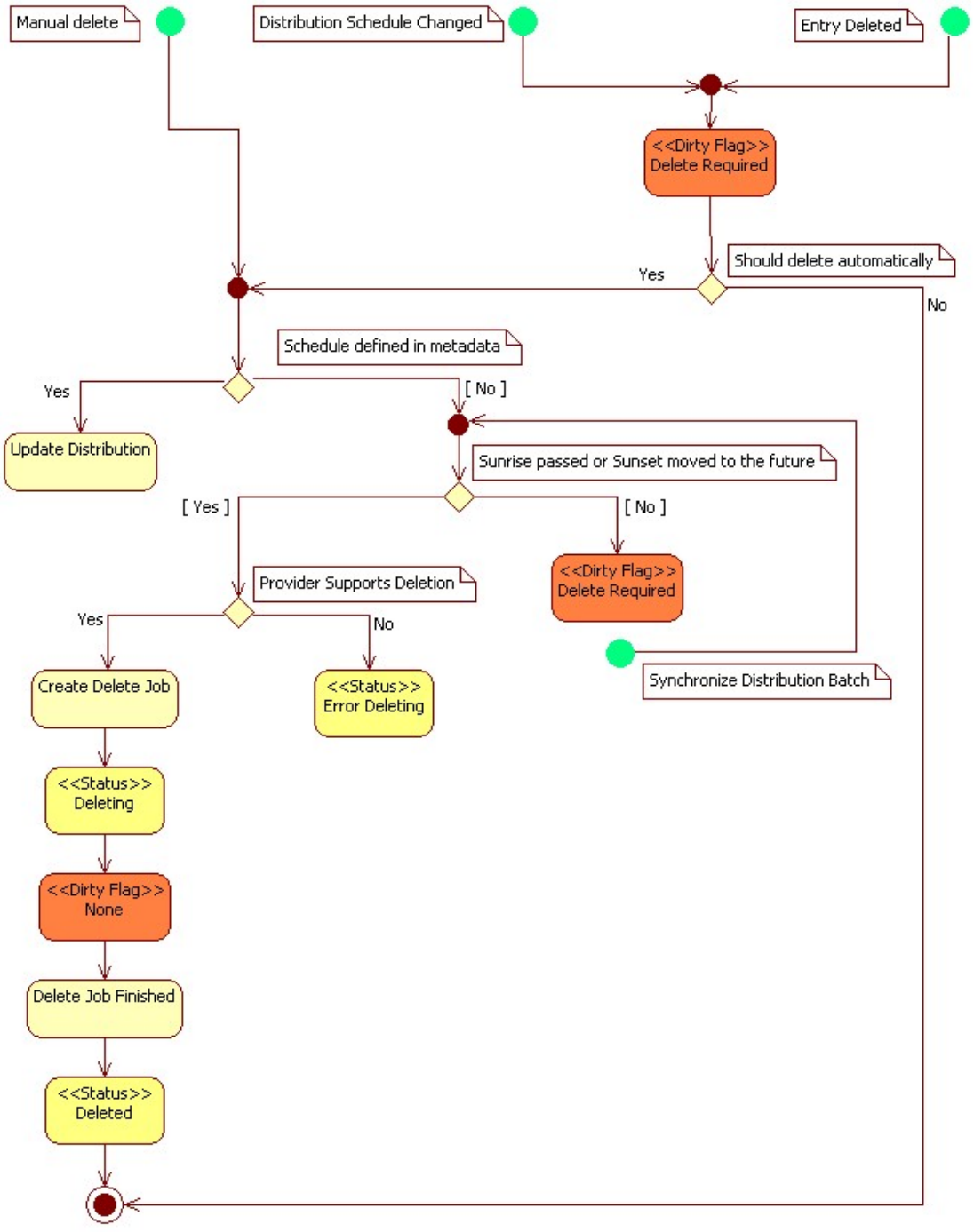
Update

The update action updates media and metadata of a package that has been submitted to a distribution destination.



Delete

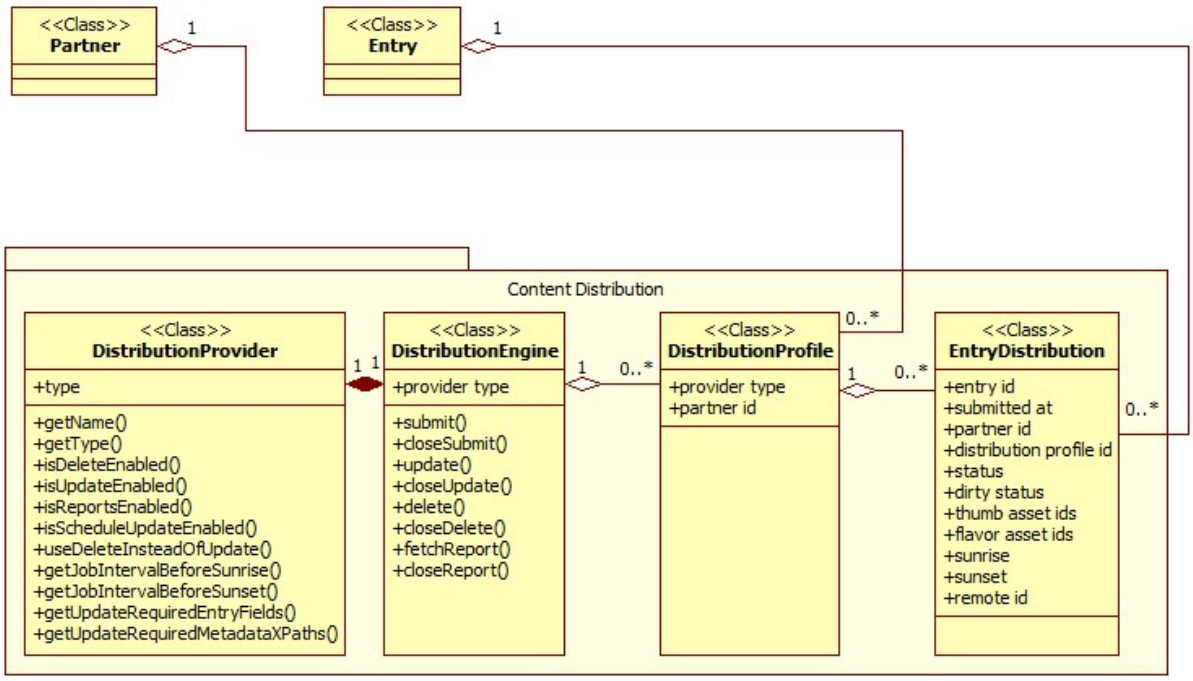
The delete action deletes a media package from a distribution destination.



CHAPTER 4

Implementing Objects

Distribution Class Diagram



NOTE: `DistributionProvider` defines all of the methods that are available for a distribution provider to implement. `DistributionProvider` is used by the Kaltura system to determine the actions that are supported by the distribution destination.

`DistributionEngine` implements the distribution protocol. `DistributionEngine` is required for each distribution provider.

`DistributionProfile` represents a distribution provider partner-specific configuration, for example, account and credentials. `DistributionProfile` is required for each distribution provider.

`EntryDistribution` provides metadata, status, and errors about the distributed media entry.

IDistributionProvider

`IDistributionProvider` is an interface for flow utility methods that are:

- Common for all distribution providers
- Required for the flow management of content distribution

To implement the objects:

You need to know the return value for each method of the interface.

getType

Retrieves the value that the plugin creator adds to the `DistributionProviderType` enumerator.

```
public function getType();
```

Context

- Called by the content distribution MRSS contributor to identify each provider according to its type.
- Translated to the `KalturaDistributionProvider` object and exposed through the API.

Return Value

Type	Description
`DistributionProviderType` (Integer)	The type of distribution provider

getName

Retrieves the friendly name of the distribution provider, used in logs, MRSS, and management tools.

```
public function getName();
```

Context

- Called by the content distribution MRSS contributor to identify each provider according to its type.
- Translated to the `KalturaDistributionProvider` object and exposed through the API.

Return Value

Type	Description
String	The friendly name of the distribution provider

isDeleteEnabled

Indicates whether the distribution provider supports content deletion.

```
public function isDeleteEnabled();
```

Context

- Called by the content distribution flow manager when delete is required or requested.
- Translated to the `KalturaDistributionProvider` object and exposed through the API.

Return Value

Type	Description
Boolean	True — The provider supports media deletion. False — The provider does not support media deletion.

isUpdateEnabled

Indicates whether the distribution provider supports modification of media and metadata that are included in the distribution package.

```
public function isUpdateEnabled();
```

Context

- Called by the content distribution flow manager when update is required or requested.
- Translated to the `KalturaDistributionProvider` object and exposed through the API.

Return Value

Type	Description
Boolean	True — The provider supports modification of media and metadata. False — The provider does not support modification of media and metadata.

isReportsEnabled

Indicates whether the distribution provider supports retrieval of reports on the distributed content.

```
public function isReportsEnabled();
```

Context

- Called by the content distribution flow manager when fetching reports is required or requested.
- Translated to the `KalturaDistributionProvider` object and exposed through the API.

Return Value

Type	Description
Boolean	True — The provider supports report retrieval. False — The provider does not support report retrieval.

isScheduleUpdateEnabled

Indicates whether the distribution package metadata may include sunrise and sunset attributes.

```
public function isScheduleUpdateEnabled();
```

Remarks

If schedule update is not enabled, the distribution provider uses the dirty flags to delay submission of an entry until sunrise and to delete an entry after sunset.

Context

- Called by the content distribution flow manager to decide whether the entry can be sent before sunrise or only upon sunrise.
- Called by the content distribution flow manager to decide whether:
    - The entry becomes unavailable automatically.
    - A delete job is created to remove the entry upon sunset.

- Translated to the `KalturaDistributionProvider` object and exposed through the API.

Return Value

Type	Description
Boolean	True — The distribution package may include sunrise and sunset attributes. False — The distribution package may not include sunrise and sunset attributes.

useDeleteInsteadOfUpdate

Indicates that the distribution provider can use the `delete` and `submit` methods instead of the `update` method.

```
public function useDeleteInsteadOfUpdate();
```

Remarks

The distribution provider uses `useDeleteInsteadOfUpdate` when the distribution provider does not support the `update` method.

Context

- Called by the content distribution flow manager to decide whether deleting the entry and re-submitting it can replace the update operation. This call occurs only if update is disabled.
- Translated to the `KalturaDistributionProvider` object and exposed through the API.

Return Value

Type	Description
Boolean	True — The distribution provider can use `delete` and `submit`. False — The distribution provider cannot use `delete` and `submit`.

getJobIntervalBeforeSunrise

Retrieves the interval (in seconds) between the time that the package must be submitted and the requested sunrise time.

```
public function getJobIntervalBeforeSunrise();
```

Remarks

Use this object only if the sunrise data cannot be sent in the package data.

Context

- Called by the content distribution flow manager to decide how long before real sunrise the submit job is created.
- Used only when `isScheduleUpdateEnabled` returns false.
- Translated to the `KalturaDistributionProvider` object and exposed through the API.

Return Value

Type	Description

Integer	The number of seconds required for the interval between creating a job and sunrise. The permissible values are zero or a positive integer.

getJobIntervalBeforeSunset

Retrieves the interval (in seconds) between the time that the deletion job must be created and the requested sunset time.

```
public function getJobIntervalBeforeSunset();
```

Remarks

Use this object only if the sunset data cannot be sent in the package data.

Context

- Called by the content distribution flow manager to decide how long before real sunset the delete job is created.
- Used only when `isScheduleUpdateEnabled` returns false.
- Translated to the `KalturaDistributionProvider` object and exposed through the API.

Return Value

Type	Description
Integer	The number of seconds required for the interval between creating a job and sunset. The permissible values are zero or a positive integer.

getUpdateRequiredEntryFields

Retrieves an array of entry attributes, where changing the value of any of the attributes triggers an update job or marks the package as requiring an update.

```
public function getUpdateRequiredEntryFields($distributionProfileId = null);
```

Remarks

The array may include one or more of the following:

- Entry Columns, for example, entry.NAME and entry.DESCRIPTION
- Customized properties, for example, width and height

Context

Called by the content distribution flow manager to decide whether the changed fields in the entry object require updating the remote destination site.

Parameters

Name	Input/Output	Type	Description
`distributionProfileId`	Input (optional)	Integer	Distribution profile identifier (see DistributionProfile)

Return Value

Type	Description

Array	The array of entry attributes.

getUpdateRequiredMetadataXPaths

Retrieves an array of XPaths, where changing the value of any of the XPaths triggers an update job or marks the package as requiring an update.

```
public function getUpdateRequiredMetadataXPaths($distributionProfileId = null);
```

Remarks

An XPath is used to navigate to a metadata XML node.

Context

Called by the content distribution flow manager to decide whether the changed nodes in the metadata XML require updating the remote destination site.

Parameters

Name	Input/Output	Type	Description
`distributionProfileId`	Input (optional)	Integer	Distribution profile identifier (see (see DistributionProfile)

Return Value

Type	Description
Array of strings	The array of XPaths

IDistributionEngine

`IDistributionEngine` is an interface that implements the distribution protocol.

The interface is executed by the batch job to perform the following actions:

- submit
- closeSubmit
- update
- closeUpdate
- delete
- closeDelete
- fetchReport
- closeReport

You can implement a subset of the actions. Actions that you enable on the provider must be implemented in your engine.

To implement the objects:

You need to know the protocol used, for example, HTML hit, XML, or API libraries.

IDistributionEngineSubmit

Extends

`IDistributionEngine`

submit

Sends a media package to a destination.

Implements the provider protocol and data structure mapping.

```
public function submit(KalturaDistributionSubmitJobData $data);
```

Context

Called by the `KAsyncDistributeSubmit` batch worker.

Parameters

Name	Input/Output	Type	Description
`data`	Input	`KalturaDistributionSubmitJobData`	The submitted data

Return Value

Type	Description
Boolean	True — The action completed successfully. False — The action will complete asynchronously.

NOTE: All exceptions mark the job as failed except KalturaDistributionException, which marks the job for retry.

IDistributionEngineCloseSubmit

Extends

`IDistributionEngine`

closeSubmit

Checks whether an asynchronous submission is complete.

```
public function closeSubmit(KalturaDistributionSubmitJobData $data);
```

Context

Called by the `KAsyncDistributeSubmitCloser` batch worker.

Parameters

Name	Input/Output	Type	Description
`data`	Input	`KalturaDistributionSubmitJobData`	The submitted data

Return Value

Type	Description
Boolean	True — The action completed successfully. False — The action is not yet complete.

NOTE: All exceptions mark the job as failed except KalturaDistributionException, which marks the job for retry.

IDistributionEngineUpdate

Extends

`IDistributionEngine`

update

Updates media and metadata of a package that has been submitted to a distribution destination.

Implements the provider protocol and data structure mapping.

```
public function update(KalturaDistributionUpdateJobData $data);
```

Context

Called by the `KAsyncDistributeUpdate` batch worker.

Parameters

Name	Input/Output	Type	Description
`data`	Input	`KalturaDistributionUpdateJobData`	The updated data

Return Value

Type	Description
Boolean	True — The action completed successfully. False — The action will complete asynchronously.

NOTE: All exceptions mark the job as failed except KalturaDistributionException, which marks the job for retry.

IDistributionEngineCloseUpdate

Extends

`IDistributionEngine`

closeUpdate

Checks whether an asynchronous update is complete.

```
public function closeUpdate(KalturaDistributionUpdateJobData $data);
```

Context

Called by the `KAsyncDistributeUpdateCloser` batch worker.

Parameters

Name	Input/Output	Type	Description
`data`	Input	`KalturaDistributionUpdateJobData`	The updated data

Return Value

Type	Description
Boolean	True — The action completed successfully. False — The action is not yet complete.

NOTE: All exceptions mark the job as failed except KalturaDistributionException, which marks the job for retry.

IDistributionEngineDelete

Extends

`IDistributionEngine`

delete

Deletes a media package from a distribution destination.

Implements the provider protocol and data structure mapping.

```
public function delete(KalturaDistributionDeleteJobData $data);
```

Context

Called by the `KAsyncDistributeDelete` batch worker.

Parameters

Name	Input/Output	Type	Description
`data`	Input	`KalturaDistributionDeleteJobData`	The deleted data

Return Value

Type	Description
Boolean	True — The action completed successfully. False — The action will complete asynchronously.

NOTE: All exceptions mark the job as failed except KalturaDistributionException, which marks the job for retry.

IDistributionEngineCloseDelete

Extends

`IDistributionEngine`

closeDelete

Checks whether an asynchronous deletion is complete.

```
public function closeDelete(KalturaDistributionDeleteJobData $data);
```

Context

Called by the `KAsyncDistributeDeleteCloser` batch worker.

Parameters

Name	Input/Output	Type	Description
`Data`	Input	`KalturaDistributionUpdateJobData`	The deleted data

Return Value

Type	Description
Boolean	True — The action completed successfully. False — The action is not yet complete.

NOTE: All exceptions mark the job as failed except KalturaDistributionException, which marks the job for retry.

IDistributionEngineReport

Extends

`IDistributionEngine`

fetchReport

Retrieves statistics regarding a distribution package.

```
public function fetchReport(KalturaDistributionFetchReportJobData $data);
```

Context

Called by the `KAsyncDistributeFetchReport` batch worker.

Parameters

Name	Input/Output	Type	Description
`data`	Input	`KalturaDistributionFetchReportJobData`	The report data

Return Value

Type	Description

Boolean	True — The action completed successfully. False — The action will complete asynchronously.

NOTE: All exceptions mark the job as failed except KalturaDistributionException, which marks the job for retry.

IDistributionEngineCloseReport

Extends

`IDistributionEngine`

closeReport

Checks whether an asynchronous retrieval of a report is complete.

```
public function closeReport(KalturaDistributionFetchReportJobData $data);
```

Context

Called by the `KAsyncDistributeFetchReportCloser` batch worker.

Parameters

Name	Input/Output	Type	Description
`data`	Input	`KalturaDistributionFetchReportJobData`	The report data

Return Value

Type	Description
Boolean	True — The action completed successfully. False — The action is not yet complete.

NOTE: All exceptions mark the job as failed except KalturaDistributionException, which marks the job for retry.

DistributionProfile

`DistributionProfile` is an object that resides in the Kaltura database.

The object stores data that is required for distribution, such as username and password.

Define `DistributionProfile` once for each distribution provider and partner.

To extend the object:

You need to know the required configuration fields and attributes.

partner_id

Retrieves the value of the `[partner_id]` column in the distribution_profile table.

```
public function getPartnerId()
```

Return Value

Type	Description
`partner_id` (Integer)	The value of the Partner ID column

provider_type

Retrieves the value of the `[provider_type]` column in the distribution_profile table.

The value is the numeric identifier from the dynamic_enum table, which maps plugin extended enumerators to a numeric ID.

```
public function getProviderType()
```

Return Value

Type	Description
`provider_type` (Integer)	The value of the Provider Type column

Form_ProviderProfileConfiguration

`Form_ProviderProfileConfiguration` is a database form object that contains the configuration of the distribution profile.

Extends

`Form_DistributionConfiguration`

To extend the object:

Implement this class to enable configuring the distribution profile from the Admin Console. This class enables you to expand the configuration form and to add your own customized fields.

__construct

Retrieves the partner ID and provider type to use them later to generate and populate the form.

```
public function __construct($partnerId, $providerType)
```

Remarks

The plugin must implement this object to add custom fields to the Admin Console configuration form.

Implementing this method is not mandatory.

Context

Called by the Admin Console.

Parameters

Name	Input/Output	Type	Description
`partnerId`	Input	Integer	The identifier of the partner
`providerType`	Input	Integer	The type of provider

saveProviderAdditionalObjects

Enables you to use the Kaltura API to save objects in addition to the distribution profile.

```
public function saveProviderAdditionalObjects(KalturaDistributionProfile
   $distributionProfile)
```

Remarks

This object enables you to configure a customized object that relates to the configured distribution profile.

Implementing this method is not mandatory.

Context

Called by the Admin Console after the distribution profile element is saved.

Parameters

Name	Input/Output	Type	Description
`distributionProfile`	Input	`KalturaDistributionProfile`	The saved `DistributionProfile` object

addProviderElements

Adds custom elements to the distribution profile configuration form.

```
abstract protected function addProviderElements();
```

Remarks

Implementing this method is mandatory.

Context

Called by the parent class.

resetUnUpdatebleAttributes

Enables you to reset distribution profile read-only attributes.

```
public function resetUnUpdatebleAttributes(KalturaDistributionProfile
   $distributionProfile)
```

Remarks

Always call the parent class method to reset the base read-only attributes:

- `id`
- `partnerId`
- `createdAt`
- `updatedAt`
- `providerType`

For more information on the attributes, see `KalturaDistributionProfile`.

Implementing this method is not mandatory.

Context

Called by the Admin Console before saving the distribution profile.

Parameters

Name	Input/Output	Type	Description
`distributionProfile`	Input	`KalturaDistributionProfile`	The `DistributionProfile` object to be saved

addProfileAction

Creates a configuration sub-form for each distribution action: submit, update, delete, and fetchReport.

```
protected function addProfileAction($action)
```

Remarks

Overwrite this method to add form elements to the action sub-forms.

Always call the parent class method.

Implementing this method is not mandatory.

Context

Called by the parent class.

Parameters

Name	Input/Output	Type	Description
`action`	Input	String	The custom distribution profile action

Return Value

Type	Description
`Zend_Form_DisplayGroup`	The sub-form element display group

KalturaDistributionJobProviderData

`KalturaDistributionJobProviderData` is an API object that specifies distribution provider data that is handled by the distribution job.

To pass additional required data from the server to the batch engine, add your own attributes to this extended object.

Extends

`KalturaObject`

To extend the object:

The plugin must not implement any of the methods.

__construct

Specifies distribution provider data that is handled by the distribution job.

```
public function __construct(KalturaDistributionJobData $distributionJobData =
   null)
```

Remarks

Using the supplied data object, you can populate additional attributes according to your needs.

Context

Called by `KalturaDistributionJobData`.

Parameters

Name	Input/Output	Type	Description
`distributionJobData`	Input (optional)	`KalturaDistributionJobData`	The job data object that will contain this object in the `providerData` attribute

kDistributionJobProviderData

`kDistributionJobProviderData` is a database object that represents distribution provider data that is handled by the distribution job. Its purpose is to store provider-specific additional data on the job data object.

To extend the object:

The plugin must not implement any of the methods.

__construct

Instantiate the object with default values.

```
public function __construct(kDistributionJobData $distributionJobData = null)
```

Remarks

Attributes may be populated based on the supplied `kDistributionJobProviderData` object.

Context

Called by `KalturaDistributionJobData`.

Parameters

Name	Input/Output	Type	Description
`distributionJobData`	Input (optional)	`kDistributionJobData`	The job data object that will contain this object in the `providerData` attribute

KalturaDistributionProfile

`KalturaDistributionProfile` is an API object that represents a distribution provider partner-specific configuration of the following variables:

Variable	Type	Description
`id`	Integer	Automatically-generated unique ID
`createdAt`	Integer	Date that profile was created. Unix timestamp in seconds.
`updatedAt`	Integer	Date that profile was last updated. Unix timestamp in seconds.
`partnerId`	Integer	A numeric identifier that uniquely identifies the partner in the Kaltura database
`providerType`	`KalturaDistributionProviderType`	Pluginable enum that indicates the type of provider.
`name`	String	The name of the distribution profile
`status`	`KalturaDistributionProfileStatus`	The status of the distribution profile. Possible values: • Enabled • Disabled • Deleted
`submitEnabled`	`KalturaDistributionProfileActionStatus`	The distribution engine supports the submit method. Possible values: • Manual • Automatic • Disabled
`updateEnabled`	`KalturaDistributionProfileActionStatus`	The distribution engine supports the update method. Possible values: • Manual • Automatic • Disabled
`deleteEnabled`	`KalturaDistributionProfileActionStatus`	The distribution engine supports the delete method. Possible values: • Manual • Automatic • Disabled
`reportEnabled`	`KalturaDistributionProfileActionStatus`	The distribution engine supports the fetchReport method. Possible values: • Manual • Automatic • Disabled
`autoCreateFlavors`	String	Comma-separated flavor parameter IDs

Variable	Type	Description
		that will be converted automatically
autoCreateThumb	String	Comma-separated thumbnail parameter IDs that will be generated automatically
optionalFlavorParamsIds	String	Comma-separated flavor parameter IDs that will be submitted if ready
requiredFlavorParamsIds	String	Comma-separated flavor parameter IDs that must be ready before submission
optionalThumbDimensions	KalturaDistributionThumbDimensionsArray	Thumbnail dimensions that will be submitted if ready
requiredThumbDimensions	KalturaDistributionThumbDimensionsArray	Thumbnail dimensions that must be ready before submission
sunriseDefaultOffset	Integer	Default sunrise if sunrise is not specified in the entry distribution. Defined in seconds following entry creation.
sunsetDefaultOffset	Integer	Default sunset if sunset is not specified in the entry distribution. Defined in seconds following entry creation.

Extends

`KalturaObject`

To extend the object:

The plugin must implement the methods to add additional API attributes to the base class.

getMapBetweenObjects

Retrieves a map of object fields to their respective setters or getters. The purpose is to translate between the API and the core objects.

```
public function getMapBetweenObjects()
```

Remarks

Implementing this method is required only if you want to add attributes that will be automatically translated between the API and the core objects.

Context

Called by the `toObject` and fromObject methods of this class.

Return Value

Type	Description
Array	The map of objects

toObject

Enables you to manipulate the core object data in addition to the setters that are called automatically by the parent class implementation.

```
public function toObject($dbObject = null, $skip = array())
```

Remarks

Implementing this method is not mandatory.

Context

Called by the API services.

Parameters

Name	Input/Output	Type	Description
`dbObject`	Input (optional)	`DistributionProfile`	The database target object
`skip`	Input	Array	Attributes to be skipped

Return Value

Type	Description
`DistributionProfile`	The database target object

fromObject

Enables you to manipulate the API object data in addition to the attributes that are populated automatically by the parent class implementation.

```
public function fromObject($sourceObject)
```

Remarks

Implementing this method is not mandatory.

Context

Called by the API services.

Parameters

Name	Input/Output	Type	Description
`sourceObject`	Input	`DistributionProfile`	The database source object

CHAPTER 5

Implementing Kaltura Plugins

A Kaltura server plugin for a distribution provider must implement the following plugins:

- `IKalturaContentDistributionProvider`
- `IKalturaEnumerator`
- `IKalturaEventConsumers`
- `IKalturaObjectLoader`
- `IKalturaPending`
- `IKalturaPermissions`

Plugins are implemented for:

- Kaltura Core
- API
- Admin Console
- Batch

IKalturaContentDistributionProvider

Extends

`IKalturaBase`

Actions

Name	Description
`getProvider`	Returns the singelton instance of the plugin distribution provider.
`getKalturaProvider`	Returns an instance of a Kaltura API distribution provider that represents the singleton instance of the plugin distribution provider.
`contributeMRSS`	Appends nodes and attributes associated with a specific distribution provider and entry to entry to the Kaltura MRSS XML.

getProvider

Returns the singelton instance of the plugin distribution provider.

```
public static function getProvider();
```

Context

- Exposed through the distribution profile.
- Called by the content distribution flow manager to get the provider and to call the provider

methods, supported features, and disabled features. The purpose is to determine what the provider can do, for example, delete or update.

Return Value

Type	Description
`IDistributionProvider`	The singelton instance of the plugin distribution provider

getKalturaProvider

Returns an instance of a Kaltura API distribution provider that represents the singleton instance of the plugin distribution provider.

```
public static function getKalturaProvider();
```

Context

- Called by the DistributionProvider API service in the list action to expose the provider as an API object.

Return Value

Type	Description
`KalturaDistributionProvider`	An instance of a Kaltura API distribution provider that represents the plugin distribution provider singleton instance

contributeMRSS

Appends nodes and attributes associated with a specific distribution provider and entry to the Kaltura MRSS XML.

```
public static function contributeMRSS(EntryDistribution $entryDistribution,
   SimpleXMLElement $mrss);
```

Remarks

The action is used to add provider-specific data to the generated MRSS. The data can be used later in an XSL transformation to specify a data structure to send to the provider destination site.

Context

Called by the content distribution MRSS contributor to append provider-specific data to the MRSS XML.

Parameters

Name	Input/Output	Type	Description
`entryDistribution`	Input	`EntryDistribution`	The distribution entry whose data is appended to the MRSS
`mrss`	Input	`SimpleXMLElement`	The MRSS to which the data is appended

IKalturaEnumerator

Extends

`IKalturaBase`

getEnums

Returns a list of enumeration class names that implement the `baseEnumName` interface.

```
public static function getEnums($baseEnumName);
```

Remarks

Plugins may add enumeration values to those used by the Kaltura core's `baseEnumName` interface. You implement `baseEnumName` by defining a class for one or more additional enum values. The `getEnums` action returns a list of the class names that you define to implement `baseEnumName`. This enables the plugin API to receive enumeration values that other plugins define, in addition to the values that the core defines.

Context

Called by the API client generator and document generator to expose the enumeration values that are added to the original enum values.

Parameters

Name	Input/Output	Type	Description
`baseEnumName`	Input	String, interface name	The core interface that defines enum values

Return Value

Type	Description
Array	A string listing the enum class names that extend `baseEnumName`

IKalturaEventConsumers

Extends

`IKalturaBase`

getEventConsumers

Retrieves the event consumers used by the plugin.

```
public static function getEventConsumers();
```

Remarks

An event consumer implements the event consumer interfaces according to the events it desires to consume. The consumer interface always requires implementing the method that is called whenever the event is raised. Implementing the method enables the plugin the react to the event raised in that method.

Context

Called by the event manager to call all of the event consumers that are relevant to the raised event.

Return Value

Type	Description
Array	The list of event consumers

IKalturaObjectLoader

Extends

`IKalturaBase`

Actions

Name	Description
loadObject	Returns an object that is known only to the plugin, and extends the `baseClass`.
getObjectClass	Retrieves a class name that is defined by the plugin and is known only to the plugin, and extends the `baseClass`.

loadObject

Returns an object that is known only to the plugin, and extends the `baseClass`.

```
public static function loadObject($baseClass, $enumValue, array $constructorArgs =
   null);
```

Context

- Called by the Core `DistributionProfilePeer` to load the relevant

NOTE: All exceptions mark the job as failed except KalturaDistributionException, which marks the job for retry.

- DistributionProfile object.
- Called by each of the batch `KAsyncDistribute` extensions to load the `DistributionEngine` (see `IDistributionEngine`) that is relevant to the job type and the provider.
- Called by the Admin Console `DistributionProfileConfigureAction` class to load the form configuration that extends Form_ProviderProfileConfiguration and is relevant to a specific configured provider.
- Called by the API `KalturaDistributionJobData` object when translated from a core object to load the KalturaDistributionJobProviderData object that is relevant to the provider.
- Called by the API `KalturaDistributionJobData` object when translated to a core object to load the kDistributionJobProviderData object that is relevant to the provider.
- Called by the API `KalturaDistributionProfileFactory` to load the correct KalturaDistributionProfile that is relevant to the provider.

Parameters

Name	Input/Output	Type	Description
`baseClass`	Input	String	The base class of the loaded object
`enumValue`	Input	String	The enumeration value of the loaded object
`constructorArgs`	Input (optional)	Array	The constructor arguments of the loaded object

Return Value

Type	Description
Object	The loaded object instance

getObjectClass

Retrieves a class name that is defined by the plugin and is known only to the plugin, and extends the `baseClass`.

```
public static function getObjectClass($baseClass, $enumValue);
```

Context

- Called by the Core `DistributionProfilePeer` to load the relevant

NOTE: All exceptions mark the job as failed except KalturaDistributionException, which marks the job for retry.

- DistributionProfile object.
- Called by each of the batch `KAsyncDistribute` extensions to load the `DistributionEngine` (see `IDistributionEngine`) that is relevant to the job type and the provider.
- Called by the Admin Console `DistributionProfileConfigureAction` class to load the form configuration that extends Form_ProviderProfileConfiguration and is relevant to a specific configured provider.
- Called by the API `KalturaDistributionJobData` object when translated from a core object to load the KalturaDistributionJobProviderData object that is relevant to the provider.
- Called by the API `KalturaDistributionJobData` object when translated to a core object to load the kDistributionJobProviderData object that is relevant to the provider.
- Called by the API `KalturaDistributionProfileFactory` to load the correct KalturaDistributionProfile that is relevant to the provider.

Parameters

Name	Input/Output	Type	Description
`baseClass`	Input	String	The base class of the searched class
`enumValue`	Input	String	The enumeration value of the searched class

Return Value

Type	Description
String	The name of the searched object's class

IKalturaPending

Extends

`IKalturaBase`

dependsOn

Returns a Kaltura dependency object that defines the relationship between two plugins.

```
public static function dependsOn();
```

Context

Called by the plugin manager to check whether all required plugins are enabled.

Return Value

Type	Description
Array	The Kaltura dependency object

IKalturaPermissions

Extends

`IKalturaBase`

isAllowedPartner

Grants or denies a partner permission to use a plugin.

```
public static function isAllowedPartner($partnerId);
```

Context

- Called by the API to decide whether a specific partner is allowed to use the API.
- Called by the content distribution flow manager to decide whether a specific event is relevant to the partner.

Parameters

Name	Input/Output	Type	Description
`partnerId`	Input	Integer	The ID of the partner being checked for permission

Return Value

Type	Description
Boolean	True — The partner is allowed to use the plugin. False — The partner is not allowed to use the plugin.

CHAPTER 6

Database Schema

A Kaltura server plugin for a distribution provider uses the following database tables to store configuration data:

- Distribution Profile
- Entry Distribution

Distribution Profile

distribution_profile

Implements `ISyncableFile` to enable synchronizing SSH public key files.

Column	Type	Required/ Optional	Foreign Key	Description
id	Integer	Required		The unique identifier of the distribution profile, automatically generated by the database server
created_at	Timestamp	Required		The date and time when the distribution profile was created, automatically generated by the ORM
updated_at	Timestamp	Required		The date and time when the distribution profile was last modified, automatically generated by the ORM
partner_id	Integer	Required	partner.id	A numeric identifier that uniquely identifies the partner in the Kaltura database

Column	Type	Required/ Optional	Foreign Key	Description
provider_type	`DistributionProviderType` (Integer)	Required		Pluginable enum that indicates the type of provider. This helps instantiate the correct `DistributionProfile` or any object that extends it.
name	Varchar	Required		The name of the distribution profile
status	`DistributionProfileStatus` (Tinyint)	Required		The status of the distribution profile. Possible values: Enabled Disabled Deleted
submit_enabled	`DistributionProfileActionStatus` (Tinyint)	Required		The distribution engine supports the `submit` method. Possible values: Manual Automatic Disabled
update_enabled	`DistributionProfileActionStatus` (Tinyint)	Required		The distribution engine supports the `update` method. Possible values: Manual Automatic Disabled
delete_enabled	`DistributionProfileActionStatus` (Tinyint)	Required		The distribution engine supports the `delete` method. Possible values:

Column	Type	Required/ Optional	Foreign Key	Description
				Manual Automatic Disabled
report_enabled	`DistributionProfileActionStatus` (Tinyint)	Required		The distribution engine supports the `fetchReport` method. Possible values: Manual Automatic Disabled
auto_create_flavors	Varchar	Optional		List of comma-separated flavor parameter IDs. Upon submission, the flavors must be automatically converted if they are missing.
auto_create_thumb	Varchar	Optional		List of comma-separated thumbnail parameter IDs. Upon submission, the thumbnails must be automatically converted if they are missing.
optional_flavor_params_ids	Varchar	Optional	flavor_params.id	List of comma-separated flavor parameter IDs. If the flavors exist, they will be submitted.
required_flavor_params_ids	Varchar	Optional	flavor_params.id	List of comma-separated flavor parameter IDs that are required for submission
optional_thumb_dimensions	`array<kDistributionThumbDimensions>` (Varchar)	Optional		Serialized array of thumbnail dimensions. If the thumbnails exist, they will be submitted.
required_thumb_dimensions	`array<kDistributionThumbDimensions>` (Varchar)	Optional		Serialized array of thumbnail dimensions that are required for

Column	Type	Required/ Optional	Foreign Key	Description
				submission
report_interval	Integer	Optional		Interval (in days) between reports that are automatically fetched from the destination provider
custom_data	`myCustomData` (String)	Optional		Container for customized properties. The properties are saved as a serialized PHP object of the `myCustomData` class.

Customized Properties

All customized property columns are optional.

Column	Type	Description
configVersion	Integer	File sync version of the synchronized *configuration* file

Indexes

Index Name	Index Columns	Context
partner_id	partner_id	Used by the Admin Console to list all profiles that belong to a specific partner
partner_status	• partner_id • status	Used by the KMC to find all of the active profiles of the current partner
partner_status_provider	• partner_id • status • provider_type	Used by an event consumer to find all active profiles that can be sent automatically

Entry Distribution

entry_distribution

Implements `ISyncableFile` to enable synchronizing result files.

Implements `IIndexable` to be indexed in the search engine. This makes it easier for the synchronizer batch to find entry distributions that require new batches.

Column	Type	Required/ Optional	Foreign Key	Description
id	Integer	Required		The unique identifier of the entry distribution, automatically generated by the database server
created_at	Timestamp	Required		The date and time when the entry distribution was created, automatically generated by the ORM
updated_at	Timestamp	Required		The date and time when the entry distribution was last modified in the database. This does not relate to modification in the remote destination, automatically generated by the ORM.
submitted_at	Timestamp	Optional		The date and time when the entry distribution submission is approved
entry_id	Varchar	Required	entry.id	The unique identifier of the media entry
partner_id	Integer	Required	partner.id	A numeric identifier that uniquely identifies the partner in the Kaltura database
distribution_profile_id	Integer	Required	distribution_profile.id	The unique identifier of the distribution profile
status	`EntryDistributionStatus` (Tinyint)	Required		The status of the entry distribution. Possible values: • Pending – submit not requested yet • Queued – waiting for required data such as metadata, thumbnails, or flavors • Ready • Deleted • Submitting – submit job created • Updating – update job created • Deleting – delete job created • Error Submitting • Error Updating

Column	Type	Required/ Optional	Foreign Key	Description
				• Error Deleting
dirty_status	`EntryDistributionDirtyStatus` (Tinyint)	Optional		The dirty status of the entry distribution. Possible values: • Submit required – a submit job will be created upon sunrise. The job will be created before sunrise, and the provider defines the interval. • Delete required – a delete job will be created upon sunset. The job will be created before sunset, and the provider defines the interval. • Update required – since update jobs always are created immediately, this flag will be dirty only if the provider does not support the update method. • None –marked in the database with a null value
thumb_asset_ids	Varchar	Optional		A comma-separated list of thumbnail asset identifiers that must be sent to the destination site
flavor_asset_ids	Varchar	Optional		A comma-separated list of flavor asset identifiers that must be sent to the destination site
sunrise	Timestamp	Optional		The time that the entry becomes accessible to end users in the destination site
sunset	Timestamp	Optional		The time that the entry becomes inaccessible to end users in the destination site
remote_id	Varchar	Optional		The identifier of the entry that returned from the provider. The ID represents the package in the destination site.
plays	Integer	Optional		The number of times the entry played from the destination site. The provider returns this value.
views	Integer	Optional		The number of times the entry was viewed from the destination site. The provider returns this value.
validation_errors	`array<kDistributionValidationE`	Optional		A serialized array of `kDistributionValidationError` objects

Column	Type	Required/ Optional	Foreign Key	Description
	`rror>` (Longvarchar)			
error_type	`KalturaBatchJob ErrorTypes` (Integer)	Optional		The type of reported error, such as FTP, HTTP, Runtime, or Curl
error_number	Integer	Optional		The number of a reported error
error_description	Varchar	Optional		The description of a reported error
last_report	Timestamp	Optional		The date and time of the latest report
custom_data	String	Optional		Container for customized properties. The properties are saved as a serialized PHP object of the `myCustomData` class.

Customized Properties

All customized property columns are optional.

Column	Type	Description
SubmitResultsVersion	Integer	File sync version of the synchronized *submission results* file
UpdateResultsVersion	Integer	File sync version of the synchronized *update results* file
DeleteResultsVersion	Integer	File sync version of the synchronized *deletion results* file
SubmitDataVersion	Integer	File sync version of the synchronized *submission sent* data file
UpdateDataVersion	Integer	File sync version of the synchronized *update sent* data file
DeleteDataVersion	Integer	File sync version of the synchronized *deletion sent* data file

Indexes

Index Name	Index Columns	Context
partner_entry_profile	• partner_id • entry_id • distribution_profile_id	Used by the KMC in the entry drill-down list for all entry distributions of all distribution profiles with their statuses

Index Name	Index Columns	Context
partner_profile_status	• partner_id • distribution_profile_id • status	Used by the KMC to show a cross-entry report on a specific distribution profile. The report may be filtered according to a specific status or may display all statuses.

CHAPTER 7

Sample Code

ExampleDistributionPlugin.php

```
<?php
/**
 * @package plugins.exampleDistribution
 */
class ExampleDistributionPlugin extends KalturaPlugin implements IKalturaPermissions, IKalturaEnumerator, IKalturaPending,
   IKalturaObjectLoader, IKalturaContentDistributionProvider
{
   const PLUGIN_NAME = 'exampleDistribution';
   const CONTENT_DSTRIBUTION_VERSION_MAJOR = 1;
   const CONTENT_DSTRIBUTION_VERSION_MINOR = 0;
   const CONTENT_DSTRIBUTION_VERSION_BUILD = 0;

   public static function getPluginName()
   {
       return self::PLUGIN_NAME;
   }

   public static function dependsOn()
   {
       $contentDistributionVersion = new KalturaVersion(
              self::CONTENT_DSTRIBUTION_VERSION_MAJOR,
              self::CONTENT_DSTRIBUTION_VERSION_MINOR,
              self::CONTENT_DSTRIBUTION_VERSION_BUILD);

       $dependency = new KalturaDependency(ContentDistributionPlugin::getPluginName(), $contentDistributionVersion);
       return array($dependency);
   }
```

```
public static function isAllowedPartner($partnerId)
{
    if($partnerId == Partner::ADMIN_CONSOLE_PARTNER_ID)
           return true;

    $partner = PartnerPeer::retrieveByPK($partnerId);
    return $partner->getPluginEnabled(ContentDistributionPlugin::getPluginName());
}

/**
 * @return array<string> list of enum classes names that extend the base enum name
 */
public static function getEnums($baseEnumName)
{
    if($baseEnumName == 'DistributionProviderType')
           return array('ExampleDistributionProviderType');

    return array();
}

/**
 * @param string $baseClass
 * @param string $enumValue
 * @param array $constructorArgs
 * @return object
 */
public static function loadObject($baseClass, $enumValue, array $constructorArgs = null)
{
    // client side apps like batch and admin console
    if (class_exists('KalturaClient') && $enumValue == KalturaDistributionProviderType::EXAMPLE)
    {
           if($baseClass == 'IDistributionEngineCloseDelete')
                  return new ExampleDistributionEngine();

           if($baseClass == 'IDistributionEngineCloseSubmit')
                  return new ExampleDistributionEngine();

           if($baseClass == 'IDistributionEngineCloseUpdate')
                  return new ExampleDistributionEngine();

           if($baseClass == 'IDistributionEngineDelete')
                  return new ExampleDistributionEngine();
```

```
            if($baseClass == 'IDistributionEngineReport')
                  return new ExampleDistributionEngine();

            if($baseClass == 'IDistributionEngineSubmit')
                  return new ExampleDistributionEngine();

            if($baseClass == 'IDistributionEngineUpdate')
                  return new ExampleDistributionEngine();

            if($baseClass == 'Form_ProviderProfileConfiguration')
            {
                  $reflect = new ReflectionClass('Form_ExampleProfileConfiguration');
                  return $reflect->newInstanceArgs($constructorArgs);
            }

            if($baseClass == 'KalturaDistributionProfile')
                  return new KalturaExampleDistributionProfile();

            if($baseClass == 'KalturaDistributionJobProviderData')
                  return new KalturaExampleDistributionJobProviderData();
    }

    // content distribution does not work in partner services 2 context because it uses dynamic enums
    if (!class_exists('kCurrentContext') || kCurrentContext::$ps_vesion != 'ps3')
          return null;

    if($baseClass == 'KalturaDistributionJobProviderData' && $enumValue ==
self::getDistributionProviderTypeCoreValue(ExampleDistributionProviderType::EXAMPLE))
    {
          $reflect = new ReflectionClass('KalturaExampleDistributionJobProviderData');
          return $reflect->newInstanceArgs($constructorArgs);
    }

    if($baseClass == 'kDistributionJobProviderData' && $enumValue ==
self::getApiValue(ExampleDistributionProviderType::EXAMPLE))
    {
          $reflect = new ReflectionClass('kExampleDistributionJobProviderData');
          return $reflect->newInstanceArgs($constructorArgs);
    }

    if($baseClass == 'KalturaDistributionProfile' && $enumValue ==
self::getDistributionProviderTypeCoreValue(ExampleDistributionProviderType::EXAMPLE))
          return new KalturaExampleDistributionProfile();
```

```
    if($baseClass == 'DistributionProfile' && $enumValue ==
self::getDistributionProviderTypeCoreValue(ExampleDistributionProviderType::EXAMPLE))
          return new ExampleDistributionProfile();

    return null;
}

/**
 * @param string $baseClass
 * @param string $enumValue
 * @return string
 */
public static function getObjectClass($baseClass, $enumValue)
{
    // client side apps like batch and admin console
    if (class_exists('KalturaClient') && $enumValue == KalturaDistributionProviderType::EXAMPLE)
    {
          if($baseClass == 'IDistributionEngineCloseDelete')
                return 'ExampleDistributionEngine';

          if($baseClass == 'IDistributionEngineCloseSubmit')
                return 'ExampleDistributionEngine';

          if($baseClass == 'IDistributionEngineCloseUpdate')
                return 'ExampleDistributionEngine';

          if($baseClass == 'IDistributionEngineDelete')
                return 'ExampleDistributionEngine';

          if($baseClass == 'IDistributionEngineReport')
                return 'ExampleDistributionEngine';

          if($baseClass == 'IDistributionEngineSubmit')
                return 'ExampleDistributionEngine';

          if($baseClass == 'IDistributionEngineUpdate')
                return 'ExampleDistributionEngine';

          if($baseClass == 'Form_ProviderProfileConfiguration')
                return 'Form_ExampleProfileConfiguration';

          if($baseClass == 'KalturaDistributionProfile')
                return 'KalturaExampleDistributionProfile';
```

```
            if($baseClass == 'KalturaDistributionJobProviderData')
                  return 'KalturaExampleDistributionJobProviderData';
    }

    // content distribution does not work in partner services 2 context because it uses dynamic enums
    if (!class_exists('kCurrentContext') || kCurrentContext::$ps_vesion != 'ps3')
           return null;

    if($baseClass == 'KalturaDistributionJobProviderData' && $enumValue ==
self::getDistributionProviderTypeCoreValue(ExampleDistributionProviderType::EXAMPLE))
           return 'KalturaExampleDistributionJobProviderData';

    if($baseClass == 'kDistributionJobProviderData' && $enumValue ==
self::getApiValue(ExampleDistributionProviderType::EXAMPLE))
           return 'kExampleDistributionJobProviderData';

    if($baseClass == 'KalturaDistributionProfile' && $enumValue ==
self::getDistributionProviderTypeCoreValue(ExampleDistributionProviderType::EXAMPLE))
           return 'KalturaExampleDistributionProfile';

    if($baseClass == 'DistributionProfile' && $enumValue ==
self::getDistributionProviderTypeCoreValue(ExampleDistributionProviderType::EXAMPLE))
           return 'ExampleDistributionProfile';

    return null;
}

/**
 * Return a distribution provider instance
 *
 * @return IDistributionProvider
 */
public static function getProvider()
{
    return ExampleDistributionProvider::get();
}

/**
 * Return an API distribution provider instance
 *
 * @return KalturaDistributionProvider
 */
public static function getKalturaProvider()
{
```

```
        $distributionProvider = new KalturaExampleDistributionProvider();
        $distributionProvider->fromObject(self::getProvider());
        return $distributionProvider;
    }

    /**
     * Append provider specific nodes and attributes to the MRSS
     *
     * @param EntryDistribution $entryDistribution
     * @param SimpleXMLElement $mrss
     */
    public static function contributeMRSS(EntryDistribution $entryDistribution, SimpleXMLElement $mrss)
    {

    }

    /**
     * @return int id of dynamic enum in the DB.
     */
    public static function getDistributionProviderTypeCoreValue($valueName)
    {
        $value = self::getPluginName() . IKalturaEnumerator::PLUGIN_VALUE_DELIMITER . $valueName;
        return kPluginableEnumsManager::apiToCore('DistributionProviderType', $value);
    }

    /**
     * @return string external API value of dynamic enum.
     */
    public static function getApiValue($valueName)
    {
        return self::getPluginName() . IKalturaEnumerator::PLUGIN_VALUE_DELIMITER . $valueName;
    }
}
```

lib

ExampleDistributionEngine.php

```
<?php
/**
```

```
 * @package plugins.exampleDistribution
 * @subpackage lib
 */
class ExampleDistributionEngine extends DistributionEngine implements
   IDistributionEngineUpdate,
   IDistributionEngineSubmit,
   IDistributionEngineReport,
   IDistributionEngineDelete,
   IDistributionEngineCloseSubmit
{
   const FTP_SERVER_URL = 'example.ftp.com';

   /**
    * Demonstrate using batch configuration
    * Contains the path to the update XML template file
    * @var string
    */
   private $updateXmlTemplate;

   /* (non-PHPdoc)
    * @see DistributionEngine::configure()
    */
   public function configure(KSchedularTaskConfig $taskConfig)
   {
       // set default value
       $this->updateXmlTemplate = dirname(__FILE__) . '/../xml/update.template.xml';

       // load value from batch configuration
       if($taskConfig->params->updateXmlTemplate)
              $this->updateXmlTemplate = $taskConfig->params->updateXmlTemplate;
   }

   /* (non-PHPdoc)
    * @see IDistributionEngineSubmit::submit()
    *
    * Demonstrate asynchronous external API usage
    */
   public function submit(KalturaDistributionSubmitJobData $data)
   {
       // validates received object types

       if(!$data->distributionProfile || !($data->distributionProfile instanceof KalturaExampleDistributionProfile))
              KalturaLog::err("Distribution profile must be of type KalturaExampleDistributionProfile");
```

```
        if(!$data->providerData || !($data->providerData instanceof KalturaExampleDistributionJobProviderData))
                KalturaLog::err("Provider data must be of type KalturaExampleDistributionJobProviderData");

        // call the actual submit action
        $this->handleSubmit($data, $data->distributionProfile, $data->providerData);

        // always return false to be closed asynchronously by the closer
        return false;
    }

    /* (non-PHPdoc)
     * @see IDistributionEngineCloseSubmit::closeSubmit()
     */
    public function closeSubmit(KalturaDistributionSubmitJobData $data)
    {
        return ExampleExternalApiService::wasSubmitSucceed($data->remoteId);
    }

    /* (non-PHPdoc)
     * @see IDistributionEngineDelete::delete()
     */
    public function delete(KalturaDistributionDeleteJobData $data)
    {
        // demonstrate asynchronous XML delivery usage from XSL

        if(!$data->distributionProfile || !($data->distributionProfile instanceof KalturaExampleDistributionProfile))
                KalturaLog::err("Distribution profile must be of type KalturaExampleDistributionProfile");

        if(!$data->providerData || !($data->providerData instanceof KalturaExampleDistributionJobProviderData))
                KalturaLog::err("Provider data must be of type KalturaExampleDistributionJobProviderData");

        $this->handleDelete($data, $data->distributionProfile, $data->providerData);

        return false;
    }

    /* (non-PHPdoc)
     * @see IDistributionEngineUpdate::update()
     *
     * demonstrate asynchronous XML delivery usage from template and uploading the media
     */
    public function update(KalturaDistributionUpdateJobData $data)
    {
        if(!$data->distributionProfile || !($data->distributionProfile instanceof KalturaExampleDistributionProfile))
```

```
            KalturaLog::err("Distribution profile must be of type KalturaExampleDistributionProfile");

    if(!$data->providerData || !($data->providerData instanceof KalturaExampleDistributionJobProviderData))
            KalturaLog::err("Provider data must be of type KalturaExampleDistributionJobProviderData");

    $this->handleUpdate($data, $data->distributionProfile, $data->providerData);

    return false;
}

/* (non-PHPdoc)
 * @see IDistributionEngineReport::fetchReport()
 *
 * Demonstrate asynchronous http url parsing
 */
public function fetchReport(KalturaDistributionFetchReportJobData $data)
{
    // TODO
    return false;
}

/**
 * @param KalturaDistributionJobData $data
 * @param KalturaExampleDistributionProfile $distributionProfile
 * @param KalturaExampleDistributionJobProviderData $providerData
 */
protected function handleSubmit(KalturaDistributionJobData $data, KalturaExampleDistributionProfile $distributionProfile,
KalturaExampleDistributionJobProviderData $providerData)
{
    $entryId = $data->entryDistribution->entryId;
    $partnerId = $distributionProfile->partnerId;
    $entry = $this->getEntry($partnerId, $entryId);

    // populate the external API object with the Kaltura entry data
    $exampleExternalApiMediaItem = new ExampleExternalApiMediaItem();
    $exampleExternalApiMediaItem->resourceId = $entry->id;
    $exampleExternalApiMediaItem->title = $entry->name;
    $exampleExternalApiMediaItem->description = $entry->description;
    $exampleExternalApiMediaItem->width = $entry->width;
    $exampleExternalApiMediaItem->height = $entry->height;

    // loads ftp manager
    $ftpManager = kFileTransferMgr::getInstance(kFileTransferMgrType::FTP);
    $ftpManager->login(self::FTP_SERVER_URL, $distributionProfile->username, $distributionProfile->password);
```

```
    // put the thumbnail on the FTP with the entry id as naming convention
    $remoteFile = $entry->id . '.jpg';
    $ftpManager->putFile($remoteFile, $providerData->thumbAssetFilePath);

    // put the video files on the FTP with the entry id as naming convention and index
    foreach($providerData->videoAssetFilePaths as $index => $videoAssetFilePath)
    {
           $localPath = $videoAssetFilePath->path;
           $pathInfo = pathinfo($localPath);
           $fileExtension = $pathInfo['extension'];

           $remoteFile = "{$entry->id}-{$index}.{$fileExtension}";
           $ftpManager->putFile($remoteFile, $localPath);
    }

    $remoteId = ExampleExternalApiService::submit($exampleExternalApiMediaItem);
    $data->remoteId = $remoteId;
}

/**
 * @param KalturaDistributionJobData $data
 * @param KalturaExampleDistributionProfile $distributionProfile
 * @param KalturaExampleDistributionJobProviderData $providerData
 */
protected function handleDelete(KalturaDistributionJobData $data, KalturaExampleDistributionProfile $distributionProfile,
KalturaExampleDistributionJobProviderData $providerData)
{
    // TODO
}

/**
 * @param KalturaDistributionJobData $data
 * @param KalturaExampleDistributionProfile $distributionProfile
 * @param KalturaExampleDistributionJobProviderData $providerData
 */
protected function handleUpdate(KalturaDistributionJobData $data, KalturaExampleDistributionProfile $distributionProfile,
KalturaExampleDistributionJobProviderData $providerData)
{
    $entryId = $data->entryDistribution->entryId;
    $partnerId = $distributionProfile->partnerId;
    $entry = $this->getEntry($partnerId, $entryId);

    $feed = new DOMDocument();
```

```
	$feed->load($this->updateXmlTemplate);
	$feed->documentElement->setAttribute('mediaId', $data->remoteId);

	$nodes = array(
		'title' => 'name',
		'description' => 'description',
		'width' => 'width',
		'height' => 'height',
	);
	foreach($nodes as $nodeName => $entryAttribute)
	{
		$nodeElements = $feed->getElementsByTagName($nodeName);
		foreach($nodeElements as $nodeElement)
			$nodeElement->textContent = $entry->$entryAttribute;
	}

	// get the first asset id
	$thumbAssetIds = explode(',', $data->entryDistribution->thumbAssetIds);
	$thumbAssetId = reset($thumbAssetIds);
	$thumbElements = $feed->getElementsByTagName('thumb');
	$thumbElement = reset($thumbElements);
	$thumbElement->textContent = $this->getThumbAssetUrl($thumbAssetId);

	$videosElements = $feed->getElementsByTagName('videos');
	$videosElement = reset($videosElements);

	$flavorAssets = $this->getFlavorAssets($partnerId, $data->entryDistribution->flavorAssetIds);
	$this->impersonate($partnerId);
	foreach($flavorAssets as $flavorAsset)
	{
		$url = $this->kalturaClient->flavorAsset->getDownloadUrl($flavorAsset->id, true);

		$videoElement = $feed->createElement('video');
		$videoElement->textContent = $url;
		$videosElement->appendChild($videoElement);
	}
	$this->unimpersonate();


	$localFile = tempnam(sys_get_temp_dir(), 'example-update-');
	$feed->save($localFile);

	// loads ftp manager
	$ftpManager = kFileTransferMgr::getInstance(kFileTransferMgrType::FTP);
```

```
        $ftpManager->login(self::FTP_SERVER_URL, $distributionProfile->username, $distributionProfile->password);

        // put the XML file on the FTP
        $remoteFile = $entryId . '.xml';
        $ftpManager->putFile($remoteFile, $localFile);

        return true;
    }
}
```

ExampleDistributionProfile.php

```
<?php
/**
 * @package plugins.exampleDistribution
 * @subpackage model
 */
class ExampleDistributionProfile extends DistributionProfile
{
    const CUSTOM_DATA_USERNAME = 'username';
    const CUSTOM_DATA_PASSWORD = 'password';
    const CUSTOM_DATA_ACCOUNT_ID = 'accountId';

    /* (non-PHPdoc)
     * @see DistributionProfile::getProvider()
     */
    public function getProvider()
    {
        return ExampleDistributionPlugin::getProvider();
    }

    public function getUsername()                    {return $this->getFromCustomData(self::CUSTOM_DATA_USERNAME);}
    public function getPassword()                    {return $this->getFromCustomData(self::CUSTOM_DATA_PASSWORD);}
    public function getAccountId()             {return $this->getFromCustomData(self::CUSTOM_DATA_ACCOUNT_ID);}

    public function setUsername($v)                  {$this->putInCustomData(self::CUSTOM_DATA_USERNAME, $v);}
    public function setPassword($v)                  {$this->putInCustomData(self::CUSTOM_DATA_PASSWORD, $v);}
    public function setAccountId($v)                 {$this->putInCustomData(self::CUSTOM_DATA_ACCOUNT_ID, $v);}

}
```

ExampleDistributionProvider.php

```
<?php
/**
 * @package plugins.exampleDistribution
 * @subpackage lib
 */
class ExampleDistributionProvider implements IDistributionProvider
{
   /**
    * @var ExampleDistributionProvider
    */
   protected static $instance;

   protected function __construct()
   {

   }

   /**
    * @return ExampleDistributionProvider
    */
   public static function get()
   {
       if(!self::$instance)
              self::$instance = new ExampleDistributionProvider();

       return self::$instance;
   }

   /* (non-PHPdoc)
    * @see IDistributionProvider::getType()
    */
   public function getType()
   {
       return ExampleDistributionPlugin::getDistributionProviderTypeCoreValue(ExampleDistributionProviderType::EXAMPLE);
   }

   /* (non-PHPdoc)
    * @see IDistributionProvider::getName()
    */
   public function getName()
   {
```

```
    return 'Example';
}

/* (non-PHPdoc)
 * @see IDistributionProvider::isDeleteEnabled()
 */
public function isDeleteEnabled()
{
    return true;
}

/* (non-PHPdoc)
 * @see IDistributionProvider::isUpdateEnabled()
 */
public function isUpdateEnabled()
{
    return true;
}

/* (non-PHPdoc)
 * @see IDistributionProvider::isReportsEnabled()
 */
public function isReportsEnabled()
{
    return true;
}

/* (non-PHPdoc)
 * @see IDistributionProvider::isScheduleUpdateEnabled()
 */
public function isScheduleUpdateEnabled()
{
    return true;
}

/* (non-PHPdoc)
 * @see IDistributionProvider::useDeleteInsteadOfUpdate()
 */
public function useDeleteInsteadOfUpdate()
{
    // irrelevant because update is enabled
    return false;
}
```

```
/* (non-PHPdoc)
 * @see IDistributionProvider::getJobIntervalBeforeSunrise()
 */
public function getJobIntervalBeforeSunrise()
{
    // irrelevant because sending schedule is supported
    return 0;
}

/* (non-PHPdoc)
 * @see IDistributionProvider::getJobIntervalBeforeSunset()
 */
public function getJobIntervalBeforeSunset()
{
    // irrelevant because sending schedule is supported
    return 0;
}

/* (non-PHPdoc)
 * @see IDistributionProvider::getUpdateRequiredEntryFields()
 */
public function getUpdateRequiredEntryFields($distributionProfileId = null)
{
    return array(
           // entry columns
           entryPeer::NAME,
           entryPeer::DESCRIPTION,
           entryPeer::TAGS,
           entryPeer::CATEGORIES,

           // customized properties
           'width',
           'height',
    );
}

/* (non-PHPdoc)
 * @see IDistributionProvider::getUpdateRequiredMetadataXPaths()
 */
public function getUpdateRequiredMetadataXPaths($distributionProfileId = null)
{
    return array(
           "/*[local-name()='metadata']/*[local-name()='ShortDescription']",
           "/*[local-name()='metadata']/*[local-name()='LongDescription']",
```

```
                "/*[local-name()='metadata']/*[local-name()='ExampleTitle']",
        );
    }
}
```

ExampleDistributionProviderType.php

```
<?php
/**
 * @package plugins.exampleDistribution
 * @subpackage lib
 */
class ExampleDistributionProviderType implements IKalturaPluginEnum, DistributionProviderType
{
    const EXAMPLE = 'EXAMPLE';

    public static function getAdditionalValues()
    {
        return array(
                'EXAMPLE' => self::EXAMPLE,
        );
    }
}
```

kExampleDistributionJobProviderData.php

```
<?php
/**
 * @package plugins.exampleDistribution
 * @subpackage model.data
 */
class kExampleDistributionJobProviderData extends kDistributionJobProviderData
{
    /**
     * Demonstrate storing array of paths in the job data
     *
     * @var array<string>
     */
    public $videoAssetFilePaths;

    /**
```

```
    * Demonstrate storing single path in the job data
    *
    * @var string
    */
   public $thumbAssetFilePath;

   public function __construct(kDistributionJobData $distributionJobData = null)
   {
       parent::__construct($distributionJobData);
   }

   /**
    * @return array<string> $videoAssetFilePaths
    */
   public function getVideoAssetFilePaths()
   {
       return $this->videoAssetFilePaths;
   }

   /**
    * @return string $thumbAssetFilePath
    */
   public function getThumbAssetFilePath()
   {
       return $this->thumbAssetFilePath;
   }

   /**
    * @param array<string> $videoAssetFilePaths
    */
   public function setVideoAssetFilePaths(array $videoAssetFilePaths)
   {
       $this->videoAssetFilePaths = $videoAssetFilePaths;
   }

   /**
    * @param string $thumbAssetFilePath
    */
   public function setThumbAssetFilePath($thumbAssetFilePath)
   {
       $this->thumbAssetFilePath = $thumbAssetFilePath;
   }
}
```

api

KalturaExampleDistributionAssetPath.php

```
<?php
/**
 * @package plugins.exampleDistribution
 * @subpackage api.objects
 */
class KalturaExampleDistributionAssetPath extends KalturaDistributionJobProviderData
{
   /**
    * @var string
    */
   public $path;
}
```

KalturaExampleDistributionAssetPathArray.php

```
<?php
/**
 * @package plugins.exampleDistribution
 * @subpackage api.objects
 */
class KalturaExampleDistributionAssetPathArray extends KalturaTypedArray
{
   public static function fromArray($arr)
   {
       $newArr = new KalturaExampleDistributionAssetPathArray();
       if ($arr == null)
              return $newArr;

       foreach ($arr as $obj)
       {
              $nObj = new KalturaExampleDistributionAssetPath();
              $nObj->fromObject($obj);
              $newArr[] = $nObj;
       }

       return $newArr;
```

```
   }

   public function __construct()
   {
       parent::__construct("KalturaExampleDistributionAssetPath");
   }
}
```

KalturaExampleDistributionJobProviderData.php

```
<?php
/**
 * @package plugins.exampleDistribution
 * @subpackage api.objects
 */
class KalturaExampleDistributionJobProviderData extends KalturaDistributionJobProviderData
{
   /**
    * Demonstrate passing array of paths to the job
    *
    * @var KalturaExampleDistributionAssetPathArray
    */
   public $videoAssetFilePaths;

   /**
    * Demonstrate passing single path to the job
    *
    * @var string
    */
   public $thumbAssetFilePath;


   /**
    * Called on the server side and enables you to populate the object with any data from the DB
    *
    * @param KalturaDistributionJobData $distributionJobData
    */
   public function __construct(KalturaDistributionJobData $distributionJobData = null)
   {
       if(!$distributionJobData)
              return;
```

```
	if(!($distributionJobData->distributionProfile instanceof KalturaExampleDistributionProfile))
		return;

	$this->videoAssetFilePaths = new KalturaExampleDistributionAssetPathArray();

	// loads all the flavor assets that should be submitted to the remote destination site
	$flavorAssets = flavorAssetPeer::retrieveByIds(explode(',', $distributionJobData->entryDistribution->flavorAssetIds));
	foreach($flavorAssets as $flavorAsset)
	{
		$videoAssetFilePath = new KalturaExampleDistributionAssetPath();
		$syncKey = $flavorAsset->getSyncKey(flavorAsset::FILE_SYNC_FLAVOR_ASSET_SUB_TYPE_ASSET);
		$videoAssetFilePath->path = kFileSyncUtils::getLocalFilePathForKey($syncKey, true);
		$this->videoAssetFilePaths[] = $videoAssetFilePath;
	}

	$thumbAssets = thumbAssetPeer::retrieveByIds(explode(',', $distributionJobData->entryDistribution->thumbAssetIds));
	if(count($thumbAssets))
	{
		$thumbAsset = reset($thumbAssets);
		$syncKey = $thumbAssets->getSyncKey(thumbAsset::FILE_SYNC_FLAVOR_ASSET_SUB_TYPE_ASSET);
		$this->thumbAssetFilePath = kFileSyncUtils::getLocalFilePathForKey($syncKey, true);
	}
}

/**
 * Maps the object attributes to getters and setters for Core-to-API translation and back
 *
 * @var array
 */
private static $map_between_objects = array
(
    "thumbAssetFilePath",
);

/* (non-PHPdoc)
 * @see KalturaObject::getMapBetweenObjects()
 */
public function getMapBetweenObjects ( )
{
    return array_merge ( parent::getMapBetweenObjects() , self::$map_between_objects );
}

/* (non-PHPdoc)
 * @see KalturaObject::toObject()
```

```
     */
    public function toObject($object = null, $skip = array())
    {
        $object = parent::toObject($object, $skip);

        if($this->videoAssetFilePaths)
        {
               $videoAssetFilePaths = array();
               foreach($this->videoAssetFilePaths as $videoAssetFilePath)
                      $videoAssetFilePaths[] = $videoAssetFilePath->path;

               $object->setVideoAssetFilePaths($videoAssetFilePaths);
        }

        return $object;
    }

    /* (non-PHPdoc)
     * @see KalturaObject::fromObject()
     */
    public function fromObject($object)
    {
        parent::fromObject($object);
        $videoAssetFilePaths = $object->getVideoAssetFilePaths();
        if($videoAssetFilePaths && is_array($videoAssetFilePaths))
        {
               $this->videoAssetFilePaths = new KalturaExampleDistributionAssetPathArray();
               foreach($videoAssetFilePaths as $assetFilePath)
               {
                      $videoAssetFilePath = new KalturaExampleDistributionAssetPath();
                      $videoAssetFilePath->path = $assetFilePath;
                      $this->videoAssetFilePaths[] = $videoAssetFilePath;
               }
        }
    }
}
```

KalturaExampleDistributionProfile.php

```
<?php
/**
 * @package plugins.exampleDistribution
```

```
 * @subpackage api.objects
 */
class KalturaExampleDistributionProfile extends KalturaDistributionProfile
{
   /**
    * @var string
    */
   public $username;

   /**
    * @var string
    */
   public $password;

   /**
    * @var string
    */
   public $accountId;

   /*
    * mapping between the field on this object (on the left) and the setter/getter on the object (on the right)
    */
   private static $map_between_objects = array
   (
       'username',
       'password',
       'accountId',
    );

   public function getMapBetweenObjects()
   {
       return array_merge(parent::getMapBetweenObjects(), self::$map_between_objects);
   }
}
```

KalturaExampleDistributionProvider.php

```
<?php
/**
 * @package plugins.exampleDistribution
 * @subpackage api.objects
 */
```

```
class KalturaExampleDistributionProvider extends KalturaDistributionProvider
{

}
```

ExampleAPI

ExampleExternalApi.php

```
<?php

/**
 * @package plugins.exampleDistribution
 * @subpackage external
 */
class ExampleExternalApiMediaItem
{
   public $resourceId;
   public $title;
   public $description;
   public $width;
   public $height;
}

/**
 * @package plugins.exampleDistribution
 * @subpackage external
 */
class ExampleExternalApiService
{
   /**
    * @param ExampleExternalApiMediaItem $mediaItem
    * @return int media id
    */
   public static function submit(ExampleExternalApiMediaItem $mediaItem)
   {
       // do something
       return rand(1, 1000);
   }

   /**
```

```
    * @param int $mediaItemId
    * @return boolean succeed or not
    */
   public static function wasSubmitSucceed($mediaItemId)
   {
       // do something
       return true;
   }
}
```

xml

update.template.xml

```
update.template.xml
<?xml version="1.0" encoding="UTF-8"?>
<ExampleExternalApiMediaItem mediaId="{fill}">
   <title>{fill}</title>
   <description>{fill}</description>
   <width>{fill}</width>
   <height>{fill}</height>
   <thumb></thumb>
   <videos/>
</ExampleExternalApiMediaItem>
```

GLOSSARY

Term	Definition
Destination	A media provider or video sharing site
Dirty	An item that has been modified but not distributed
Distribution	Publishing media entries in a destination site
Distribution Provider	A module that publishes Kaltura entries in the destination site
Entry	Kaltura's database and API representation of a content entity and its metadata. Entry types include media, video, audio, image, data, mix, document, and playlist. Entry metadata includes type, storage location, title, tag, and rating.
Flavor	A rendition of a video entry with specific attributes intended for playback using Kaltura's Adaptive Bitrate technology. For example, a video intended for both HD and iPhone may require flavors with different attributes such as encoding, bitrate, and resolution.
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.
MRSS	Media RSS file format
ORM	Object-Relational Mapping. Allows access to a database using a set of objects.
Partner	An individual or organization with a Kaltura system account
Partner ID	A numeric identifier that uniquely identifies a partner in the Kaltura database
Sunrise	The time that an entry becomes accessible to end users in the destination site
Sunset	The time that an entry becomes inaccessible to end users in the destination site