BrightSign®

API REFERENCE MANUAL

BrightSign Network Version 3.6

BrightSign, LLC. 16795 Lark Ave., Suite 200 Los Gatos, CA 95032 | 408-852-9263 | www.brightsign.biz

TABLE OF CONTENTS

Introduction 1

Entities 1

Identifiers 1

Dependency 1

Methods 2

Entity Retrieval Methods 2

Entity Update Methods 3

Object Permissions 4

SOAP Endpoints 4

SOAP Access Point URLs 5

User Authentication 6

Development Tools 6

User Authenticate() 6

Content 7

Content Entity 7

ContentFolder Entity 7

ContentType Enumeration 8
ContentTransition Enumeration 8
DynamicPlaylistContent Entity-Relation 9
PresentationContent Entity-Relation 9
DynamicPlaylistInfo Structure 10
PresentationInfo Structure 11
Content Management Web Methods 11
PagedList<Content> GetAllContent(string marker, int pageSize) 11
List<ContentFolder> GetContentFolders(string virtualPath) 12
PagedList<Content> GetFolderContent(string virtualPath, string marker, int pageSize) 12
List<Content> GetSpecifiedContent(int[] contentIds) 13
PagedList<Content> FindContent(string fileNamePattern, string marker, int pageSize) 13
Content GetContent(int contentId) 14
ContentFolder CreateContentFolder(ContentFolder entity) 14
bool MoveContent(int[] contentIds, string newVirtualPath) 15
bool CheckContentUsage(int contentId) 16
bool DeleteContent(int[] contentIds) 16
Content Upload 18
Overview 18
Web Page Upload Work Flow 18

Web Page Update Work Flow 19

Content Upload Web Methods 20

ContentUploadStatus StartFileUpload(string uploadToken, string sessionToken, string filename, long filesize, int chunksCount, enum ContentType, string virtualPath, base64Binary fileThumb, string sha1Hash) 21

ContentUploadStatus AppendChunk(string uploadToken, int partNumber, binary data, long offset) 22

ContentUploadStatus CompleteFileUpload(string uploadToken, string sessionToken, string filename, long filesize, int chunksCount, enum ContentType, string virtualPath, base64Binary fileThumb, string sha1Hash) 23

WebPageUploadStatus StartWebPageUploadSession(array webpageAssets[], string uploadToken, string sessionToken, string filename, long filesize, int chunksCount, enum ContentType, string virtualPath, base64Binary fileThumb, string sha1Hash) 24

WebPageUploadStatus CompleteWebPageUploadSession(string sessionToken) 26

ContentUploadStatus CancelFileUpload(string uploadToken) 28

ContentUploadStatus GetFileUploadStatus(string uploadToken) 29

WebPageUploadStatus GetWebPageUploadStatus(string sessionToken) 29

Dynamic Playlist 31

ImageVideoDynamicPlaylist Entity 31

AudioDynamicPlaylist Entity 31

Dynamic Playlist Management Web Methods 32

PagedList<DynamicPlaylist> GetDynamicPlaylists(string marker, int pageSize) 33

PagedList<ImageVideoDynamicPlaylist> GetImageVideoDynamicPlaylists(string marker, int pageSize) 34

PagedList<AudioDynamicPlaylist> GetAudioDynamicPlaylists(string marker, int pageSize) 34

List<DynamicPlaylist> GetSpecifiedDynamicPlaylists(int[] dynamicPlaylistIds) ... 35
PagedList<DynamicPlaylist> FindDynamicPlaylists(string namePattern, string marker, int pageSize) ... 35
PagedList<ImageVideoDynamicPlaylist> FindImageVideoDynamicPlaylists(string namePattern, string marker, int pageSize) ... 36
PagedList<AudioDynamicPlaylist> FindAudioDynamicPlaylists(string namePattern, string marker, int pageSize) ... 37
DynamicPlaylist GetDynamicPlaylist(int dynamicPlaylistId, bool loadContent) ... 38
DynamicPlaylist GetDynamicPlaylistByName(string name, bool loadContent) ... 38
bool CheckDynamicPlaylistName(string name) ... 39
bool CheckDynamicPlaylistUsage(int dynamicPlaylistId) ... 39
bool CheckDynamicPlaylistUsageByName(string name) ... 40
DynamicPlaylist CreateDynamicPlaylist(DynamicPlaylist entity) ... 40
bool UpdateDynamicPlaylist(DynamicPlaylist entity) ... 41
bool RenameDynamicPlaylist(int dynamicPlaylistId, string newName) ... 42
bool DeleteDynamicPlaylists(int[] dynamicPlaylistIds) ... 42

Live Text Feed ... 44

LiveTextFeed Entity ... 44

Live Text Feed Management Web Methods ... 44

PagedList<LiveTextFeed> GetLiveTextFeeds(string marker, int pageSize) ... 45
List<LiveTextFeed> GetSpecifiedLiveTextFeeds(int[] liveTextFeedIds) ... 45
PagedList<LiveTextFeed> FindLiveTextFeeds(string namePattern, string marker, int pageSize) ... 46

LiveTextFeed GetLiveTextFeed(int liveTextFeedId, bool loadContent) 47
LiveTextFeed GetLiveTextFeedByName(string name, bool loadContent) 47
bool CheckLiveTextFeedName(string name) 48
bool CheckLiveTextFeedUsage(int liveTextFeedId) 48
bool CheckLiveTextFeedUsageByName(string name) 49
LiveTextFeed CreateLiveTextFeed(LiveTextFeed entity) 49
LiveTextFeed CloneLiveTextFeed(string feedUrl) 50
bool UpdateLiveTextFeed(LiveTextFeed entity) 50
bool RenameLiveTextFeed(int liveTextFeedId, string newName) 51
bool DeleteLiveTextFeeds(int[] liveTextFeedIds) 52

Web Page 53

WebPage Entity 53

WebPageAsset Entity-Relation 53

PresentationContent Entity-Relation 54

Web Page Management Web Methods 54

PagedList<WebPage> GetWebPages(string marker, int pageSize) 55
List<WebPage> GetSpecifiedWebPages(int[] webPageIds) 55
PagedList<WebPage> FindWebPages(string namePattern, string marker, int pageSize) 56
WebPage GetWebPage(int webPageId, bool loadAssets) 56

WebPage GetWebPageByName(string name, bool loadContent) 57
bool CheckWebPageName(string name) 57
bool CheckWebPageUsage(int liveTextFeedId) 58
bool CheckWebPageUsageByName(string name) 58
bool RenameWebPage(int webPageId, string newName) 59
bool DeleteWebPages(int[] webPageIds) 59

Device Web Page 61

DeviceWebPage Entity 61

Device Web Page Management Web Methods 61

PagedList<WebPage> GetDeviceWebPages(int marker, int pageSize) 62
List<WebPage> GetSpecifiedDeviceWebPages(int[] webPageIds) 62
PagedList<WebPage> FindDeviceWebPages(string namePattern, int marker, int pageSize) 63
DeviceWebPage GetDeviceWebPage(int webPageId, bool loadAssets) 63
DeviceWebPage GetDeviceWebPageByName(string name, bool loadContent) 64
bool CheckDeviceWebPageName(string name) 64
bool CheckDeviceWebPageUsage(int liveTextFeedId) 65
bool CheckDeviceWebPageUsageByName(string name) 65
bool RenameDeviceWebPage(int webPageId, string newName) 66
bool DeleteDeviceWebPages(int[] webPageIds) 66

Presentation 68

Presentation Entity 68

DeviceModel Enumeration 69

PresentationLanguage Enumeration 69

ScreenSettings Structure 69

PresentationZone Structure 72

VideoOrImagesPresentationZone Structure 72

EnhancedAudioPresentationZone 73

GroupInfo Structure 74

Presentation Management Web Methods 74

PagedList<Presentation> GetPresentations(string marker, int pageSize) 75

List<Presentation> GetSpecifiedPresentations(int[] presentationIds) 76

PagedList<Presentation> FindPresentations(string namePattern, string marker, int pageSize) 76

Presentation GetPresentation(int presentationId, bool loadAssets) 77

Presentation GetPresentationByName(string name, bool loadContent) 78

ScreenResolution[] GetSupportedScreenResolutions(DeviceModel deviceModel, ConnectorType connectorType) 78

bool CheckPresentationName(string name) 79

bool CheckPresentationUsage(int presentationId) 79

Presentation CreatePresentation(Presentation entity) 80

Presentation UpdatePresentation(Presentation entity) 81

bool UpdatePresentationScreenSettings(int presentationId, ScreenSettings screenSettings) 82

bool UpdatePresentationZone(int presentationId, PresentationZone entity) 83

bool DeletePresentations(int[] presentationIds) 84

Group 85

Group Entity 85

Group Management Web Methods 86

PagedList<Group> GetGroups(string marker, int pageSize) 86

List<Group> GetSpecifiedGroups(int[] presentationIds) 87

PagedList<Group> FindGroups(string namePattern, string marker, int pageSize) 87

Group GetGroup (int presentationId, bool loadAssets) 88

Group GetGroupByName(string name, bool loadContent) 88

Group CreateGroup(Group entity) 89

bool DeleteGroup(int groupId, int reassignmentGroupId) 89

Schedule 91

ScheduledPresentation Entity-Relation 91

Schedule Management Web Methods 92

PagedList<ScheduledPresentation> GetGroupSchedule(int groupId, string marker, int pageSize) 92

ScheduledPresentation AddScheduledPresentation(int groupId, ScheduledPresentation entity) 93

bool UpdateScheduledPresentation(ScheduledPresentation entity) 94

bool OverwriteSchedule(int groupId, ScheduledPresentation[] entities) 95

bool RemoveScheduledPresentation(int scheduledPresentationId) 96

Device 98

Device Entity 98

DeviceLogReport Entity 101

DeviceSubscriptionType Enumeration 102

DeviceLogType Enumeration 102

DeviceSubscriptionStatus Enumeration 102

DeviceConnectionsPeriod Enumeration 103

DeviceNetworkSettings Structure 103

DeviceWiredSettings Structure 104

DeviceWirelessSettings Structure 104

DeviceLogsSettings Structure 105

DeviceHealthStatus Enumeration 105

DeviceError Structure 106

DeviceDownload Structure 106

Device Management Web Methods 106

PagedList<Device> GetAllDevices(string marker, int pageSize) 107

List<Device> GetSpecifiedDevices(int[] deviceIds) 108

PagedList<Device> FindDevices(string namePattern, string marker, int pageSize) 108

Device GetDevice(int deviceId) 109

Device GetDeviceBySerial(string serial) 109

PagedList<DeviceError> GetDeviceErrors(int deviceId, string marker, int pageSize) 110

PagedList<DeviceDownload> GetDeviceDownloads(int deviceId, string marker, int pageSize) 110

bool RenameDevice(int deviceId, string newName, string newDescription) 111

bool UpdateDeviceLogsSettings(int deviceId, DeviceLogsSettings settings) 111

bool ReassignDevices(int[] deviceIds, int newGroupId) 112

bool ForceDevicesReboot(int[] deviceIds) 113

bool CancelDevicesReboot(int[] deviceIds) 113

bool ForceDevicesRecovery(int[] deviceIds, bool reformat) 114

bool CancelDevicesRecovery(int[] deviceIds) 114

bool ForceDevicesLogsUpload(int[] deviceIds) 115

bool CancelDevicesLogsUpload(int[] deviceIds) 115

PagedList<DeviceLogReport> GetDeviceLogReports(int deviceId, DeviceLogType logType, string marker, int pageSize) 116

bool DeleteDevices(int[] deviceIds) 117

INTRODUCTION

The BrightSign Network API exposes a large set of WebUI functionality using a standardized set of entities, methods, and properties. Developers can use this API to build new interfaces that have a different look and feel from, but provide a similar feature set to, the BrightSign Network WebUI.

This API Reference Manual first outlines the general working principles of interfacing with the BrightSign Network API. It then describes the properties and methods of each entity in detail.

Entities

Identifiers

Each entity has a unique ID consisting of an incremental integer value that functions as its Primary Key. This allows for simple identification and retrieval of a particular object instance. This value is usually not revealed to end users of the UI because it is only useful for internal client/server operations.

Some entities also have Alternate Keys (the `[string] Name` property, for example) because the system often requires the uniqueness of an entity within a BSN account. You can retrieve an entity using any of its keys, whether primary or alternate.

Dependency

Almost all entities have “parent” and “child” entities contained within a dependency tree: For example, a Dynamic Playlist entity includes Content entities on one hand, but is referenced by Presentation entities on the other. In this case, each Content entity is the “child” of the Dynamic Playlist entity, while each Presentation entity functions as the “parent” of the Dynamic Playlist entity.

If a particular object instance has one or more parent objects, then it is considered “in use”. In most cases, this status will affect the operations that can be performed with it.

Methods

The BSN Web API provides different methods to perform similar operations. It is possible, for example, to retrieve/update either all or a specified subgroup of properties. This allows you to choose which methods suit your deployment best in terms of usability, responsiveness, etc.

Entity Retrieval Methods

Currently, there are four types of entity retrieval method:

`Get{EntityName(s)}`

Methods of this type retrieve all entities of the corresponding type and return paged results with initialized parent dependencies and/or usage indicators. For performance reasons, they do not return child entities. The page size is limited to a certain number depending on the method.

The BrightSign Network Web API implements a markers-based approach similar to the Amazon REST 3 API: The client specifies the marker (entity ID) of the starting element in the list, as well as the number of entities to receive. The server will indicate whether the response is truncated (i.e. there are more elements that can be retrieved by the next request) and provide the marker for the next object instance.

Note: *If a client has retrieved a first page and is in the process of retrieving the second while another client adds, updates, or removes two objects on different pages, the first client may receive an inconsistent state.*

`GetSpecified{EntityName(s)}`

Methods of this type retrieve entities with specified keys. The results will not be paged; as a result, there is a limit on the number of entities that can be requested. This type of method will initialize only parent dependencies.

`Find{EntityName(s)}`

Methods of this type allow you to search for entities using a `[string] Name` pattern (including wildcards). This method type only exists for entities that have a Name alternate key. These methods return paged results with initialized parent dependencies and/or usage indicators.

`Get{EntityName} / Get{EntityName}ByName`

Methods of this type retrieve a single object instance using one of its Keys. Unlike other method types, these methods allow retrieval of child entities along with the requested entity. This behavior is useful for entity properties dialogues (e.g. a Group Properties dialogue that shows general information about a particular group) and entity management pages (e.g. a Dynamic Playlist management page that can be initialized with a single call).

Entity Update Methods

Currently, there are two types of entity update method:

> **General Update Methods**: e.g. `UpdateGroup`
> These methods receive a complete object in its updated state and attempt to detect and store all changes on the server side. These methods are useful in cases where a client retrieves an object first, allows the user to make changes, and then passes the new state to the server without the need to track user changes or construct call chains for applying a particular change (e.g. selecting the **Edit** button of a Dynamic Playlist in the BrightSign Network WebUI)—this is a use case that fits most WebUI and some BrightAuthor scenarios.
>
> These methods can also update object relations, so the client will be able to store not only all new property values, but also, for instance, updated sets of Dynamic Playlists or presentation contents in a single transactional service call without intermediate states.

Specific Update Methods: e.g. `UpdateGroupsAutorun`
These methods receive Entity IDs and one or more values for updating logically related properties within that entity (for example, changing the minimum required firmware values for a group). Methods of this type don't require the client to retrieve the object first. This may be useful for small dialogs, groups of controls (such as those in BrightAuthor), or internal client logic for sealing off certain parts of functionality from end users.

Object Permissions

Currently, all supported entities have a predetermined set of Object Permissions that cannot be changed. Customizable permissions will be implemented in the future.

SOAP Endpoints

Currently, the BrightSign Network Web API consists of two web services with nearly identical configurations: the Application Service and the Content Upload Service. Each web service has two SOAP endpoints: One is configured to use the WS-* specification and the other to use the WS-I Basic Profile.

WS-*

The WS-* endpoint is intended for rich, multifunctional server or desktop applications that support features such as sessions and transactions. The endpoint also supports its own rich security module and complex messaging format.

It would be difficult and time consuming to manually implement the required functionality for WS-*, including message serialization/de-serialization, encoding/decoding, connection management, and state management. Therefore, in order to utilize this endpoint, you will first need find a library that can create a service reference (i.e. proxy class) for your language and/or platform. Windows features built-in tools for creating service references for .NET applications, where the WS-* specification is the default web service endpoint configuration. Official or third-party tools/libraries have also been built for some, but not all, languages and platforms.

Once you find such a tool/library for your platform, you will be able to create a service reference for the web service and work with it using familiar methods in the code.

WS-I Basic

The WS-I Basic endpoint is defined for simple clients that support a small number of functions or don't support the WS-* specification. This specification does not support sessions or transactions. It does not have its own security module, using IIS instead. As a result, the message format is much simpler. WS-I Basic is also supported by all languages and platforms; it is even possible to use it directly without a service reference (i.e. proxy class) if needed (for example, through JavaScript).

SOAP Access Point URLs

Use the following URLs to create service references:

- https://api.brightsignnetwork.com/2014/04/SOAP/WSDL/
- https://api.brightsignnetwork.com/Uploads/2014/04/SOAP/WSDL/

The service-endpoints addresses are specified in the WSDLs. Most SOAP tools and libraries used to create references will find the addresses automatically. The current service endpoint URLs are as follows:

- http://api.brightsignnetwork.com/2014/04/SOAP/WS/
- https://api.brightsignnetwork.com/2014/04/SOAP/Basic/
- http://api.brightsignnetwork.com/Uploads/2014/04/SOAP/WS/
- https://api.brightsignnetwork.com/Uploads/2014/04/SOAP/Basic/

Note: *Each URL has a specified API version (e.g. "2014/04/"). API versions will rarely, if ever, be changed after production release, the exception being changes that will extend functionality but not affect existing clients. Older API version URLs may be removed at a later time if deemed obsolete.*

User Authentication

Credentials

Both BSN endpoints use simple username-password authentication; the username has the following format: `{AccountName/{UserLogin}`. These credentials should be passed in the header of all SOAP requests according to the corresponding SOAP specification. The complex security rules of the WS-* endpoints are defined in the WS-Security specification. The WS-I Basic Profile, on the other hand, defines just two special elements in the message header for the username and password.

Encoding

The WS-* endpoint is configured to use message-level encoding exposed over HTTP: Each request/response is encoded by the application first, then wrapped by a service message and sent in plaintext over the Internet.

The WS-I Basic endpoint, on the other hand, is configured to use transport encoding, with message credentials exposed over HTTPS: Each request/response is sent to the web server or client in plaintext; it is then encoded and transferred over the Internet via SSL/TLS.

Development Tools

The BSN WebUI contains a simple method that can be used for configuration checks and client login dialogs:

```
User Authenticate()
```

This method validates the passed credentials and returns the corresponding User instance. Note that User credentials should be specified in all other method calls.

CONTENT

Content Entity

The Content entity has the following properties:

- `[int] Id`:(Read only) The identifier and primary key of the Content entity.
- `[string] FileName`: The virtual name of the file represented by the current Content instance. This string may differ from the file name on the client device before being uploaded. This property can be set by both client and server, and is used in many capacities both within BSN and on devices.
- `[string] PhysicalPath`:(Read only) An external URL for the associated file contained in persistent storage.
- `[string] VirtualPath`: A virtual path to the associated file within the Library of the BSN account.
- `[ContentType] Type`:(Read only) The simplified content type of the associated file. This property is represented with a ContentType enumeration value.
- `[long] FileSize`:(Read only) The size of the associated file in bytes.
- `[string] FileHash`:(Read only) The SHA1 hash of the associated file.
- `[string] ThumbPath`:(Read only) An external URL for the file thumbnail image contained in persistent storage.
- `[DateTime] UploadDate`:(Read only) A UTC timestamp indicating when the associated file was uploaded to BSN.
- `[DynamicPlaylistInfo[]]DynamicPlaylists`: (Read only) An array of DynamicPlaylistInfo structures that denote parent Dynamic Playlists.
- `[PresentationInfo[]] Presentations`: (Read Only) An array of PresentationInfo structures that denote parent presentations.

ContentFolder Entity

The ContentFolder entity has the following properties:

- `[int] Id`:(Read only) The identifier and primary key of the ContentFolder entity.
- `[int] AccountId`:(Read only) The identifier of the account that owns the ContentFolder entity.

- `[string] Name`: The user-defined name of the folder represented by the ContentFolder instance. This name must be unique in the scope of the parent folder.
- `[string] VirtualPath`: A virtual path to the associated folder within the Library of the BSN account (for example, “\Shared\Incoming\” or “\Users\jdoe@example.com\Home\”).
- `[string] ThumbPath`:(Read only) An external URL for the file thumbnail image contained in persistent storage.
- `[DateTime] CreationDate`:(Read only) A UTC timestamp indicating when the associated folder was created in BSN.

ContentType Enumeration

The ContentType enumeration specifies the type of the file associated with the Content instance. It has the following properties.

- `Image`: A JPG, PNG, or BMP image file
- `Video`: An MPG, MP4, TS, MOV, VOB, or WMV video file
- `Audio`: An MP3 or WAV audio file
- `DynamicPlaylist`: A Dynamic Playlist file
- `Other`: An unknown file type

ContentTransition Enumeration

The ContentTransition enumeration represents the screen effect that is shown during the transition between the previous presentation state and the current content state. It has the following properties:

- `NoEffect`
- `WipeFromTop`
- `WipeFromBottom`
- `WipeFromLeft`
- `WipeFromRight`
- `ExplodeFromCenter`
- `ExplodeFromTopLeft`

- `ExplodeFromTopRight`
- `ExplodeFromBottomLeft`
- `ExplodeFromBottomRight`
- `VenetianBlindsVertical`
- `VenetianBlindsHorizontal`
- `CombVertical`
- `CombHorizontal`
- `FadeToBackground`
- `FadeToNewImage`
- `SlideTop`
- `SlideBottom`
- `SlideLeft`
- `SlideRight`

DynamicPlaylistContent Entity-Relation

The DynamicPlaylistContent entity-relation represents associations between Content and ImageVideoDynamicPlaylist or AudioDynamicPlaylist instances. It has the following properties:

- `[int] ContentId`: The identifier and primary key of the associated Content instance. This value can be set by both client and server.
- `[string] FileName`: The virtual name of the file represented by the current Content instance.
- `[TimeSpan] DisplayDuration`: The amount of time that the device displays an associated image. This property does not apply to audio or video files.

PresentationContent Entity-Relation

The PresentationContent entity-relation represents the association between Content and Presentation instances. It has the following properties:

- `[int] ContentId`: The identifier and primary key of the associated Content instance. This value can be set by both client and server.
- `[string] ContentName`: The name of the content represented by the PresentationContent entity-relation. When the entity-relation represents the relationship between a Presentation instance and a Content instance, this property contains the virtual name of the content file. Alternatively, when the entity-relation represents the relationship between a Presentation instance and a ImageVideoDynamicPlaylist/AudioDynamicPlaylist instance, this property contains the `[string] Name` of the Dynamic Playlist.
- `[string] StateName`: The name of the presentation state that corresponds to the associated Content instance.
- `[TimeSpan] DisplayDuration`: The amount of time that the device displays the associated image.
- `[ContentTransition] Transition`: A ContentTransition enumeration representing the screen effect that is shown during the transition between the previous presentation state and the current content state (i.e. the associated content state). This is a pre-defined value for the entity-relation.

Note: *The `[TimeSpan] DisplayDuration` and `[ContentTransition] Transition` properties only apply to image files. Content management methods will automatically set the former property to 0 and the latter property to "NoEffect" (without errors) if the associated PresentationContent instance does not represent an image file.*

- `[ContentType] ContentType`: A ContentType enumeration that allows a system to distinguish between a media file and a Dynamic Playlist.

DynamicPlaylistInfo Structure

The DynamicPlaylistInfo structure provides information about a parent ImageVideoDynamicPlaylist or AudioDynamicPlaylist instance. It has the following properties:

- `[int] Id`:(Read Only) The identifier and primary key of the instance
- `[string] Name`: The user-defined name of the instance. This string value is an alternate key that is unique in the scope of the BrightSign Network account.

PresentationInfo Structure

The PresentationInfo structure provides information about a parent Presentation instance. It has the following properties:

- `[int] Id`:(Read Only) The identifier and primary key of the Presentation instance
- `[string] Name`: The user-defined name of the Presentation instance. This string value must be unique in the scope of the BrightSign Network account.

Content Management Web Methods

- `PagedList<Content> GetAllContent(string marker, int pageSize)`
- `List<ContentFolder> GetContentFolders(string virtualPath)`
- `PagedList<Content> GetFolderContent(string virtualPath, string marker, int pageSize)`
- `List<Content> GetSpecifiedContent(int[] contentIds)`
- `PagedList<Content> FindContent(string fileNamePattern, string marker, int pageSize)`
- `Content GetContent(int contentId)`
- `ContentFolder CreateContentFolder(ContentFolder entity)`
- `bool MoveContent(int[] contentIds, string newVirtualPath)`
- `bool CheckContentUsage(int contentId)`
- `bool DeleteContent(int[] contentIds)`

Note: *Content update and creation operations can only be performed by the File Upload Service.*

`PagedList<Content> GetAllContent(string marker, int pageSize)`

Description

Retrieves the next page of the Content list, sorted alphabetically by `[string] FileName`. The returned list will contain no more items than the defined page size. This method only supports retrieval of image, audio, and video file types.

Required Permissions

Content: View Content

Parameters

- `[string] marker:` The `[string] FileName` of the last Content instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize:` The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`List<ContentFolder> GetContentFolders(string virtualPath)`

Description

Retrieves the first level of subfolders of the parent ContentFolder entity, which is specified using its virtual path. Paging is not implemented for this method because of the limit placed on the maximum number of subfolders within a parent folder.

Required Permissions

Content: View Content

Parameters

- `[string] virtualPath`: The virtual path of the parent ContentFolder entity containing the requested subfolders. The root virtual path is "\".

`PagedList<Content> GetFolderContent(string virtualPath, string marker, int pageSize)`

Description

Retrieves the first level of files within the parent ContentFolder entity, which is specified using its virtual path. The returned list is organized by File Name and may not contain more items than the defined page size. This method only supports retrieval of image, audio, and video file types.

Required Permissions

Content: View Content

Parameters

- `[string] virtualPath`: The virtual path of the parent ContentFolder entity. The root virtual path is "\".
- `[string] marker:` The `[string] FileName` of the last Content instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`List<Content> GetSpecifiedContent(int[] contentIds)`

Description

Retrieves a list of Content instances matching the specified identifiers. The results are sorted alphabetically by Content `[string] FileName`. The identifiers of nonexistent Content instances will be ignored.

Required Permissions

Content: View Content

Parameters

- `[Int[]] contentIds`: An array of `[int] Id` values for the Content instances being requested. The maximum number of items is limited to 100 by the server. Attempting to request more than the maximum allowed number of objects will cause an error, while passing an empty array will lead to an immediate empty response without an error.

`PagedList<Content> FindContent(string fileNamePattern, string marker, int pageSize)`

Description

Retrieves the next page of a Content list containing file names matched with the specified pattern. The returned list is organized by `[string] FileName` and may not contain more items than the defined page size. This method only supports retrieval of image, audio, and video file types.

Required Permissions

Content: View Content

Parameters

- `[string] fileNamePattern`: The exact `[string] FileName` of the Content instance (or its wildcard-based pattern). Supported wildcards currently include "*", "?", and "['and']".
- `[string] marker:` The `[string] FileName` of the last Content instance on the previous page.
- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`Content GetContent(int contentId)`

Description

Retrieves a single Content instance with the specified identifier. This method returns Null if there are no Content instances containing the specified identifier. This method only supports retrieval of image, audio, and video file types.

Required Permissions

Content: View Content

Parameters

- `[int] contentId`: The identifier and primary key of the associated Content instance.

`ContentFolder CreateContentFolder(ContentFolder entity)`

Description

Creates a new virtual folder within the BSN Library. The folder will have a specified name and location defined by its virtual path. This method returns the newly created object with all initialized properties (or Null if an error occurs during the creation process).

Required Permissions

Content: View Content

Parameters

- `[ContentFolder] entity`: A ContentFolder instance with an initialized `[string] Name`, `[string] VirtualPath`, and `[string] ThumbPath` (if needed). All other property values for the ContentFolder instance will be ignored. If this parameter is set to Null, then the method will immediately return Null without error. A descriptive error will be returned to the client if any of the following conditions occur:
  - The `[string] VirtualPath` of the initialized ContentFolder does not contain a leading slash("\").
  - The `[string] ThumbPath` does not begin with an absolute URI (i.e. "http://" or "https://").
  - The `[string] Name` of the initialized ContentFolder is more than 128 characters.
  - A ContentFolder instance within the same parent folder has the same name.

```
bool MoveContent(int[] contentIds, string newVirtualPath)
```

Description

Moves the specified Content and/or ContentFolder instances to the specified virtual folder. This method returns True upon success and False upon failure.

Required Permissions

Content: Update Content

Parameters

- `[int[]] contentIds`: An array of `[int] Id` values for the Content and/or ContentFolder instances that will be updated. The number of items is limited to 100 by the server. Attempting to request more than the maximum allowed number of objects will cause an error. If the client passes an empty array, or if none of the identifiers in the array match existing content instances, the server will immediately return an empty response without an error.
- `[string] newVirtualPath`: The new virtual path for the specified Content and/or ContentFolder instances (e.g. "\Shared\Incoming\").

`bool CheckContentUsage(int contentId)`

Description

Determines whether the specified Content instance is referenced by Presentations, Dynamic Playlists, Web Pages, or Device Web Pages. This method returns True if the specified Content instance has parent dependencies (i.e. it is in use).

Note that this method only supports retrieval of image, audio, and video file types. Also note that when using this method to check whether a Content instance can be deleted, a False ("not in use") status may change in the time between calling this method and calling `DeleteContent()`.

Required Permissions

Content: View Content

Parameters

- `[int] contentId`: The identifier and primary key of the associated Content instance. If there are no Content instances with the specified identifier, this method will return False without an error.

`bool DeleteContent(int[] contentIds)`

Description

Deletes the specified Content and ContentFolder instances, as well as any associated files, in both the database and persistent storage. This method returns True only if the operation was completely successful. None of the affected instances can be in use by any Presentation, Dynamic Playlist, Web Page, or Device Webpage. In these cases a descriptive error will be returned to the client.

Required Permissions

Content: Delete Content

Parameters

- `[int[]] contentIds`: An array of `[int] Id` values reflecting the Content instances to be deleted. The maximum number of items is limited to 100 by the server. Attempting to request more than the allowed number of objects will cause an error. A descriptive error will be returned if one or more Content `[int] Id` is invalid. If the client passes an empty array, or if none of the identifiers in the array match existing content instances, the server will immediately return an empty response without an error.

CONTENT UPLOAD

Overview

Content Upload methods use a separate SOAP endpoint from all other Web API methods. To upload a file, the client must first call `StartFileUpload()`, then call `AppendChunk()` one or more times, then call `CompleteFileUpload()`. It is also possible to upload multiple associated webpage asset files using a session token provided by the `StartWebPageUploadSession()` method. The workflow for uploading webpage assets is outlined in the following section.

The Content Upload service allows uploading files using multiple threads in parallel. To use multi-threading, the client must first call `StartFileUpload()` and obtain an upload token. The client then uploads chunks in multiple threads (using `AppendChunk()` with different parameters), specifying the `partNumber` and `offset` for each thread along with the same upload token. After the chunks are uploaded, the client must call `CompleteFileUpload()`. The client may also upload several different files while at the same time specifying different upload tokens.

Web Page Upload Work Flow

1. Call `StartWebPageUploadSession()` using parameters for the webpage file and associated content files. Retrieve the initialized session and upload tokens from the `WebPageUploadStatus` return.
2. Call `StartFileUpload()` using parameters for the webpage file, as well as the session token and primary upload token from the `WebPageUploadStatus` return. Make sure to specify the `[enum]ContentType` as “Webpage”.
3. Call `AppendChunk()` for the number of times needed to complete the webpage file upload. Use the session token and primary upload token from the `WebPageUploadStatus` return.
4. Call `CompleteFileUpload()` using the same content upload arguments utilized in **Step 2**.
5. Repeat the following steps for each webpage asset file:

a. Call `StartFileUpload()` using parameters for the asset file, as well as the session token and asset upload token from the `WebPageUploadStatus` return.
b. Call `AppendChunk()` for the number of times needed to complete the asset file upload. Use the session token and asset upload token from the `WebPageUploadStatus` return.
c. Call `CompleteFileUpload()` using the same content upload arguments utilized in **Step 5a**.

6. Call `CompleteWebPageUploadSession()` using the same content upload arguments utilized in **Step 1**.

Web Page Update Work Flow

Note: *Orphaned web assets are marked for deletion after 24 hours.*

1. Call `StartWebPageUploadSession()` using an existing `[string]WebPageId` parameter and arguments for the new webpage file. Retrieve the initialized session and upload tokens from the `WebPageUploadStatus` return.
2. Call `StartFileUpload()` using parameters for the new webpage file, as well as the session and upload tokens from the `WebPageUploadStatus` return. You will also need to specify the `[string]ContentId` (which is the same as the `[string]WebPageId`) of the webpage file to update. Ensure the `[enum]ContentType` is specified as “Webpage”.
3. Call `AppendChunk()` for the number of times needed to complete the webpage file upload. Use the session token and primary upload token from the `WebPageUploadStatus` return.

Note: *When updating a webpage, the main HTML file must always be updated (even if changes are only being made to asset files).*

4. Call `CompleteFileUpload()` using the same content upload arguments utilized in **Step 2**.
5. Repeat the following steps for each webpage asset file:
    a. Call `StartFileUpload()` using parameters for the asset file, as well as the session token and asset upload token from the `WebPageUploadStatus` return. The `ContentUploadStatus` return will contain an optional Content Negotiation response.
    a. If the Content Negotiation status matches the current asset, call `CancelFileUpload()` using the session and asset upload tokens.
    b. If the Content Negotiation status does not match the current asset:

i. Call `AppendChunk()` for the number of times needed to complete the asset file upload. Use the session token and asset upload token from the `WebPageUploadStatus` return.
ii. Call `CompleteFileUpload()` using the same content upload arguments utilized in **Step 5a**.

Content Upload Web Methods

- `ContentUploadStatus StartFileUpload(string uploadToken, string sessionToken, string filename, long filesize, int chunksCount, enum ContentType, string virtualPath, base64Binary fileThumb, string sha1Hash)`
- `ContentUploadStatus AppendChunk(string uploadToken, int partNumber, binary data, long offset)`
- `ContentUploadStatus CompleteFileUpload(string uploadToken, string sessionToken, string filename, long filesize, int chunksCount, enum ContentType, string virtualPath, base64Binary fileThumb, string sha1Hash)`
- `WebPageUploadStatus StartWebPageUploadSession(array webpageAssets[], string uploadToken, string sessionToken, string filename, long filesize, int chunksCount, enum ContentType, string virtualPath, base64Binary fileThumb, string sha1Hash)`
- `WebPageUploadStatus CompleteWebPageUploadSession(string sessionToken)`
- `ContentUploadStatus CancelFileUpload(string uploadToken)`
- `ContentUploadStatus GetFileUploadStatus(string uploadToken)`
- `WebPageUploadStatus GetWebPageUploadStatus(string sessionToken)`

`ContentUploadStatus StartFileUpload(string uploadToken, string sessionToken, string filename, long filesize, int chunksCount, enum ContentType, string virtualPath, base64Binary fileThumb, string sha1Hash)`

Description

Initializes the content upload process by returning an upload token, which is a descriptor value that is utilized by the `AppendChunk()` and `CompleteFileUpload()` methods. This method may also receive an upload token if it is being used to upload a webpage file.

Required Permissions

Content: Upload Content

Parameters

- `[string] uploadToken`: The token of an upload that was initialized by the `StartWebPageUploadSession()` method. This parameter is only specified if the file being uploaded is a webpage file.
- `[string] sessionToken`: The token of an upload session initialized by the `StartWebPageUploadSession()` method. This parameter enables uploading of files related to a webpage file; it should only be specified if a webpage file or associated asset file is being uploaded.
- `[string] filename`: The name of the file to be uploaded. A descriptive error will be returned if any of the following conditions occur:
    - This string is empty.
    - The string has more than 128 characters.
    - The file name is not valid.
- `[long] filesize`: The size of the file in bytes. This parameter must be an unsigned integer value no greater than 5000000000 (i.e. 5GB). A descriptive error will be returned if the value is negative.
- `[int] chunksCount`: The number of chunks the file consists of (starting at 1). The optimal chunk size is 256KiB. Files that are larger than this chunk size must be split into smaller parts by the client. Multiple chunks should be uploaded using multiple `AppendChunk()` calls specifying the same upload token.

- `[enum: Auto, Image, Video, Audio, Webpage] contentType`: The type of the content file. The default type is “Image”.
- `[string] virtualPath`: The path in the BSN Library to which the file should be uploaded (the default is “\Shared\Incoming\”). If the path does not exist, it will be created. It must also contain leading and trailing slashes and be no longer than 128 characters.
- `[base64Binary] fileThumb`: The optional thumbnail data as a binary attachment. This parameter will be generated by the server if it is not specified.
- `[string] sha1Hash`: The SHA1 hash of the current file. The server performs verification of the uploaded file using the specified hash. A descriptive error will be returned if the specified hash does not match the hash value calculated by the server. This parameter is optional; the server-side hash verification process is skipped if this parameter is not specified.

`ContentUploadStatus AppendChunk(string uploadToken, int partNumber, binary data, long offset)`

Description

Appends chunks onto files that are larger than 256Kib. It returns True if the upload is successful. Otherwise, it returns False.

Required Permissions

Content: Upload Content, Update Content

> **Note**: *The Update Content permission is only required if the `CompleteFileUpload()` is called with a `[string]contentId` value that corresponds to an existing content entity (i.e. the client is attempting to overwrite a content entity).*

Parameters

- `[string] uploadToken`: The token of the upload that was initialized by the `StartFileUpload()` or `StartWebPageUpload()` method. A descriptive error is returned if the string value is empty or not specified.

- `[string] sessionToken`: The token of the upload session initialized by the `StartWebPageUpload()` method. This token should only be provided if the client is uploading a set of webpage assets.
- `[int] partNumber`: The number of the part, starting from 0.
- `[base64Binary] data`: The binary data of the file chunk.
- `[long] offset`: The offset of the current file chunk. The first chunk has an offset of 0. A descriptive error is returned if the offset value is not positive.

```
ContentUploadStatus CompleteFileUpload(string uploadToken, string sessionToken, string filename, long filesize, int chunksCount, enum ContentType, string virtualPath, base64Binary fileThumb, string sha1Hash)
```

Description

Returns the Content ID of the uploaded file if file upload is successful. Otherwise, it returns a descriptive error. If a specified parameter is different between the `StartFileUpload()` and `CompleteFileUpload()` calls on a file, the specification in the `CompleteFileUpload()` call will overwrite the original `StartFileUpload()` specification.

Required Permissions

Content: Upload Content

Parameters

- `[string] uploadToken`: The token of the upload that was initialized by the `StartFileUpload()` method. A descriptive error is returned if the string value is empty or not specified.
- `[string] sessionToken`: The token of the upload session initialized by the `StartWebPageUpload()` method. This token should only be provided if the client is uploading a set of webpage assets.
- `[string] filename`: The name of the uploaded file. A descriptive error will be returned if any of the following conditions occur:
    - This string is empty.
    - The string has more than 128 characters.
    - The file name is not valid.

- `[long] filesize`: The size of the video file in bytes. This parameter must be an unsigned integer value no greater than 5000000000 (i.e. 5GB). A descriptive error will be returned if the value is negative.
- `[int] chunksCount`: The number of chunks the file consists of (starting at 1). The optimal chunk size is 256KiB. Files that are larger than this chunk size must be split into smaller parts by the client. Multiple chunks should be uploaded using multiple `AppendChunk()` calls specifying the same upload token.
- `[string] sha1Hash`: The SHA1 hash of the current file. The server performs verification of the uploaded file using the specified hash. A descriptive error will be returned if the specified hash does not match the hash value calculated by the server. This parameter is optional; the server-side hash verification process is skipped if this parameter is not specified.
- `[enum: Auto, Image, Video, Audio, Webpage] contentType`: The type of the content file. The default type is Image.
- `[string] virtualPath`: The path in the BSN Library to which the file should be uploaded (the default is "\Shared\Incoming\"). If the path does not exist, it will be created. It must also contain leading and trailing slashes and be no longer than 128 characters.
- `[base64Binary] fileThumb`: The optional thumbnail data as a binary attachment. This parameter will be generated by the server if it is not specified.

```
WebPageUploadStatus StartWebPageUploadSession(array webpageAssets[], string uploadToken,
string sessionToken, string filename, long filesize, int chunksCount, enum ContentType,
string virtualPath, base64Binary fileThumb, string sha1Hash)
```

Description

Initializes a session for uploading a set of webpage asset files. This method accepts a set of parameters for the HTML file similar to `StartFileUpload()`. It also accepts an array of parameters for each asset file associated with the webpage. If successful, this method will return a session token that will be utilized in each individual set of `StartFileUpload()`/`CompleteFileUpload()` transactions.

Required Permissions

Content: Upload Content

Parameters

- `[array] webpageAssets`: An array of one or more sets of `WebPageAssetUploadArguments`, which can contain the following:
    - `[string] filename`: The name of the file to be uploaded. A descriptive error will be returned if any of the following conditions occur:
        - This string is empty.
        - The string has more than 128 characters.
        - The file name is not valid.
    - `[long] filesize`: The size of the file in bytes. This parameter must be an unsigned integer value no greater than 5000000000 (i.e. 5GB). A descriptive error will be returned if the value is negative.
    - `[int] chunksCount`: The number of chunks the file consists of (starting at 1). The optimal chunk size is 256KiB. Files that are larger than this chunk size must be split into smaller parts by the client. Multiple chunks should be uploaded using multiple `AppendChunk()` calls specifying the same upload token.
    - `[enum: Auto, Image, Video, Audio, Webpage] contentType`: The type of the asset file. The default type is Image.
    - `[string] virtualPath`: The path in the BSN Library to which the file should be uploaded. If the path does not exist, it will be created. It must also contain leading and trailing slashes and be no longer than 128 characters.
    - `[string] RelativePath`: The relative path of the associated asset file in relation to the webpage HTML file.
    - `[string] sha1Hash`: The SHA1 hash of the current file. The server performs verification of the uploaded file using the specified hash. A descriptive error will be returned if the specified hash does not match the hash value calculated by the server. This parameter is optional; the server-side hash verification process is skipped if this parameter is not specified.
    - `[string] webpageAssetId`
- `[string] filename`: The name of the webpage file to be uploaded. A descriptive error will be returned if any of the following conditions occur:
    - This string is empty.

    - The string has more than 128 characters.
    - The file name is not valid.
- `[long] filesize`: The size of the webpage in bytes. This parameter must be an unsigned integer value no greater than 5000000000 (i.e. 5GB). A descriptive error will be returned if the value is negative.
- `[int] chunksCount`: The number of chunks the file consists of (starting at 1). The optimal chunk size is 256KiB. Files that are larger than this chunk size must be split into smaller parts by the client. Multiple chunks should be uploaded using multiple `AppendChunk()` calls specifying the same upload token.
- `[enum: Auto, Image, Video, Audio, Webpage] contentType`: The type of the content file, which should be specified as "Webpage".
- `[string] virtualPath`: The path in the BSN Library to which the file should be uploaded (the default is "\Shared\Incoming\"). If the path does not exist, it will be created. It must also contain leading and trailing slashes and be no longer than 128 characters.
- `[base64Binary] fileThumb`: The optional thumbnail data as a binary attachment. This parameter will be generated by the server if it is not specified.
- `[string] sha1Hash`: The SHA1 hash of the webpage file. The server performs verification of the uploaded file using the specified hash. A descriptive error will be returned if the specified hash does not match the hash value calculated by the server. This parameter is optional; the server-side hash verification process is skipped if this parameter is not specified.

`WebPageUploadStatus CompleteWebPageUploadSession(string sessionToken)`

Description

Returns the Web Page ID of the uploaded webpage file if the upload session is successful. Otherwise, it returns a descriptive error. If a specified parameter is different between the `StartWebPageUploadSession()` and `CompleteWebPageUploadSession()`, the specification in the `CompleteWebPageUploadSession()` call will overwrite the original `StartWebPageUploadSession()` specification. This is true for both the webpage file and associated asset files.

Required Permissions

Content: Upload Content

Parameters

- `[array] webpageAssets`: An array of one or more sets of `WebPageAssetUploadArguments`, which can contain the following:
    - `[string] filename`: The name of the file to be uploaded. A descriptive error will be returned if any of the following conditions occur:
        - This string is empty.
        - The string has more than 128 characters.
        - The file name is not valid.
    - `[long] filesize`: The size of the file in bytes. This parameter must be an unsigned integer value no greater than 5000000000 (i.e. 5GB). A descriptive error will be returned if the value is negative.
    - `[int] chunksCount`: The number of chunks the file consists of (starting at 1). The optimal chunk size is 256KiB. Files that are larger than this chunk size must be split into smaller parts by the client. Multiple chunks should be uploaded using multiple `AppendChunk()` calls specifying the same upload token.
    - `[enum: Auto, Image, Video, Audio, Webpage] contentType`: The type of the asset file. The default type is Image.
    - `[string] virtualPath`: The path in the BSN Library to which the file should be uploaded. If the path does not exist, it will be created. It must also contain leading and trailing slashes and be no longer than 128 characters.
    - `[string] RelativePath`: The relative path of the associated asset file in relation to the webpage HTML file.
    - `[string] sha1Hash`: The SHA1 hash of the current file. The server performs verification of the uploaded file using the specified hash. A descriptive error will be returned if the specified hash does not match the hash value calculated by the server. This parameter is optional; the server-side hash verification process is skipped if this parameter is not specified.

    - `[string] webpageAssetId`: The `[string]ContentId` of the webpage asset file. The client should specify this value only when an asset file is being updated.
- `[string] filename`: The name of the uploaded webpage. A descriptive error will be returned if any of the following conditions occur:
    - This string is empty.
    - The string has more than 128 characters.
    - The file name is not valid.
- `[long] filesize`: The size of the webpage in bytes. This parameter must be an unsigned integer value no greater than 5000000000 (i.e. 5GB). A descriptive error will be returned if the value is negative.
- `[int] chunksCount`: The number of chunks the file consists of (starting at 1). The optimal chunk size is 256KiB. Files that are larger than this chunk size must be split into smaller parts by the client. Multiple chunks should be uploaded using multiple `AppendChunk()` calls specifying the same upload token.
- `[enum: Auto, Image, Video, Audio, Webpage] contentType`: The type of the content file, which should be specified as “Webpage”.
- `[string] virtualPath`: The path in the BSN Library to which the file should be uploaded (the default is “\Shared\Incoming\”). If the path does not exist, it will be created. It must also contain leading and trailing slashes and be no longer than 128 characters.
- `[base64Binary] fileThumb`: The optional thumbnail data as a binary attachment. This parameter will be generated by the server if it is not specified.
- `[string] sha1Hash`: The SHA1 hash of the webpage file. The server performs verification of the uploaded file using the specified hash. A descriptive error will be returned if the specified hash does not match the hash value calculated by the server. This parameter is optional; the server-side hash verification process is skipped if this parameter is not specified.

`ContentUploadStatus CancelFileUpload(string uploadToken)`

Description

Cancels the specified content upload and deletes all uploaded file chunks. This method returns status information about the cancelled content upload.

Required Permissions

None

Parameters

- `[string] uploadToken`: The token of the content upload that should be cancelled. If the content upload with the specified token does not exist, the method will immediately return Null without error.

`ContentUploadStatus GetFileUploadStatus(string uploadToken)`

Description

Returns general information and upload status information about the content file associated with the specified upload token.

Required Permissions

None

Parameters

`[string] sessionToken`: The token of the upload session initialized by the `StartFileUpload()` method. If the specified session token does not correspond to an existing session, the method will return Null without error.

`WebPageUploadStatus GetWebPageUploadStatus(string sessionToken)`

Description

Returns general information and upload status information about all webpage asset files associated with the specified session token.

Required Permissions

None

Parameters

- `[string] sessionToken`: The token of the upload session initialized by the `StartWebPageUpload()` method. If the specified upload token does not correspond to an existing upload, the method will return Null without error.

DYNAMIC PLAYLIST

ImageVideoDynamicPlaylist Entity

The ImageVideoDynamicPlaylist entity represents a Dynamic Playlist containing image or video files only. It has the following properties.

- `[int] Id`:(Read Only) The identifier and primary key of the ImageVideoDynamicPlaylist instance.
- `[string] Name`: The user-defined name of the ImageVideoDynamicPlaylist instance. This string value is an alternate key that is unique in the scope of the BrightSign Network account.
- `[string] PhysicalPath`:(Read Only) An external URL of the Dynamic Playlist MRSS file in persistent storage.
- `[long] FileSize`:(Read Only) The size of the associated MRSS file in bytes.
- `[string] FileHash`:(Read Only) The SHA1 hash of the associated MRSS file contents.
- `[DateTime] CreationDate`:(Read Only) The UTC timestamp indicating when the current ImageVideoDynamicPlaylist instance was created in the BrightSign Network.
- `[List<DynamicPlaylistContent>] Content`: A list of DynamicPlaylistContent entity-relations that reference content instances contained within the Dynamic Playlist. This list is set to Null when not initialized by the server.
- `[PresentationInfo[]] Presentations`:(Read only) An array of PresentationInfo structures that denote parent presentations.

AudioDynamicPlaylist Entity

The AudioDynamicPlaylist entity represents a Dynamic Playlist containing audio files only. It has the following properties:
Note: *The BrightSign Network device autorun version (6.7.37) does not currently support publishing Audio Dynamic Playlists. As a result, you can create and manage Audio Dynamic Playlists, but you can't use them in presentations. This will be corrected in the next BrightSign Network update.*

- `[int] Id`:(Read Only) The identifier and primary key of the AudioDynamicPlaylist instance.
- `[string] Name`: The user-defined name of the AudioDynamicPlaylist instance. This string value is an alternate key that is unique in the scope of the BrightSign Network account.

- `[string] PhysicalPath`:(Read Only) An external URL of the Dynamic Playlist MRSS file in persistent storage.
- `[long] FileSize`:(Read Only) The size of the associated MRSS file in bytes.
- `[string] FileHash`:(Read Only) The SHA1 hash of the associated MRSS file contents.
- `[DateTime] CreationDate`:(Read Only) The UTC timestamp indicating when the current AudioDynamicPlaylist instance was created in the BrightSign Network.
- `[List<DynamicPlaylistContent>] Content`: A list of DynamicPlaylistContent entity-relations that reference content instances contained within the Dynamic Playlist. This list is set to Null when not initialized by the server.
- `[PresentationInfo[]] Presentations`:(Read only) An array of PresentationInfo structures that denote parent presentations.

Dynamic Playlist Management Web Methods

- `PagedList<DynamicPlaylist> GetDynamicPlaylists(string marker, int pageSize)`
- `PagedList<ImageVideoDynamicPlaylist> GetImageVideoDynamicPlaylists(string marker, int pageSize)`
- `PagedList<AudioDynamicPlaylist> GetAudioDynamicPlaylists(string marker, int pageSize)`
- `List<DynamicPlaylist> GetSpecifiedDynamicPlaylists(int[] dynamicPlaylistIds)`
- `PagedList<DynamicPlaylist> FindDynamicPlaylists(string namePattern, string marker, int pageSize)`
- `PagedList<ImageVideoDynamicPlaylist> FindImageVideoDynamicPlaylists(string namePattern, string marker, int pageSize)`
- `PagedList<AudioDynamicPlaylist> FindAudioDynamicPlaylists(string namePattern, string marker, int pageSize)`
- `DynamicPlaylist GetDynamicPlaylist(int dynamicPlaylistId, bool loadContent)`
- `DynamicPlaylist GetDynamicPlaylistByName(string name, bool loadContent)`
- `bool CheckDynamicPlaylistName(string name)`
- `bool CheckDynamicPlaylistUsage(int dynamicPlaylistId)`

- `bool CheckDynamicPlaylistUsageByName(string name)`
- `DynamicPlaylist CreateDynamicPlaylist(DynamicPlaylist entity)`
- `bool UpdateDynamicPlaylist(DynamicPlaylist entity)`
- `bool RenameDynamicPlaylist(int dynamicPlaylistId, string newName)`
- `bool DeleteDynamicPlaylists(int[] dynamicPlaylistIds)`

`PagedList<DynamicPlaylist> GetDynamicPlaylists(string marker, int pageSize)`

Description

Retrieves the next page of the Dynamic Playlist list, sorted alphabetically by `[string] Name`. AudioDynamicPlaylist and/or ImageVideoDynamicPlaylist instances are sorted alphabetically by `[string] Name`. The returned list will contain no more items than the defined page size.

Required Permissions

Dynamic Playlist: View Dynamic Playlists

Parameters

- `[string] marker`: The `[string] Name` of the last instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`PagedList<ImageVideoDynamicPlaylist> GetImageVideoDynamicPlaylists(string marker, int pageSize)`

Description

Retrieves the next page of the Dynamic Playlist list, sorted alphabetically by `[string] Name`.

The ImageVideoDynamicPlaylist instances are sorted alphabetically by `[string] Name`. The returned list will contain no more items than the defined page size.

Required Permissions

Dynamic Playlist: View Dynamic Playlists

Parameters

- `[string] marker`: The `[string] Name` of the last ImageVideoDynamicPlaylist instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`PagedList<AudioDynamicPlaylist> GetAudioDynamicPlaylists(string marker, int pageSize)`

Description

Retrieves the next page of the Dynamic Playlist list, sorted alphabetically by `[string] Name`. AudioDynamicPlaylist instances are sorted alphabetically by `[string] Name`. The returned list will contain no more items than the defined page size.

Required Permissions

Dynamic Playlist: View Dynamic Playlists

Parameters

- `[string] marker`: The `[string] Name` of the last AudioDynamicPlaylist instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`List<DynamicPlaylist> GetSpecifiedDynamicPlaylists(int[] dynamicPlaylistIds)`

Description

Retrieves a list of ImageVideoDynamicPlaylist and/or AudioDynamicPlaylist instances matching the specified identifiers. Instances are sorted alphabetically by `[string] Name`. The identifiers of nonexistent instances will be ignored.

Required Permissions

Dynamic Playlist: View Dynamic Playlists

Parameters

- `[int[]] dynamicPlaylistIds`: An array of `[int] Id` values for the instances being requested. The maximum number of items is limited to 100 by the server. Attempting to request more than the maximum allowed number of objects will cause an error, while passing an empty array will lead to an immediate empty response without an error.

`PagedList<DynamicPlaylist> FindDynamicPlaylists(string namePattern, string marker, int pageSize)`

Description

Retrieves the next page of a Dynamic Playlist list containing names matched with the specified pattern. The returned list is organized alphabetically by `[string] Name` and may not contain more items than the defined page size. This

method retrieves both ImageVideoDynamicPlaylist and AudioDynamicPlaylist instances; these instances can be differentiated by data type.

Required Permissions

Dynamic Playlists: View Dynamic Playlists

Parameters

- `[string] namePattern`: The exact `[string] Name` of the ImageVideoDynamicPlaylist or AudioDynamicPlaylist instance (or its wildcard-based pattern). Supported wildcards currently include "*", "?", and "[‘and’]".
- `[string] marker`: The `[string] Name` of the last instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize`: `[int] pageSize:` The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

```
PagedList<ImageVideoDynamicPlaylist> FindImageVideoDynamicPlaylists(string namePattern,
string marker, int pageSize)
```

Description

Retrieves the next page of an ImageVideoDynamicPlaylist instance list containing names matched with the specified pattern. The returned list is organized alphabetically by `[string] Name` and may not contain more items than the defined page size.

Required Permissions

Dynamic Playlists: View Dynamic Playlists

Parameters

- `[string] namePattern`: The exact `[string] Name` of the ImageVideoDynamicPlaylist instance (or its wildcard-based pattern). Supported wildcards currently include “*”, “?”, and “[‘and’]”.
- `[string] marker`: The `[string] Name` of the last instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize`: `[int] pageSize:` The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`PagedList<AudioDynamicPlaylist> FindAudioDynamicPlaylists(string namePattern, string marker, int pageSize)`

Description

Retrieves the next page of an AudioDynamicPlaylist instance list containing names matched with the specified pattern. The returned list is organized alphabetically by `[string] Name` and may not contain more items than the defined page size.

Required Permissions

Dynamic Playlists: View Dynamic Playlists

Parameters

- `[string] namePattern`: The exact `[string] Name` of the AudioDynamicPlaylist instance (or its wildcard-based pattern). Supported wildcards currently include “*”, “?”, and “[‘and’]”.
- `[string] marker`: The `[string] Name` of the last instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize`: `[int] pageSize:` The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is

truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`DynamicPlaylist GetDynamicPlaylist(int dynamicPlaylistId, bool loadContent)`

Description

Retrieves a single ImageVideoDynamicPlaylist or AudioDynamicPlaylist instance with the specified identifier. This method returns Null if the instance with the specified identifier does not exist.

Required Permissions

Dynamic Playlist: View Dynamic Playlists, View Content – Content: View Content

Parameters

- `[int] dynamicPlaylistId`: The identifier and primary key of the requested instance.
- `[bool] loadContent`: A flag that specifies whether the method should also initialize and return a list of all Content instances in use by the requested Dynamic Playlist.

Note: *If the Dynamic Playlist was created using a legacy version of BrightAuthor, the* `[int] DisplayDuration` *of image files may be returned set to 0.*

`DynamicPlaylist GetDynamicPlaylistByName(string name, bool loadContent)`

Description

Retrieves a single ImageVideoDynamicPlaylist or AudioDynamicPlaylist instance with the specified name. This method returns Null if the instance with the specified name does not exist.

Required Permissions

Dynamic Playlist: View Dynamic Playlists, View Content – Content: View Content

Parameters

- `[string] name`: The user-defined name (and alternate key) of the instance.

- `[bool] loadContent`: A flag that specifies whether the method should also initialize and return a list of all Content instances in use by the requested Dynamic Playlist.

Note: *If the Dynamic Playlist was created using a legacy version of BrightAuthor, the `[int] DisplayDuration` of image files may be returned set to 0.*

`bool CheckDynamicPlaylistName(string name)`

Description

Determines if the specified Dynamic Playlist name is in use within the BrightSign Network account. If there is a Dynamic Playlist with the specified name, this method will return True. Otherwise, it will return False.

Note that when using this method to check whether a Dynamic Playlist instance can be created, a False status may change in the time between calling this method and calling `CreateDynamicPlaylist()`.

Required Permissions

Dynamic Playlist: View Dynamic Playlists

Parameters

- `[string] name`: The Dynamic Playlist name value to be evaluated.

`bool CheckDynamicPlaylistUsage(int dynamicPlaylistId)`

Description

Determines if the ImageVideoDynamicPlaylist or AudioDynamicPlaylist instance (specified with its primary key) is referenced by one or more presentations. If the specified instance has parent dependencies (i.e. it is in use), this method will return True. Otherwise, it will return False.

Note that when using this method to check whether a instance can be deleted, a False (“not in use”) status may change in the time between calling this method and calling `DeleteDynamicPlaylist()`.

Required Permissions

Dynamic Playlist: View Dynamic Playlists

Parameters

- `[int] dynamicPlaylistId`: The identifier and primary key of the ImageVideoDynamicPlaylist or AudioDynamicPlaylist instance to be evaluated. If an instance with the specified identifier does not exist, the method will return False without error.

`bool CheckDynamicPlaylistUsageByName(string name)`

Description

Determines if the ImageVideoDynamicPlaylist or AudioDynamicPlaylist instance (specified with its alternate, user-defined key) is referenced by one or more presentations. If the specified instance has parent dependencies (i.e. it is in use), this method will return True. Otherwise, it will return False.

Note that when using this method to check whether an instance can be deleted, a False ("not in use") status may change in the time between calling this method and calling `DeleteDynamicPlaylist()`.

Required Permissions

Dynamic Playlist: View Dynamic Playlists

Parameters

- `[string] name`: The user-defined name (i.e. the alternate key) of the Dynamic Playlist to be evaluated. If an instance with the specified identifier does not exist, the method will return False without error.

`DynamicPlaylist CreateDynamicPlaylist(DynamicPlaylist entity)`

Description

Creates a new ImageVideoDynamicPlaylist or AudioDynamicPlaylist instance and a related MRSS file with the specified name and Content entities. This method returns the newly created object with all initialized properties if successful. If object creation is unsuccessful, this method will return a Null value.

Required Permissions

Dynamic Playlist: Create Dynamic Playlist – Content: Assign Content

Parameters

- `[DynamicPlaylist] entity`: An ImageVideoDynamicPlaylist or AudioDynamicPlaylist object instance with initialized `[string] Name` and `[List<DynamicPlaylistContent>] Contents` properties. All other properties will be ignored during object creation. If this parameter is set to Null, then the method will immediately return Null without error. A descriptive error will be returned to the client if any of the following conditions occur:
    - The name of the Dynamic Playlist is more than 50 characters.
    - The content files do not match the type of Dynamic Playlist instance that has been passed to the method (e.g. an AudioDynamicPlaylist contains image or video files, or vice versa).
    - Another ImageVideoDynamicPlaylist or AudioDynamicPlaylist instance has the same name.
    - The `[int] DisplayDuration` of a DynamicPlaylistContent entity-relation is set to less than 1 second. This only applies to entity-relations representing image files.

`bool UpdateDynamicPlaylist(DynamicPlaylist entity)`

Description

Updates the `[string] Name` and `[List<DynamicPlaylistContent>] Contents` properties of the specified ImageVideoDynamicPlaylist or AudioDynamicPlaylist instance and its related MRSS file. This method returns True upon success and False upon failure.

Required Permissions

Dynamic Playlist: Update Dynamic Playlist – Content: Assign Content, Unassign Content

Parameters

- `[DynamicPlaylist] entity`: An ImageVideoDynamicPlaylist or AudioDynamicPlaylist object instance with initialized `[string] Name` and `[List<DynamicPlaylistContent>] Contents` properties. All other properties will be ignored. If the instance is set to Null, then the method will immediately return Null without error. A descriptive error will be returned to the client if any of the following conditions occur:

- The passed string is more than 50 characters.
- The passed string does not match a preexisting Dynamic Playlist.
- The `[int] DisplayDuration` of a DynamicPlaylistContent entity-relation is set to less than 1 second. This only applies to entity-relations representing image files.

`bool RenameDynamicPlaylist(int dynamicPlaylistId, string newName)`

Description

Updates the `[string] Name` of the specified ImageVideoDynamicPlaylist or AudioDynamicPlaylist instance. This method returns True upon success and False upon failure.

Required Permissions

Dynamic Playlist: Update Dynamic Playlist, Rename Dynamic Playlist

Parameters

- `[int] dynamicPlaylistId`: The identifier and primary key of the instance. If an instance with the specified identifier does not exist, the method will return False without error.
- `[string] newName`: The new name for the specified instance. A descriptive error will be returned to the client if the passed string is more than 50 characters.

`bool DeleteDynamicPlaylists(int[] dynamicPlaylistIds)`

Description

Deletes the specified ImageVideoDynamicPlaylist and/or AudioDynamicPlaylist instances from both the database and persistent storage. Related MRSS files are also removed. This method returns True upon success and False upon failure.

Required Permissions

Dynamic Playlist: Delete Dynamic Playlist

Parameters

- `[int[]] dynamicPlaylistIds`: An array of `[int] Id` values for the instances to be deleted. The number of items that can be deleted is limited to 100 by the server. Requesting more than the maximum allowed number of objects will cause an error. An error will also be returned if one or more specified `[int] Id` value refers to a non-existent ImageVideoDynamicPlaylist or AudioDynamicPlaylist instance. Passing an empty array or Dynamic Playlist identifiers that don’t exist will lead to an immediate empty response without an error.

LIVE TEXT FEED

LiveTextFeed Entity

The LiveTextFeed entity has the following properties:

- `[int] Id`:(Read Only) The identifier and primary key of the LiveTextFeed instance.
- `[string] Name`: The user-defined name of the LiveTextFeed instance. This alternate key must be unique in the scope of the BSN account.
- `[string] PhysicalPath`:(Read Only) An external URL of the associated RSS file in persistent storage.
- `[long] FileSize`:(Read Only) The size of the associated RSS file in bytes.
- `[string] FileHash`:(Read Only): A SHA1 hash of the contents in the associated RSS file.
- `[DateTime] CreationDate`:(Read Only) A UTC timestamp indicating when the Live Text Feed was created in the BrightSign Network.
- `[Dictionary<string,string>] Contents`: A list of key-value pairs that make up the Live Text Feed. This list is set to Null when not initialized by the server.
- `[PresentationInfo[]] Presentations`:(Read Only) An array of PresentationInfo structures that denote parent presentations.

Live Text Feed Management Web Methods

- `PagedList<LiveTextFeed> GetLiveTextFeeds(string marker, int pageSize)`
- `List<LiveTextFeed> GetSpecifiedLiveTextFeeds(int[] liveTextFeedIds)`
- `PagedList<LiveTextFeed> FindLiveTextFeeds(string namePattern, string marker, int pageSize)`
- `LiveTextFeed GetLiveTextFeed(int liveTextFeedId, bool loadContent)`
- `LiveTextFeed GetLiveTextFeedByName(string name, bool loadContent)`
- `bool CheckLiveTextFeedName(string name)`
- `bool CheckLiveTextFeedUsage(int liveTextFeedId)`

- bool CheckLiveTextFeedUsageByName(string name)
- LiveTextFeed CreateLiveTextFeed(LiveTextFeed entity)
- LiveTextFeed CloneLiveTextFeed(string feedUrl)
- bool UpdateLiveTextFeed(LiveTextFeed entity)
- bool RenameLiveTextFeed(int liveTextFeedId, string newName)
- bool DeleteLiveTextFeeds(int[] liveTextFeedIds)

`PagedList<LiveTextFeed> GetLiveTextFeeds(string marker, int pageSize)`

Description

Retrieves the next page of the LiveTextFeed list, sorted by `[string] Name`. The returned list will contain no more items than the defined page size.

Required Permissions

Live Text Feed: View Live Text Feeds

Parameters

- `[string] marker`: The `[string] Name` of the last LiveTextFeed instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`List<LiveTextFeed> GetSpecifiedLiveTextFeeds(int[] liveTextFeedIds)`

Description

Retrieves a list of LiveTextFeed instances matching the specified identifiers, sorted by `[string] Name`. The identifiers of nonexistent LiveTextFeed instances will be ignored.

Required Permissions

Live Text Feed: View Live Text Feeds

Parameters

- `[int[]] liveTextFeedIds`: An array of `[int] Id` values for the LiveTextFeed instances being requested. The number of returned items is limited to 100 by the server. Attempting to request more than the maximum allowed number of objects will cause an error, while passing an empty array will lead to an immediate empty response without an error.

`PagedList<LiveTextFeed> FindLiveTextFeeds(string namePattern, string marker, int pageSize)`

Description

Retrieves the next page of a LiveTextFeed list containing names matched with the specified pattern. The returned list is organized by `[string] Name` and may not contain more items than the defined page size.

Required Permissions

Live Text Feed: View Live Text Feeds

Parameters

- `[string] namePattern`: The exact `[string] Name` of the LiveTextFeed instance (or its wildcard-based pattern). Supported wildcards currently include "*", "?", and "['and']".
- `[string] marker`: The `[string] Name` of the last LiveTextFeed instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`LiveTextFeed GetLiveTextFeed(int liveTextFeedId, bool loadContent)`

Description

Retrieves a single LiveTextFeed instance with the specified `[int] Id`. This method returns Null if the LiveTextFeed instance with the specified identifier does not exist.

Required Permissions

Live Text Feed: View Live Text Feed, View Contents

Parameters

- `[int] liveTextFeedId`: The identifier and primary key of the LiveTextFeed instance to be retrieved.
- `[bool] loadContent`: A flag specifying whether the method should also initialize and return the Contents of the Live Text Feed.

`LiveTextFeed GetLiveTextFeedByName(string name, bool loadContent)`

Description

Retrieves the LiveTextFeed instance with the specified `[string] Name`. This method returns Null if the LiveTextFeed instance with the specified name does not exist.

Required Permissions

Live Text Feed: View Live Text Feed, View Contents

Parameters

- `[string] name`: The user-defined Name of the LiveTextFeed instance to be retrieved
- `[bool] loadContent:` : A flag specifying whether the method should also initialize and return the Contents of the Live Text Feed.

`bool CheckLiveTextFeedName(string name)`

Description

Determines whether the specified Live Text Feed `[string] Name` is currently in use. This method returns True if a Live Text Feed with the specified name currently exists.

Note that when using this method to check whether a LiveTextFeed instance can be created, a False status may change in the time between calling this method and calling `CreateLiveTextFeed()`.

Required Permissions

None

Parameters

- `[string] name`: The Live Text Feed Name value to be evaluated.

`bool CheckLiveTextFeedUsage(int liveTextFeedId)`

Description

Determines whether the LiveTextFeed instance (specified by its primary key) is referenced by one or more presentations. This method returns True if the LiveTextFeed instance has parent dependencies.

Note that when using this method to check whether a LiveTextFeed instance can be deleted, a False status may change in the time between calling this method and calling `DeleteLiveTextFeed()`.

Required Permissions

Live Text Feed: View Live Text Feed

Parameters

- `[int] liveTextFeedId`: The identifier and primary key of the LiveTextFeed instance to evaluate. If a LiveTextFeed instance with the specified `[int] Id` does not exist, this method will return False without an error.

`bool CheckLiveTextFeedUsageByName(string name)`

Description

Determines whether a LiveTextFeed instance (specified with its alternate, user-defined key) is referenced by one or more presentations. This method returns True if the LiveTextFeed instance has parent dependencies.

Note that when using this method to check whether a LiveTextFeed instance can be deleted, a False status may change in the time between calling this method and calling `DeleteLiveTextFeed()`.

Required Permissions

Live Text Feed: View Live Text Feed

Parameters

- `[string] name`: The user-defined name of the LiveTextFeed instance to evaluate. If a LiveTextFeed instance with the specified `[string] Name` does not exist, this method will return False without an error.

`LiveTextFeed CreateLiveTextFeed(LiveTextFeed entity)`

Description

Creates a new LiveTextFeed instance and related RSS file with the specified `[string] Name` and `[Dictionary<string,string>] Contents` parameters. This method returns the created object with all initialized properties if successful. If an error occurs during the creation process, the method will return Null.

A descriptive error will be returned if the key strings in the key-value pairs are not unique or if the number of key-value pairs in the object instance surpasses the limit defined on the server.

Required Permissions

Live Text Feed: Create Live Text Feed

Parameters

- `[LiveTextFeed] entity`: A LiveTextFeed object instance with initialized Name and Contents parameters. All other properties will be ignored during object creation. If this parameter is set to Null, then the method will

immediately return Null without an error. A descriptive error will be returned to the client if the LiveTextFeed name is more than 50 characters or if another LiveTextFeed instance has the same name.

`LiveTextFeed CloneLiveTextFeed(string feedUrl)`

Description

Creates a new LiveTextFeed instance and related RSS file using the name and key-value pairs copied from the specified external RSS URL. If successful, this method returns the newly created object with all initialized properties. If an error occurs during object creation, this method returns Null.

A descriptive error will be returned if the key strings in the key-value pairs are not unique or if the number of key-value pairs in the new object instance surpasses the limit defined on the server.

Required Permissions

Live Text Feed: Create Live Text Feed

Parameters

- `[string] feedUrl`: An external URL of a publicly available RSS feed. The RSS Title will be used as the LiveTextFeed instance name and the RSS title-description pairs will converted to key-value pairs. If this parameter is Null or invalid, then the method will immediately return Null without an error. This method will return a descriptive error if the string length of the RSS channel title, plus the “Copy ({index}) of” prefix, exceeds 50 characters.

`bool UpdateLiveTextFeed(LiveTextFeed entity)`

Description

Updates the Name and Contents parameters of the specified LiveTextFeed instance and related RSS file. This method returns True only if completely successful. Otherwise, it returns False.

A descriptive error will be returned if the key strings in the key-value pairs are not unique or if the number of key-value pairs in the object instance surpasses the limit defined on the server.

Required Permissions

Live Text Feed: View Live Text Feed

Parameters

- `[LiveTextFeed] entity`: A LiveTextFeed object instance containing the `[int] Id` of the instance to update, as well as the new Name and Contents parameters for the updated instance. If this parameter is Null or invalid, then the method will immediately return Null without an error. A descriptive error will be returned if any of the following conditions occur:
    - The `[int] Id` does not correspond to an existing LiveTextFeed instance.
    - The length of the specified LiveTextFeed instance `[string] Name` exceeds 50 characters.
    - The specified `[string] Name` is currently in use by another LiveTextFeed instance.
    - The key strings in the key-value pairs are not unique.
    - The number of key-value pairs in the object instance surpasses the limit defined on the server.

```
bool RenameLiveTextFeed(int liveTextFeedId, string newName)
```

Description

Updates the `[string] Name` of the specified LiveTextFeed instance. This method returns True upon success and False upon failure.

Required Permissions

Live Text Feed: Update Live Text Feed, Rename Live Text Feed

Parameters

- `[int] LiveTextFeedId`: The identifier and primary key of the LiveTextFeed instance. If a LiveTextFeed instance with the specified identifier does not exist, the method will return False without an error.
- `[string] newName`: A new Name parameter for the specified LiveTextFeed instance. A descriptive error will be returned if the new Name string is greater than 50 characters.

`bool DeleteLiveTextFeeds(int[] liveTextFeedIds)`

Description

Deletes the specified LiveTextFeed instance(s) and related RSS file(s) in both the database and persistent storage. This method returns True only if completely successful. Otherwise, it returns False.

Required Permissions

Live Text Feed: Delete Live Text Feed

Parameters

- `[int[]] liveTextFeedIds`: An array of `[int] Id` values for the LiveTextFeed instances to be deleted. The number of requested items is limited to 100 by the server. Attempting to request more than the maximum allowed number of objects will cause an error. An error will also be returned if an `[int] Id` parameter does not correspond to an existing LiveTextFeed instance. Passing an empty array or Live Text Feed identifiers that don't exist will lead to an immediate empty response without an error.

WEB PAGE

WebPage Entity

The WebPage entity has the following properties:

- `[int] Id`:(Read Only) The identifier and primary key of the WebPage instance.
- `[string] Name`: The user-defined name of the WebPage instance. This alternate key must be unique in the scope of the BrightSign Network account.
- `[string] PhysicalPath`:(Read Only) An external URL of the HTML file in persistent storage.
- `[long] FileSize`:(Read Only) The size of the associated HTML file in bytes.
- `[string] FileHash`:(Read Only) A SHA1 hash of the associated HTML file content.
- `[DateTime] UploadDate`:(Read Only) A UTC timestamp indicating when the Web Page was uploaded to the BrightSign Network.
- `[WebPageAsset[]] Assets`: A list of WebPageAsset entity-relations that reference Content instances that are in use by the WebPage entity. The list is set to Null when not initialized by the server.
- `[PresentationInfo[]] Presentations`:(Read Only) An array of PresentationInfo structures denoting presentations that are currently utilizing the WebPage instance.

WebPageAsset Entity-Relation

The WebPageAsset entity-relation represents the association between a WebPage instance and a Content instance. It has the following properties:

- `[string] FileName`: The virtual name of the file represented by the associated Content instance.
- `[string] RelativePath`:(Read Only) The relative path of the associated asset file in relation to the Web Page HTML file.
- `[string] PhysicalPath`:(Read Only) The external URL of the associated asset file in persistent storage.
- `[string] FileHash`:(Read Only) The SHA-1 hash of the associated asset file contents. The string is provided in the following format: “SHA1:{FileHash}”.

- `[long] FileSize`:(Read Only) The size (in bytes) of the associated asset file.

PresentationContent Entity-Relation

The PresentationContent entity-relation represents the association between a Presentation instance and a Content instance. It has the following properties:

- `[int] ContentId`:(Read Only) The identifier and primary key of the associated Content instance.
- `[string] FileName`: The virtual name of the file represented by the current Content instance.
- `[int] PresentationId`:(Read Only) The identifier and primary key of the associated Presentation instance.
- `[string] PresentationName`:(Read Only) The user-defined name of the associated Presentation instance.
- `[string] StateName`: The name of the presentation state that corresponds to the associated Content instance.
- `[TimeSpan] DisplayDuration`: The amount of time that the device displays an associated image.
- `[string] ContentTransition`: The screen effect that is shown during the transition between the previous presentation state and the current content state (i.e. the associated content state).

Web Page Management Web Methods

- `PagedList<WebPage> GetWebPages(string marker, int pageSize)`
- `List<WebPage> GetSpecifiedWebPages(int[] webPageIds)`
- `PagedList<WebPage> FindWebPages(string namePattern, int marker, int pageSize)`
- `WebPage GetWebPage(int webPageId, bool loadAssets)`
- `WebPage GetWebPageByName(string name, bool loadAssets)`
- `bool CheckWebPageName(string name)`
- `bool CheckWebPageUsage(int webPageId)`
- `bool CheckWebPageUsageByName(string name)`
- `bool RenameWebPage(int webPageId, string newName)`
- `bool DeleteWebPages(int[] webPageIds)`

`PagedList<WebPage> GetWebPages(string marker, int pageSize)`

Description

Retrieves the next page of the WebPage list, sorted by `[string] Name`. The returned list will contain no more items than the defined page size.

Required Permissions

Web Page: View Web Pages

Parameters

- `[string] marker`: The `[string] Name` of the last WebPage instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`List<WebPage> GetSpecifiedWebPages(int[] webPageIds)`

Description

Retrieves a list of WebPage instances matching the specified identifiers, sorted by `[string] Name`. The identifiers of nonexistent WebPage instances will be ignored.

Required Permissions

Web Page: View WebPages

Parameters

- `[int[]] webPageIds`: An array of `[int] Id` values for the WebPage instances being requested. The number of returned items is limited . Attempting to request more than the maximum allowed number of objects will cause an error, while passing an empty array will lead to an immediate empty response without an error.

`PagedList<WebPage> FindWebPages(string namePattern, string marker, int pageSize)`

Description

Retrieves the next page of a WebPage list containing names matched with the specified pattern. The returned list is organized by `[string] Name` and may not contain more items than the defined page size.

Required Permissions

Web Page: View Web Pages

Parameters

- `[string] namePattern`: The exact `[string] Name` of the WebPage instance (or its wildcard-based pattern). Supported wildcards currently include "*", "?", and "['and']".
- `[string] marker`: The `[string] Name` of the last WebPage instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`WebPage GetWebPage(int webPageId, bool loadAssets)`

Description

Retrieves a single WebPage instance with the specified `[int] Id`. This method returns Null if the WebPage instance with the specified identifier does not exist.

Required Permissions

Web Page: View Web Pages

Parameters

- `[int] webPageId`: The identifier and primary key of the WebPage instance to be retrieved.

- `[bool] loadAssets`: A flag specifying whether the method should also initialize and return a list of all assets currently being used by the Web Page.

`WebPage GetWebPageByName(string name, bool loadContent)`

Description

Retrieves the WebPage instance with the specified `[string] Name`. This method returns Null if the WebPage instance with the specified name does not exist.

Required Permissions

Web Page: View Web Pages

Parameters

- `[string] name`: The user-defined Name of the WebPage instance to be retrieved. The length of the string is limited to 50 characters.
- `[bool] loadAssets`: A flag specifying whether the method should also initialize and return a list of all assets currently being used by the Web Page.

`bool CheckWebPageName(string name)`

Description

Determines whether the specified WebPage entity `[string] Name` is currently in use. This method returns True if a Web Page with the specified name currently exists.

Note that when using this method to check whether a Web Page can be uploaded, a False status may change after this method is called.

Required Permissions

None

Parameters

- `[string] name`: The Web Page Name value to be evaluated.

`bool CheckWebPageUsage(int liveTextFeedId)`

Description

Determines whether the WebPage instance (specified by its primary key) is referenced by one or more presentations. This method returns True if the WebPage instance has parent dependencies.

Note that when using this method to check whether a WebPage instance can be deleted, a False status may change in the time between calling this method and calling `DeleteWebPages()`.

Required Permissions

Web Page: View Web Pages

Parameters

- `[int] webPageId`: The identifier and primary key of the WebPage instance to evaluate. If a WebPage instance with the specified `[int] Id` does not exist, this method will return False without an error.

`bool CheckWebPageUsageByName(string name)`

Description

Determines whether a WebPage instance (specified with its alternate, user-defined key) is referenced by one or more presentations. This method returns True if the WebPage instance has parent dependencies.

Note that when using this method to check whether a WebPage instance can be deleted, a False status may change in the time between calling this method and calling `DeleteWebPages()`.

Required Permissions

Web Page: View Web Pages

Parameters

- `[string] name`: The user-defined name of the WebPage instance to evaluate. If a WebPage instance with the specified `[string] Name` does not exist, this method will return False without an error.

`bool RenameWebPage(int webPageId, string newName)`

Description

Updates the `[string] Name` of the specified WebPage instance. This method returns True upon success and False upon failure.

Required Permissions

Web Page: Update Web Page, Rename Web Page

Parameters

- `[int] webPageId`: The identifier and primary key of the WebPage instance. If a WebPage instance with the specified identifier does not exist, the method will return False without an error.
- `[string] newName`: A new Name parameter for the specified WebPage instance. A descriptive error will be returned if the new Name string is greater than 50 characters.

`bool DeleteWebPages(int[] webPageIds)`

Description

Deletes the specified WebPage instance(s) and related files in both the database and persistent storage. This method returns True only if the operation is completely successful. Otherwise, it returns False.

Required Permissions

Web Page: Delete Web Page

Parameters

- `[int[]] webPageIds`: An array of `[int] Id` values for the WebPage instances to be deleted. The number of items is limited to 100 by the server. Attempting to request more than the maximum allowed number of objects will

cause an error. Passing Web Page identifiers that don’t exist will also result in an error return. Passing an empty array will lead to an immediate empty response without an error.

DEVICE WEB PAGE

DeviceWebPage Entity

The DeviceWebPage entity has the following properties:

- `[int] Id`:(Read Only) The identifier and primary key of the DeviceWebPage instance.
- `[int] AccountId`:(Read Only) The identifier of the account that owns the DeviceWebPage instance.
- `[string] Name`: The user-defined name of the DeviceWebPage instance. This alternate key must be unique in the scope of the BrightSign Network account.
- `[string] PhysicalPath`:(Read Only) An external URL of the HTML file in persistent storage.
- `[long] FileSize`:(Read Only) The size of the associated HTML file in bytes.
- `[string] FileHash`:(Read Only) A SHA1 hash of the associated HTML file content.
- `[DateTime] UploadDate`:(Read Only) A UTC timestamp indicating when the Device Web Page was uploaded to the BrightSign Network.
- `[List<WebPageAsset>] Assets`: A list of WebPageAsset entity-relations that reference Content instances that are in use by the Device WebPage entity. The list is set to Null when not initialized by the server.
- `[List<PresentationContent>] Presentations`:(Read Only) A list of PresentationContent entity-relations denoting parent Presentations.

Device Web Page Management Web Methods

- `PagedList<DeviceWebPage> GetDeviceWebPages(int marker, int pageSize)`
- `List<DeviceWebPage> GetSpecifiedDeviceWebPages(int[] devicewebPageIds)`
- `PagedList<DeviceWebPage> FindDeviceWebPages(string namePattern, int marker, int pageSize)`
- `DeviceWebPage GetDeviceWebPage(int deviceWebPageId, bool loadAssets)`
- `DeviceWebPage GetDeviceWebPageByName(string name, bool loadAssets)`
- `bool CheckDeviceWebPageName(string name)`

- `bool CheckDeviceWebPageUsage(int deviceWebPageId)`
- `bool CheckDeviceWebPageUsageByName(string name)`
- `bool RenameDeviceWebPage(int deviceWebPageId, string newName)`
- `bool DeleteDeviceWebPages(int[] deviceWebPageIds)`

`PagedList<WebPage> GetDeviceWebPages(int marker, int pageSize)`

Description

Retrieves the next page of the DeviceWebPage list, sorted by `[string] Name`. The returned list will contain no more items than the defined page size.

Required Permissions

None

Parameters

- `[int] marker`: The identifier of the last DeviceWebPage instance on the previous page. If this value is negative, the method will retrieve the first page.
- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`List<WebPage> GetSpecifiedDeviceWebPages(int[] webPageIds)`

Description

Retrieves a list of DeviceWebPage instances matching the specified identifiers. The identifiers of nonexistent DeviceWebPage instances will be ignored.

Required Permissions

None

Parameters

- `[int[]] deviceWebPageIds`: An array of `[int] Id` values for the DeviceWebPage instances being requested. The number of items is limited on the server side. Attempting to request more than the maximum allowed number of objects will cause an error, while passing an empty array will lead to an immediate empty response without an error.

`PagedList<WebPage> FindDeviceWebPages(string namePattern, int marker, int pageSize)`

Description

Retrieves the next page of a DeviceWebPage list containing names matched with the specified pattern. The returned list is organized by `[string] Name` and may not contain more items than the defined page size.

Required Permissions

None

Parameters

- `[string] namePattern`: The exact `[string] Name` of the DeviceWebPage instance (or its wildcard-based pattern). Supported wildcards currently include "*", "?", and "['and']".
- `[int] marker`: The `[int] Id` of the last DeviceWebPage instance on the previous page. If the integer is not positive, then the method will retrieve the first page.
- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`DeviceWebPage GetDeviceWebPage(int webPageId, bool loadAssets)`

Description

Retrieves a single DeviceWebPage instance with the specified `[int] Id`. This method returns Null if the Device WebPage instance with the specified identifier does not exist.

Required Permissions

Web Page: View Web Pages

Parameters

- `[int] deviceWebPageId`: The identifier and primary key of the DeviceWebPage instance to be retrieved.
- `[bool] loadAssets`: A flag specifying whether the method should also initialize and return a list of all assets currently being used by the Web Page.

`DeviceWebPage GetDeviceWebPageByName(string name, bool loadContent)`

Description

Retrieves the DeviceWebPage instance with the specified `[string] Name`. This method returns Null if the DeviceWebPage instance with the specified name does not exist.

Required Permissions

None

Parameters

- `[string] name`: The user-defined Name of the DeviceWebPage instance to be retrieved.
- `[bool] loadAssets`: A flag specifying whether the method should also initialize and return a list of all assets currently being used by the Device Web Page.

`bool CheckDeviceWebPageName(string name)`

Description

Determines whether the specified DeviceWebPage `[string] Name` is currently in use. This method returns True if a Device Web Page with the specified name currently exists.

Note that when using this method to check whether a Device Web Page can be uploaded, a False status may change after this method is called.

Required Permissions
None

Parameters
- `[string] name`: The Device Web Page Name value to be evaluated.

`bool CheckDeviceWebPageUsage(int liveTextFeedId)`

Description
Determines whether the DeviceWebPage instance (specified by its primary key) is referenced by one or more presentations. This method returns True if the DeviceWebPage instance has parent dependencies.

Note that when using this method to check whether a DeviceWebPage instance can be deleted, a False status may change in the time between calling this method and calling `DeleteDeviceWebPages()`.

Required Permissions
None

Parameters
- `[int] deviceWebPageId`: The identifier and primary key of the DeviceWebPage instance to evaluate. If a DeviceWebPage instance with the specified `[int] Id` does not exist, this method will return False without an error.

`bool CheckDeviceWebPageUsageByName(string name)`

Description
Determines whether a DeviceWebPage instance (specified with its alternate, user-defined key) is referenced by one or more presentations. This method returns True if the DeviceWebPage instance has parent dependencies.

Note that when using this method to check whether a DeviceWebPage instance can be deleted, a False status may change in the time between calling this method and calling `DeleteDeviceWebPages()`.

Required Permissions

None

Parameters

- `[string] name`: The user-defined name of the DeviceWebPage instance to evaluate. If a DeviceWebPage instance with the specified `[string] Name` does not exist, this method will return False without an error.

`bool RenameDeviceWebPage(int webPageId, string newName)`

Description

Updates the `[string] Name` of the specified DeviceWebPage instance. This method returns True upon success and False upon failure.

Required Permissions

None

Parameters

- `[int] devicWebPageId`: The identifier and primary key of the DeviceWebPage instance. If a DeviceWebPage instance with the specified identifier does not exist, the method will return False without an error.
- `[string] newName`: A new Name parameter for the specified DeviceWebPage instance. A descriptive error will be returned if the new Name string is greater than 50 characters.

`bool DeleteDeviceWebPages(int[] webPageIds)`

Description

Deletes the specified DeviceWebPage instance(s) and related files in both the database and persistent storage. This method returns True only if the operation is completely successful. Otherwise, it returns False.

Required Permissions

None

Parameters

- `[int[]] deviceWebPageIds`: An array of `[int] Id` values for the DeviceWebPage instances to be deleted. The number of items is limited on the server side. Attempting to request more than the maximum allowed number of objects will cause an error. Passing an empty array or Device Web Page identifiers that don’t exist will lead to an immediate empty response without an error.

PRESENTATION

Presentation Entity

The Presentation entities has the following properties:

- `[int] Id`:(Read Only) The identifier and primary key of the Presentation instance.
- `[string] Name`: The user-defined name of the Presentation instance. This string value must be unique in the scope of the BrightSign Network account.
- `[DateTime] CreationDate`:(Read Only) A UTC timestamp indicating when the presentation was created within the BrightSign Network.
- `[DateTime] LastModifiedDate`:(Read Only) A UTC timestamp indicating the last time that presentation was modified.
- `[string] AutorunVersion`:(Read Only) A string version of the device autorun referenced by the Presentation instance. A “(custom)” maker will be included with this string if the presentation references a custom autorun.
- `[DeviceModel] DeviceModel`: The target device model of the Presentation instance represented by a DeviceModel enumeration value.
- `[ScreenSettings] ScreenSettings`: The screen settings of a Presentation instance represented by a ScreenSettings structure.
- `[PresentationLanguage] Language`: A PresentationLanguage enumeration that represents the target language of the presentation. This property currently has no effect on the operation of a presentation.
- `[List<PresentationZone>] Zones`: A list of entity-relations inherited from the PresentationZone structure. The list references Content and ImageVideoDynamicPlaylist/AudioDynamicPlaylist instances used by the current Presentation instance.
- `[GroupInfo[]] Groups`:(Read Only) An array of GroupInfo structures that represent parent Group instances for which the Presentation instance is scheduled.

DeviceModel Enumeration

The DeviceModel enumeration can contain the following values:

- `Unknown`: This value is used by the server for presentations created in BrightAuthor.
- `HD210`
- `HD1010`
- `HD220`
- `HD1020`
- `XD230`
- `XD1030`
- `XD1230`
- `TD1012`
- `AU320`

PresentationLanguage Enumeration

The PresentationLanguage enumeration can contain the following values:

- `Unknown`: This property is used by the server for presentations created in BrightAuthor.
- `English`
- `French`
- `Italian`
- `German`
- `Spanish`
- `Swedish`

ScreenSettings Structure

The ScreenSettings structure has the following properties:

- `[ScreenResolution] Resolution`: A ScreenResolution enumeration representing the resolution of a Presentation entity. It can contain the following values:

Note: *This enumeration is set to Null for presentations created in BrightAuthor*.

- 1920x1080x60p
- 1920x1080x59.94p
- 1920x1080x50p
- 1920x1080x30p
- 1920x1080x29.97p
- 1920x1080x25p
- 1920x1080x24p
- 1920x1080x60i
- 1920x1080x59.94i
- 1920x1080x50i
- 1440x900x75p
- 1440x900x60p
- 1400x1050x75p
- 1400x1050x60p
- 1360x768x60p
- 1280x1024x75p
- 1280x1024x60p
- 1280x960x60p
- 1280x800x75p
- 1280x800x60p
- 1280x768x60p
- 1280x720x60p
- 1280x720x59.94p
- 1280x720x50p
- 1024x768x75p
- 1024x768x60p
- 800x600x75p

    - 800x600x60p
    - 720x576x50p
    - 720x480x60p
    - 720x480x59.94p
    - 640x480x60p
    - SECAM
    - NTSC-COMPONENT
    - PAL-COMPONENT
    - NTSC-M
    - NTSC-M-JPN
    - PAL-I
    - PAL-BG
    - PAL-N
    - PAL-NC
    - PAL-M
- `[string] BackgroundColor`: The background color of a presentation entity. The color value is represented by with the string “RGB: {R:XX}{G:XX}{B:XX)”, where “XX” is equivalent to a two-digit hexadecimal number (e.g. “RGB:4787C7”)
- `[ScreenOrientation] Orientation`: A ScreenOrientation enumeration, which can contain the following values:
    - Landscape
    - Portrait
- `[ConnectorType] Connector`: A ConnectorType enumeration, which can contain the following values:
    - VGA
    - HDMI
    - Component
- `[ScreenOverscan] Overscan`: A ScreenOverscan enumeration, which can contain the following values:
    - NoOverscan

    - `OverscanActionSafeArea`
    - `OverscanTitleSafeArea`

PresentationZone Structure

The PresentationZone structure has the following properties:

- `[int] Id`: The identifier and primary key of the zone.
- `[string] Name`: The user-defined name of the zone. This name of a presentation zone must be unique within the scope of a presentation.
- `[int] X`: The horizontal resolution of the presentation zone.
- `[int] Y`: The vertical resolution of the presentation zone.
- `[int] Width`
- `[int] Height`
- `[List<PresentationContent>] Contents`: A list of PresentationContent entity-relations that represent the association between Content instances and the Presentation instance.

VideoOrImagesPresentationZone Structure

The VideoOrImagesPresentationZone structure is inherited from the PresentationZone structure. It has the following properties:

- `[ViewMode] ViewMode`: A ViewMode enumeration, which can contain the following values:
    - `ScaleToFill`
    - `LetterboxedAndCentered`
    - `FillScreenAndCentered`
- `[ImageMode] ImageMode`: An ImageMode enumeration, which can contain the following values:
    - `CenterImage`
    - `ScaleToFit`
    - `ScaleToFillAndCrop`
    - `ScaleToFill`

- `[AnalogOutputMode] AnalogOutput`: An AnalogOutputMode enumeration, which can contain the following values:
  - `None`
  - `PCM`
- `[HDMIOutputMode] HDMIOutput`: An HDMIOutputMode enumeration, which can contain the following values:
  - `None`
  - `PCM`
  - `PassThrough`
- `[SPDIFOutputMode] SPDIFOutput`: A SPDIFOutputMode enumeration, which can contain the following values:
  - `None`
  - `PCM`
  - `PassThrough`
- `[AudioMixingType] AudioMixing`: An AudioMixingType enumeration, which can contain the following values:
  - `Stereo`
  - `Left`
  - `Right`
- `[int] VideoVolume`: The volume of the video file track, represented as an integer between 0 and 100.
- `[int] AudioVolume`: The volume of the audio file track, represented as an integer between 0 and 100.

EnhancedAudioPresentationZone

The EnhancedAudioPresentationZone structure is inherited from the PresentationZone structure. This structure is reflects presentations created for the AU320. It has the following properties:

- `[AnalogOutputMode] AnalogOutput`: An AnalogOutputMode enumeration, which can contain the following values:
  - `None`
  - `PCM`
- `[HDMIOutputMode] HDMIOutput`: An HDMIOutputMode enumeration, which can contain the following values:

    - None
    - PCM
    - PassThrough
- [SPDIFOutputMode] SPDIFOutput: A SPDIFOutputMode enumeration, which can contain the following values:
    - None
    - PCM
    - PassThrough
- [AudioMixingType] AudioMixing: An AudioMixingType enumeration, which can contain the following values:
    - Stereo
    - Left
    - Right
- [int] AudioVolume: The volume of the audio file track, represented as an integer between 0 and 100.

GroupInfo Structure

The GroupInfo structure is used to represent a parent Group instance for which a Presentation instance is scheduled. It has the following properties:

- [int] Id: The identifier and primary key of the parent Group instance
- [string] Name: The user-defined name of the parent Group instance

Presentation Management Web Methods

- PagedList<Presentation> GetPresentations(string marker, int pageSize)
- List<Presentation> GetSpecifiedPresentations(int[] presentationIds)
- PagedList<Presentation> FindPresentations(string namePattern, string marker, int pageSize)
- Presentation GetPresentation(int presentationId, bool loadContent)
- Presentation GetPresentationByName(string name, bool loadContent)

- ScreenResolution[] GetSupportedScreenResolutions(DeviceModel deviceModel, ConnectorType connectorType)
- bool CheckPresentationName(string name)
- bool CheckPresentationUsage(int presentationId)
- Presentation CreatePresentation(Presentation entity)
- Presentation UpdatePresentation(Presentation entity)
- bool UpdatePresentationScreenSettings(int presentationId, ScreenSettings screenSettings)
- bool UpdatePresentationZone(PresentationZone entity)
- bool DeletePresentations(int[] presentationIds)

`PagedList<Presentation> GetPresentations(string marker, int pageSize)`

Description

Retrieves the next page of the Presentation list, sorted by `[string] Name`. The returned list will contain no more items than the defined page size.

> **Note**: *This method will not be able to initialize the DeviceModel enumeration, PresentationLanguage enumeration, or ScreenSettings structure if a target presentation was created in BrightAuthor. The server will return "unknown" and Null values instead.*

Required Permissions

Presentation: View Presentations

Parameters

- `[string] marker`: The `[string] Name` of the last Presentation instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is

not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`List<Presentation> GetSpecifiedPresentations(int[] presentationIds)`

Description

Retrieves a list of Presentation instances matching the specified identifiers. The results are organized by `[string] Name`. The identifiers of nonexistent Presentation instances will be ignored.

> **Note**: *This method will not be able to initialize the DeviceModel enumeration, PresentationLanguage enumeration, or ScreenSettings structure if a target presentation was created in BrightAuthor. The server will return "unknown" and Null values instead.*

Required Permissions

Presentation: View Presentations

Parameters

- `[int[]] presentationIds`: An array of `[int] Id` values for the Presentation instances being requested. The number of requested items is limited to 100 by the server. Attempting to request more than the maximum allowed number of objects will cause an error, while passing an empty array will lead to an immediate empty response without an error.

`PagedList<Presentation> FindPresentations(string namePattern, string marker, int pageSize)`

Description

Retrieves the next page of a Presentation list containing names matched with the specified pattern. The returned list is organized by `[string] Name` and may not contain more items than the defined page size.

Note: *This method will not be able to initialize the DeviceModel enumeration, PresentationLanguage enumeration, or ScreenSettings structure if a target presentation was created in BrightAuthor. The server will return "unknown" and Null values instead.*

Required Permissions

Presentation: View Presentations

Parameters

- `[string] namePattern`: The exact `[string] Name` of the Presentation instance (or its wildcard-based pattern). Supported wildcards currently include "*", "?", and "['and']".
- `[string] marker`: The `[string] Name` of the last Presentation instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize:` The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

```
Presentation GetPresentation(int presentationId, bool loadAssets)
```

Description

Retrieves a single Presentation instance with the specified `[int] Id`. This method returns Null if he Presentation instance with the specified identifier does not exist.

Note: *This method will not be able to initialize the DeviceModel enumeration, PresentationLanguage enumeration, or ScreenSettings structure if a target presentation was created in BrightAuthor. The server will return "unknown" and Null values instead.*

Required Permissions

Presentation: View Presentations

Parameters

- `[int] presentationId`: The identifier and primary key of the Presentation instance to be retrieved.
- `[bool] loadContent`: A flag specifying whether the method should also initialize and return a list of all Content instances used by the presentation. Setting this flag to True will return an error if the specified presentation was created in BrightAuthor.

`Presentation GetPresentationByName(string name, bool loadContent)`

Description

Retrieves the Presentation instance with the specified `[string] Name`. This method returns Null if the Presentation instance with the specified name does not exist.

Note: *This method will not be able to initialize the DeviceModel enumeration, PresentationLanguage enumeration, or ScreenSettings structure if a target presentation was created in BrightAuthor. The server will return "unknown" and Null values instead.*

Required Permissions

Presentation: View Presentations

Parameters

- `[string] name`: The user-defined Name of the Presentation instance to be retrieved.
- `[bool] loadContent`: A flag specifying whether the method should also initialize and return a list of Content instances used by the presentation. Setting this flag to True will return an error if the specified presentation was created in BrightAuthor.

`ScreenResolution[] GetSupportedScreenResolutions(DeviceModel deviceModel, ConnectorType connectorType)`

Description

Lists screen resolutions supported on the specified device model using the specified connector type.

Required Permissions

None

Parameters

- `[DeviceModel] deviceModel`: A DeviceModel enumeration indicating the model of BrightSign player to evaluate. Passing an unknown device model will result in a Null return value.
- `[ConnectorType] ConnectorType`: A ConnectorType enumeration indicating the A/V connector on the model to evaluate.

`bool CheckPresentationName(string name)`

Description

Determines whether the specified Presentation `[string] Name` is currently in use. This method returns True if a Presentation instance with the specified name currently exists.

Note that when using this method to check whether a Presentation can be uploaded, a False status may change between calling this method and calling `CreatePresentation()`.

Required Permissions

Presentation: View Presentations

Parameters

- `[string] name`: The Presentation Name value to be evaluated.

`bool CheckPresentationUsage(int presentationId)`

Description

Determines whether the Presentation instance (specified by its primary key) is referenced by one or more Group instances. This method returns True if the Presentation instance has parent dependencies.

Note that when using this method to check whether a Presentation instance can be deleted, a False status may change in the time between calling this method and calling `DeletePresentations()`.

Required Permissions
Presentation: View Presentations

Parameters

- `[int] presentationId`: The identifier and primary key of the Presentation instance to evaluate. If a Presentation instance with the specified `[int] Id` does not exist, this method will return False without an error.

`Presentation CreatePresentation(Presentation entity)`

Description
Creates a new Presentation instance and related server files using an initialized Presentation entity. If successful, this method will return the newly created object with all initialized properties. If an error occurs, the method will return a Null value.

> **Note**: *This method has the same limitations as the presentation creation module in the WebUI: A presentation must be a single VideoOrImages zone, and many features—including Live Text Feeds, HTML widgets, custom autoruns, and interactive functions—are not available. Some or all of this functionality may be implemented in subsequent versions of the BrightSign Network Web API.*

Required Permissions
Presentation: Create Presentation – Content: Assign Content

Parameters

- `[Presentation] entity`: A Presentation object instance with initialized Name, DeviceModel, ScreenSettings, and Zones properties. The Zones list property should contain all required Content for the presentation. All other property values will be ignored. If this parameter is set to Null, then the server will immediately return Null without an error. The server will return a descriptive error if any of the following conditions occur:

- The resolution in the ScreenSettings structure is not compatible with the specified device model (see the Output Resolutions FAQ for more details).
- The DeviceModel enumeration value is “unknown”.
- The PresentationLanguage enumeration value is “unknown”.
- The length of the Presentation `[string] Name` is greater than 100 characters.
- The Presentation `[string] Name` is already in use by another presentation in the account.
- The `[int] AudioVolume` or `[int] VideoVolume` of the VideoOrImagesPresentationZone structure is less than 0 or greater than 100.
- The Zones list property of the Presentation instance does not contain a PresentationZone structure.
- The number of PresentationZone structures or Content instances exceeds the limit defined on the server.
- The `[string] Name` of a PresentationZone structure is not unique within the scope of the Presentation instance.
- The length of a PresentationZone `[string] Name` exceeds the limit defined on the server.

`Presentation UpdatePresentation(Presentation entity)`

Description

Updates the settings and Contents list of a Presentation instance (as well as related files in persistent storage). This method returns True only if the operation was completely successful. Otherwise, it returns False. Note that the same limitations apply to this method as to `CreatePresentation()`.

> **Note**: *This method cannot be used to update a presentation created in BrightAuthor. The server will return an error immediately if this is attempted.*

Required Permissions

Presentation: Update Presentation, Add Content, Remove Content – Content: Assign Content, Unassign Content

Parameters

- `[Presentation] entity`: A presentation object instance with a specified identifier, screen settings, etc. This method cannot be used to update a presentation name or target device model; the server will ignore the name or

device model of a passed presentation entity. The instance must also have a Zones list property containing PresentationContent and DynamicPlaylistContent entity-relations. If this parameter is set to Null, then the server will immediately return False without an error. The server will return a descriptive error if any of the following conditions occur:

- The resolution specified in the ScreenSettings structure is not compatible with the player specified in the DeviceModel enumeration (see the Output Resolutions FAQ for more details).
- The PresentationLanguage enumeration value is "unknown".
- The Presentation `[int] Id` does not correspond to an existing Presentation instance.
- The length of the Presentation `[string] Name` exceeds the 256 character limit.
- The Presentation `[string] Name` is already in use by an existing Presentation instance.
- The `[int] AudioVolume` or `[int] VideoVolume` of the VideoOrImagesPresentationZone structure is less than 0 or greater than 100.
- The Zones list property of the Presentation instance does not contain a PresentationZone structure.
- The number of PresentationZone structures or Content instances exceeds the limit defined on the server.

Note: *Currently, presentations created on the BrightSign Network can contain only one zone.*

- The `[string] Name` of a PresentationZone structure is not unique within the scope of the Presentation instance.
- The length of a PresentationZone `[string] Name` exceeds the limit set on the server side.

```
bool UpdatePresentationScreenSettings(int presentationId, ScreenSettings screenSettings)
```

Description

Updates the ScreenSettings structure of an existing Presentation instance and its related service files in persistent storage. This method returns True only if the operation is completely successful. Otherwise, it will return False.

Note: *This method cannot be used to update a presentation created in BrightAuthor. The server will return an error immediately if this is attempted.*

Required Permissions

Presentation: Update Presentation

Parameters

- `[int] presentationId`: The identifier and primary key of the Presentation instance to update. If a Presentation instance with the specified `[int] Id` does not exist, this method will return an error to the client.
- `[ScreenSettings] screenSettings`: A ScreenSettings structure containing updated values for the specified Presentation instance. A descriptive error will be returned if the resolution specified in the ScreenSettings structure is not compatible with the player specified in the DeviceModel enumeration (see the Output Resolutions FAQ for more details).
- . If this parameter is set to Null, this method will immediately return False without an error.

```
bool UpdatePresentationZone(int presentationId, PresentationZone entity)
```

Description

Updates the PresentationZone structure of an existing Presentation instance and its related service files in storage. This method returns True only if the operation is completely successful. Otherwise, it will return False.

> **Note**: *This method cannot be used to update a presentation created in BrightAuthor. The server will return an error immediately if this is attempted.*

Required Permissions

Presentation: Update Presentation, Add Content, Remove Content, Assign Content – Content: Unassign Content

Parameters

- `[int] presentationId`: The identifier of the target Presentation instance. If this parameter is set to Null or set to a negative number, then the method will return False without an error. If the `[int] Id` is positive but does not correspond to an existing Presentation instance, the server will return a descriptive error.

- `[PresentationZone] entity`: A PresentationZone structure with an initialized `[int] Id` and an updated set of properties, including a new Contents list. If this parameter is set to Null, then the method will immediately return Null without an error. The server will return a descriptive error if any of the following conditions occur:
    - The specified PresentationZone `[int] Id` does not match an existing PresentationZone structure.
    - The length of the PresentationZone `[string] Name` exceeds the limit defined on the server side.
    - The PresentationZone `[string] Name` is not unique within the scope of the account.
    - The `[int] AudioVolume` or `[int] VideoVolume` of the VideoOrImagesPresentationZone structure is less than 0 or greater than 100.

`bool DeletePresentations(int[] presentationIds)`

Description

Deletes one or more Presentation instances and related service files in both the database and persistent storage. This method returns True only if the operation is completely successful. Otherwise, it will return False.

Required Permissions

Presentation: Delete Presentation

Parameters

- `[int[]] presentationsIds`: An array of identifiers indicating the Presentation instances that should be deleted. The number of passed items is limited to 100 by the server. Attempting to delete more than the allowed number of objects will result in an error. An error will also be returned if an `[int] Id` does not correspond to an existing Presentation instance. Passing an empty array, or passing an array containing an identifier that does not match an existing Presentation instance, will lead to an immediate False response without an error.

GROUP

Group Entity

The Group entity has the following properties:

- `[int] Id`:(Read Only) The identifier and primary key of the Group instance.
- `[string] Name`: The user-defined name of the Group instance. This is an alternate key and must be unique within the scope of the account.
- `[DateTime] CreationDate`:(Read Only) A UTC timestamp indicating when the Group instance was created within the BrightSign Network.
- `[string] MinAutorunVersion`:(Read Only) The minimum version of device autorun required to play presentations scheduled for the Group instance.
- `[string] HDx10Firmware`: The minimum version of device firmware required to play presentations for the following player models: HD210w, HD1010, HD1010w.
- `[string] HDx20Firmware`: The minimum version of device firmware required to play presentations for the following player models: AU320, HD220, HD1020.
- `[string] XDx30Firmware`: The minimum version of device firmware required to play presentations for the following player models: XD230, XD1030, XD1230.
- `[bool] EnableSerialDebugging`: A flag specifying whether serial debugging should be enabled for all players assigned to the Group instance.
- `[bool] EnableSystemLogDebugging`: A flag specifying whether system log debugging should be enabled for all players assigned to the Group instance.
- `[int] DevicesCount`:(Read Only) An integer indicating the number of players assigned to the Group instance.
- `[List<Device>] Devices`:(Read Only) A list of Device entities that are assigned to the current group.
- `[PresentationInfo[]] Presentations`:(Read Only) An array of PresentationInfo structures denoting presentations that are currently scheduled for the Group instance.

Group Management Web Methods

- PagedList<Group> GetGroups(string marker, int pageSize)
- List<Group> GetSpecifiedGroups(int[] groupIds)
- PagedList<Group> FindGroups(string namePattern, string marker, int pageSize)
- Group GetGroup(int groupId, bool loadDevices)
- Group GetGroupByName(string name, bool loadDevices)
- Group CreateGroup(Group entity)
- bool DeleteGroup(int groupId, int reassignmentGroupId)

`PagedList<Group> GetGroups(string marker, int pageSize)`

Description

Retrieves the next page of the Group list, sorted by `[string] Name`. The returned list will contain no more items than the defined page size.

Required Permissions

Group: View Groups

Parameters

- `[string] marker`: The `[string] Name` of the last Group instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`List<Group> GetSpecifiedGroups(int[] presentationIds)`

Description

Retrieves a list of Group instances matching the specified identifiers. The list is sorted by the `[string] Name` of the Group instances. The identifiers of nonexistent Group instances will be ignored.

Required Permissions

Group: View Groups

Parameters

- `[int[]] groupIds`: An array of `[int] Id` values for the Group instances being requested. The number of returned items is limited to 100 by the server. Attempting to request more than the maximum allowed number of objects will cause an error, while passing an empty array will lead to an immediate empty response without an error.

`PagedList<Group> FindGroups(string namePattern, string marker, int pageSize)`

Description

Retrieves the next page of a Group list containing names matched with the specified pattern. The returned list is organized by `[string] Name` and may not contain more items than the defined page size.

Required Permissions

Presentation: View Presentations

Parameters

- `[string] namePattern`: The exact `[string] Name` of the Group instance (or its wildcard-based pattern). Supported wildcards currently include "*", "?", and "['and']".
- `[string] marker`: The `[string] Name` of the last Group instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is

not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`Group GetGroup (int presentationId, bool loadAssets)`

Description

Retrieves a single Group instance with the specified `[int] Id`. This method returns Null if he Group instance with the specified identifier does not exist.

Required Permissions

Group: View Groups

Parameters

- `[int] groupId`: The identifier and primary key of the Group instance to be retrieved.
- `[bool] loadDevices`: A flag specifying whether the method should also initialize and return a list of all Device instances that are assigned to the specified Group.

`Group GetGroupByName(string name, bool loadContent)`

Description

Retrieves the Group instance with the specified `[string] Name`. This method returns Null if the Group instance with the specified name does not exist.

Required Permissions

Group: View Groups

Parameters

- `[string] name`: The user-defined Name of the Group instance to be retrieved.
- `[bool] loadDevices`: A flag specifying whether the method should also initialize and return a list of all Device instances that are assigned to the specified Group.

`Group CreateGroup(Group entity)`

Description

Creates a new Group instance with the specified name and settings. If successful, this method will return a Group instance with all initialized properties. If an error occurs, the method will return Null.

Required Permissions

Group: Create Group

Parameters

- `[Group] entity`: A Group object instance with an initialized name, device autorun version, firmware versions, and debugging settings. Other property values will be ignored. If this parameter is set to Null, the method will immediately return False without an error. A descriptive error will be returned if any of the following conditions occur:
  - The length of the specified Group `[string] Name` is greater than 50 characters.
  - The specified Group `[string] Name` is already in use by another Group instance within the account.
  - Creating the Group instance will cause the total number of Group instances to exceed the limit set on the server.

`bool DeleteGroup(int groupId, int reassignmentGroupId)`

Description

Deletes the specified Group instance and assigns all devices within it to another specified Group instance. This method only returns True if completely successful. Otherwise, it returns False.

Required Parameters

Group: Delete Group, Remove Device – Device: Change Target Group

Parameters

- `[int] groupId`: The identifier of the Group instance to be deleted. If the `[int] Id` value is negative, the method will immediately return False without an error. A descriptive error will be returned if the Group specified for

deletion is reserved (i.e. the default “Unassigned” Group instance) or if the `[int] Id` value is positive and does not correspond to an existing Group instance.

- `[int] reassignmentGroupId`: The identifier of the Group instance to which affected devices should be reassigned. If this value is negative, the devices will be reassigned to the default “Unassigned” Group instance. If a Group instance with the specified positive `[int] Id` does not exist, the method will return a descriptive error.

SCHEDULE

ScheduledPresentation Entity-Relation

The ScheduledPresentation entity-relation represents the relationship between a Presentation and Group instance in terms of its schedule. This entity-relation has the following properties.

- `[int] Id`:(Read Only) The identifier and primary key of a scheduled Presentation instance.
- `[int] PresentationId`: The identifier of the Presentation instance that is scheduled for the associated Group instance.
- `[string] PresentationName`: The name of the Presentation instance that is scheduled for the associated Group instance.
- `[bool] IsRecurrent`: A flag specifying whether the related presentation is played periodically at specified times and days of the week.
- `[Nullable<DateTime>] EventDate`: A DateTime value specifying the date when a non-recurrent presentation should begin playing. A Null value can be set for recurrent presentations that do not have a defined start date.
- `[TimeSpan] StartTime`: A TimeSpan value specifying the time when a presentation should begin playing.
- `[TimeSpan] Duration`: A TimeSpan value specifying how long a presentation should play.
- `[Nullable<DateTime>] RecurrenceStartDate`: A DateTime value indicating the date when a recurrently scheduled presentation should begin playing. A Null value can be set for recurrent presentations that do not have a defined start date.
- `[Nullable<DateTime>] RecurrenceEndDate`: A DateTime value indicating the date when a recurrently scheduled presentation should terminate. A Null value can be set for recurrent presentations that do not have a defined end date.
- `[DayOfWeek] DaysOfWeek`: A value indicating the days of the week during which a recurrently scheduled presentation should play.

Note: *A presentation scheduled for "all day, every day" can be created with the following property settings:* `StartTime` *as "00:00:00",* `EndTime` *as "1.00:00:00" (or "24:00:00"), and* `DaysofWeek` *as "EveryDay";* `EventDate`, `RecurrenceStartDate`, *and* `RecurrenceEndDate` *are set to Null.*

- `[DateTime] CreationDate`:(Read Only) A UTC timestamp indicating when the related Presentation instance was scheduled for the associated Group instance.
- `[DateTime] LastModifiedDate`:(Read Only) A UTC timestamp indicating when the current schedule was last updated.
- `[DateTime] ExpirationDate`:(Read Only) A UTC timestamp indicating when the current schedule will expire.

Schedule Management Web Methods

- `PagedList<ScheduledPresentation> GetGroupSchedule(int groupId, string marker, int pageSize)`
- `ScheduledPresentation AddScheduledPresentation(ScheduledPresentation entity)`
- `bool UpdateScheduledPresentation(ScheduledPresentation entity)`
- `bool OverwriteSchedule(int groupId, ScheduledPresentation[] entities)`
- `bool RemoveScheduledPresentation(int scheduledPresentationId)`

`PagedList<ScheduledPresentation> GetGroupSchedule(int groupId, string marker, int pageSize)`

Description

Retrieves the next page of the ScheduledPresentation list associated with the specified Group instance. The list is sorted by the `[DateTime] ExpirationDate` of the ScheduledPresentation entity-relations. The returned list will contain no more items than the defined page size.

Required Permissions

Schedule: View Presentations

Parameters

- `[int] groupID`: The identifier of the Group instance that is associated with the schedule. If a Group instance with the specified identifier does not exist, the method will immediately return Null without an error.
- `[string] marker`: The `[string] ExpirationDate` of the last ScheduledPresentation instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`ScheduledPresentation AddScheduledPresentation(int groupId, ScheduledPresentation entity)`

Description

Schedules a Presentation instance for the specified Group instance and updates related service files in the storage. If successful, this method will return the newly created ScheduledPresentation entity-relation. This method will return Null if an error occurs.

Required Parameters

Schedule: Update Schedule, Add Presentation – Presentation: Assign Presentation

Parameters

- `[int] groupId`: The identifier of the Group instance for which the Presentation instance should be scheduled.
- `[ScheduledPresentation] entity`: A ScheduledPresentation object instance with a Presentation `[int] Id` and/or Presentation `[string] Name`. A Group `[int] Id` and/or Group `[string] Name` is also required. If this parameter is set to Null, then this method will immediately return Null without an error. A descriptive error will be returned if any of the following conditions occur:
    - The specified Presentation `[int] Id` or Presentation `[string] Name` does not correspond to an existing Presentation instance.

- The specified Group `[int] Id` or Group `[string] Name` does not correspond to an existing Group instance.
- The `[TimeSpan] StartTime` value of the ScheduledPresentation entity-relation is less than “00:00:00” or greater than “24:00:00”.
- The `[TimeSpan] Duration` value of the ScheduledPresentation entity-relation is less than “00:00:00” or greater than “24:00:00” minus the `[TimeSpan] StartTime`.
- A recurrent schedule does not have Start Time, Duration, or Days of Week values.
- A recurrent schedule contains a Recurrence Start Date value but not a Recurrence End Date value, or vice-versa.
- A specified Recurrence Start Date value is greater than the Recurrence End Date value.
- A non-recurrent schedule does not have Event Date, Start Time, or Duration values.
- The specified ScheduledPresentation instance conflicts with another Presentation instance already scheduled for the group.

`bool UpdateScheduledPresentation(ScheduledPresentation entity)`

Description

Updates the settings of the specified ScheduledPresentation instance and related service files in storage.

Required Permissions

Schedule: Update Schedule, Add Presentation, Remove Presentation – Presentation: Assign Presentation, Unassign Presentation

Parameters

- `[ScheduledPresentation] entity`: A ScheduledPresentation object instance with an initialized ScheduledPresentation `[int] Id` and an updated Presentation `[int] Id` and/or Presentation `[string] Name`. The instance must also contain Start Time, Duration, Is Recurrent, Event Date, Recurrence Start Date, Recurrence End Date, and Days Of Week properties. All other properties will be ignored. If this parameter is set to Null, then

the method will immediately return Null without an Error. A descriptive error will be returned if any of the following conditions occur:

- The specified ScheduledPresentation `[int] Id` does not correspond to an existing ScheduledPresentation instance.
- The specified Presentation `[int] Id` or Presentation `[string] Name` does not correspond to an existing Presentation instance.
- The `[TimeSpan] StartTime` value of the ScheduledPresentation entity-relation is less than “00:00:00” or greater than “24:00:00”.
- The `[TimeSpan] Duration` value of the ScheduledPresentation entity-relation is less than “00:00:00” or greater than “24:00:00” minus the `[TimeSpan] StartTime`.
- A recurrent schedule does not have Start Time, Duration, or Days of Week values.
- A recurrent schedule contains a Recurrence Start Date value but not a Recurrence End Date value, or vice-versa.
- A specified Recurrence Start Date value is greater than the Recurrence End Date value.
- A non-recurrent schedule does not have Event Date, Start Time, or Duration values.
- The specified ScheduledPresentation instance conflicts with another Presentation instance already scheduled for the group.

```
bool OverwriteSchedule(int groupId, ScheduledPresentation[] entities)
```

Description

Completely overwrites one or more schedules associated with the specified Group instance and updates related service files in the storage. This method returns True only if it is completely successful. Otherwise, it will return False.

Required Permissions

Schedule: Update Schedule, Add Presentation, Remove Presentation – Presentation: Assign Presentation, Unassign Presentation

Parameters

- `[int] groupId`: The identifier of the Group instance to modify. If the Group `[int] Id` does not correspond to an existing Group instance, the method will immediately return False without an error.
- `[ScheduledPresentation[]] entities`: A list of ScheduledPresentation instances, each containing an initialized Presentation `[int] Id` and/or Presentation `[string] Name`. Each instance should also contain Start Time, Duration, Is Recurrent, Event Date, Recurrence Start Date, Recurrence End Date, and Days of Week properties. All other properties will be ignored. If this parameter is set to Null or is passed an empty list, the method will delete all existing ScheduledPresentation entities associated with the specified Group instance. A descriptive error will be returned if any of the following conditions occur:
    - The specified Presentation `[int] Id` or Presentation `[string] Name` does not correspond to an existing Presentation instance.
    - The `[TimeSpan] StartTime` value of the ScheduledPresentation entity-relation is less than “00:00:00” or greater than “24:00:00”.
    - The `[TimeSpan] Duration` value of the ScheduledPresentation entity-relation is less than “00:00:00” or greater than “24:00:00” minus the `[TimeSpan] StartTime`.
    - A recurrent schedule does not have Start Time, Duration, or Days of Week values.
    - A recurrent schedule contains a Recurrence Start Date value but not a Recurrence End Date value, or vice-versa.
    - A specified Recurrence Start Date value is greater than the Recurrence End Date value.
    - A non-recurrent schedule does not have Event Date, Start Time, or Duration values.
    - The specified ScheduledPresentation instances conflict with each other.

```
bool RemoveScheduledPresentation(int scheduledPresentationId)
```

Description

Deletes the specified ScheduledPresentation instance and updates related service files in the storage. This method returns True only if completely successful. Otherwise, it will return False.

Required Permissions

Schedule: Update Schedule, Remove Presentation – Presentation Unassign Presentation

Parameters

- `[int] ScheduledPresentationId`: An identifier indicating the ScheduledPresentation instance that should be deleted.

DEVICE

Device Entity

The Device entity has the following properties:

- `[int] Id`:(Read Only) The identifier and primary key of the Device instance.
- `[string] Serial`:(Read Only) The serial number of the device. This is a unique alternate key for the Device instance.
- `[string] Name`: The user-defined name of the device. This designation does not need to be unique within the scope of an account.
- `[string] Description`: The user-defined description of the device. The description does not need to be unique within the scope of an account.
- `[DeviceSubscription] CurrentSubscription`: (Read Only) A DeviceSubscription instance representing the BrightSign Network subscription that is currently in use by the associated device.
- `[int] TargetGroupId`: The identifier of the Group instance to which the device is assigned.
- `[int] ReportedGroupId`:(Read Only) The identifier of the Group instance that the device reported belonging to during its last synchronization with the server.
- `[Nullable<int>] PresentationId`: (Read Only) The identifier of the Presentation instance that the device reported playing during its last synchronization with the server. A Null value indicates that the device was not playing a presentation during its last synchronization with the server.
- `[string] PresentationName`:(Read Only) The user-defined name of the Presentation instance that the device reported playing during its last synchronization with the server.
- `[DeviceModel] Model`:(Read Only) A DeviceModel enumeration indicating the model number of the device.
- `[string] AutorunVersion`:(Read Only) The version of autorun script that the device was using the last time it synced with the server.
- `[string] FirmwareVersion`:(Read Only) The version of firmware that the device was using the last time it synced with the server.

- `[string] CardSize`:(Read Only) The total size of the device storage. This information is displayed in the format “{0} MiB” or “{0:F2} GiB” (where F2 is equivalent to a fixed-point number with two decimal digits (e.g. “1.75 GiB”)).
- `[string] CardFreeSize`:(Read Only) The amount of free space on the device storage. This information is displayed in the format “{0} MiB” or “{0:F2} GiB” (where F2 is equivalent to a fixed-point number with two decimal digits (e.g. “1.75 GiB”)).
- `[string] TargetTimeZone`: The string name of the time zone that the device should use.
- `[string] ReportedTimeZone`: (Read Only) The string name of the time zone that the device reported using the last time it synced with the server.
- `[DeviceConnectionsPeriod] ContentCheckPeriod`: A DeviceConnectionsPeriod enumeration representing the time interval between synchronization events, when the device checks with the BrightSign Network servers for content updates.
- `[Nullable<TimeSpan>] ContentDownloadsStartTime`: The time each day (according to the reported time zone) when the device should begin downloading content from the BrightSign Network persistent storage. A Null value indicates that the device can download content at any time during the day.
- `[Nullable<TimeSpan>] ContentDownloadsEndTime`: The time each day (according to the reported time zone) when the device should finish downloading content from the BrightSign Network persistent storage. A Null value indicates that the device can download content at any time during the day.
- `[DeviceConnectionsPeriod] HealthReportingPeriod`: A DeviceConnectionsPeriod enumeration representing the interval between health-report events, when the device checks in with the server to report its status.
- `[Nullable<TimeSpan>] HealthReportingStartTime`: The time each day (according to the reported time zone) when the device should begin sending health status updates to the BrightSign Network servers. A Null value indicates that the device can send reports at any time during the day.
- `[Nullable<TimeSpan>] HealthReportingEndTime`: The time each day (according to the reported time zone) when the device should stop sending health status updates to the BrightSign network servers. A Null value indicates that the device can send reports at any time during the day.
- `[DateTime] LastContentCheckTime`:(Read Only) A UTC timestamp indicating when the device last checked in with the BrightSign Network servers for content updates.

- `[DateTime] LastSyncSpecProcessedTime`
- `[DateTime] LastContentDownloadStartTime`:(Read Only) A UTC timestamp indicating when the device last began downloading content from the BrightSign Network servers.
- `[DateTime] LastContetnDownloadEndTime`:(Read Only) A UTC timestamp indicating when the device last completed downloading content from the BrightSign Network servers.
- `[DateTime] LastHeartbeatTime`: (Read Only) A UTC timestamp indicating when the device last reported its health status to the BrightSign Network servers.
- `[DateTime] LastErrorTime`: (Read Only) A UTC timestamp indicating the most recent time that the device reported a current error.
- `[DeviceHealthStatus] HealthStatus`:(Read Only) A DeviceHealthStatus enumeration value representing the current health status of the device.
- `[DeviceNetworkSettings] NetworkSettings`: A DeviceNetworkSettings structure containing network settings for the associated device.
- `[DeviceLogsSettings] LogsSettings`: A DeviceLogsSettings structure containing logging settings for the associated device.
- `[bool] EnableDiagnosticWebServer`: A flag specifying whether the Diagnostic Web Server should be enabled on the device.
- `[string] DiagnosticWebServerPassword`: A password for restricting access to the Diagnostic Web Server. For security reasons, this property will always return Null (even if it has a value) when called by the following device management web methods: `GetAllDevices`, `GetSpecifiedDevices`, `FindDevices`, `GetDevice`, `GetDeviceBySerial`.

DeviceSubscription Entity

The DeviceSubscription entity represents the application of a purchased BrightSign Network subscription to a device. It has the following properties:

- `[int] Id`:(Read Only) The identifier and primary key of a DeviceSubscription instance.

- `[int] AccountId`:(Read Only) The identifier of the BrightSign Network account that owns the DeviceSubscription instance.
- `[int] DeviceId`:(Read Only) The identifier of the Device instance that is currently using the DeviceSubscription instance. This value can be Null if the subscription is not currently assigned to a device.
- `[DeviceSubscriptionType] Type`:(Read Only) A DeviceSubscriptionType enumeration value representing the type of the DeviceSubscription instance.
- `[DeviceSubscriptionStatus] Status`:(Read Only) A DeviceSubscriptionStatus enumeration value representing the current status of the DeviceSubscription instance.
- `[DateTime] CreationDate`:(Read Only) A UTC timestamp indicating when the DeviceSubscription instance was created in the BrightSign Network.
- `[Nullable<DateTime>] ActivationDate`:(Read Only) A UTC timestamp indicating when the subscription was activated by a device. This property is Null if the subscription has not been activated yet.
- `[Nullable<DateTime>] SuspensionDate`:(Read Only) A UTC timestamp indicating when the status of the DeviceSubscription was changed to “Suspended”. This property is Null if the DeviceSubscriptionStatus is currently “Active”.
- `[Nullable<DateTime>] ExpirationDate`: A UTC timestamp indicating when the DeviceSubscription will expire. This property is Null if the DeviceSubscription instance has not been activated.

DeviceLogReport Entity

The DeviceLogReport entity represents a single device log file stored on the servers. It has the following properties:

- `[int] Id`:(Read Only) The identifier and primary key of the DeviceLogReport instance.
- `[string] Name`:(Read Only) The name of the DeviceLogReport instance that is visible to end users.
- `[DeviceLogType] Type`:(Read Only) The type of the DeviceLogReport instance. This value is represented by a DeviceLogType enumeration.
- `[DateTime] StartDate`:(Read Only) A UTC timestamp that represents the beginning of the time range covered by the DeviceLogReport instance.

- `[DateTime] EndDate`:(Read Only) A UTC timestamp that represents the end of the time range covered by the DeviceLogReport instance.
- `[DateTime] LastUpdate`:(Read Only) A UTC timestamp indicating when the latest portion of the device log has been appended to the current DeviceLogReport instance.
- `[string] FilePath`:(Read Only) An external URL indicating where the device log report file is saved in the persistent storage.

DeviceSubscriptionType Enumeration

The DeviceSubscriptionType enumeration can have the following values:

- `Grace`: A free subscription that is created automatically for a device that connects to the BrightSign Network for the first time. A grace subscription lasts for 30 days and can only be used once by a single device.
- `Demo`: A free subscription that can only be created by the BrightSign Network administrators. BrightSign network accounts are limited to a maximum of four demo subscriptions, and each device assigned a demo subscription has content downloads capped at 1 GiB.
- `Monthly`: A commercial subscription that is valid for thirty days after the activation date.
- `Quarterly`: A commercial subscription that is valid for three months after the activation date.
- `Yearly`: A Commercial subscription that is valid for one year after the activation date.

DeviceLogType Enumeration

The DeviceLogType enumeration can have the following values (see this FAQ for more information about log types):

- `Event`
- `Playback`
- `Diagnostic`
- `State`

DeviceSubscriptionStatus Enumeration

The DeviceSubscriptionStatus enumeration can have the following values:

- `Active`: The normal operational status of a subscription.
- `Suspending`: An intermediate status between “Active” and “Suspended”, indicating that the subscription is still operational but will be suspended in less than 14 days.
- `Suspended`: The status of a subscription that is non-operational but has not expired yet.

DeviceConnectionsPeriod Enumeration

The DeviceConnectionsPeriod enumeration represents the interval of time between device-server communication events. It can have the following values:

- `Custom`: The value assigned to all devices that are configured using BrightAuthor.
- `FiveMinutes`
- `FifteenMinutes`
- `ThirtyMinutes`
- `OneHour`
- `SixHours`
- `TwelveHours`
- `OneDay`

DeviceNetworkSettings Structure

The DeviceNetworkSettings structure has the following values:

- `[string] ProxyServer`: The host name and port of a proxy server for the device to work through.
- `[string] TimeServer`: The URL of an NTP service endpoint or HTTP 1.1 resource that the device should use as a source for clock synchronization.
- `[string] Broadcast`
- `[DeviceWiredSettings] Wired`: A DeviceWiredSettings structure that contains wired connection settings for the device.
- `[DeviceWirelessSettings] Wireless`: A DeviceWirelessSettings structure that contains wireless settings for the device.

- `[int] WiredConnection Priority`: A number that defines the priority of a wired connection for the device.
- `[int] WirelessConnectionPriority`: A number that defines the priority of a wireless connection for the device

DeviceWiredSettings Structure

The DeviceWiredSettings structure has the following values:

- `[bool] UseDHCP`: A flag specifying whether the device should use DHCP to obtain settings for the wired connection.

Note: *If the `[bool] UseDHCP` property is set to False, then the following properties should be defined.*

- `[string] IpAddress`: A user-defined IPv4 address for the device wired connection.
- `[string] SubnetMask`: A user-defined IPv4 subnetwork mask for the device wired connection.
- `[string] DefaultGateway`: A user-defined IPv4 default gateway address for the device wired connection.
- `[string] DNS1`: A user-defined IPv4 address of a preferred DNS server for the device wired connection.
- `[string] DNS2`: A user-defined IPv4 address of an alternate DNS server for the device wired connection.
- `[string] DNS3`: A user-defined IPv4 address of a second alternate DNS server for the device wired connection.

DeviceWirelessSettings Structure

The DeviceWirelessSettings structure has the following values:

- `[bool] Enabled`: A flag specifying whether the wireless module should be enabled or disabled.
- `[string] SSID`: The service set identifier of the wireless network to which the device should connect.
- `[string] Passphrase`: The security key for the specified wireless network. For security reasons, this property will always return Null (even if it has a value) when called by the following device management web methods: `GetAllDevices`, `GetSpecifiedDevices`, `FindDevices`, `GetDevice`, `GetDeviceBySerial`.
- `[bool] UseDHCP`: A flag specifying whether the device should use DHCP to obtain settings for the wireless connection.

Note: *If the `[bool] UseDHCP` property is set to False, then the following properties should be defined.*

- `[string] IpAddress`: A user-defined IPv4 address for the device wireless connection.

- `[string] SubnetMask`: A user-defined IPv4 subnetwork mask for the device wireless connection.
- `[string] DefaultGateway`: A user-defined IPv4 address of a default gateway for the device wireless connection.
- `[string] DNS1`: A user-defined IPv4 address of a preferred DNS server for the device wireless connection.
- `[string] DNS2`: A user-defined IPv4 address of an alternate DNS server for the device wireless connection.
- `[string] DNS3`: A user-defined IPv4 address of a second alternate DNS server for the device wireless connection.

DeviceLogsSettings Structure

The DeviceLogsSettings structure has the following values:

- `[bool] EnableDiagnosticLog`: A flag specifying whether the device should generate a diagnostic log.
- `[bool] EnableEventLog`: A flag specifying whether the device should generate an event log.
- `[bool] EnablePlaybackLog`: A flag specifying whether the device should generate a playback log.
- `[bool] EnableStateLog`: A flag specifying whether the device should generate a state log.
- `[bool] UploadAtBoot`: A flag specifying whether the device should upload the current logs to the BrightSign Network during the boot process.
- `[Nullable<TimeSpan>] UploadTime`: The time each day (according to the reported time zone) when the device should upload its current logs. A Null value indicates that the device should not regularly upload logs. The device can still upload logs during the boot process or via direct commands from the user.

DeviceHealthStatus Enumeration

The DeviceHealthStatus enumeration has the following values:

- `Normal`
- `Warning`
- `Error`
- `NoSubscription`

DeviceError Structure

The DeviceError structure has the following values.

- `[DateTime] Timestamp`:(Read Only) A UTC timestamp indicating the most recent time that the device reported a current error.
- `[sting] ErrorName`
- `[string] ErrorEvent`
- `[string] Reason`
- `[int] ResponseCode`

DeviceDownload Structure

The DeviceDownload structure has the following values:

- `[DateTime] Timestamp`:(Read Only) A UTC timestamp indicating when the device reported the current download.
- `[string] FileName`:(Read Only) The name of the file that is being (or has been) downloaded by the device.
- `[string] FileHash`:(Read Only) A SHA1 hash of the file that is being (or has been) downloaded by the device.
- `[int] Progress`:(Read Only) A value ranging from 0 to 100 indicating the progress of the current download.
- `[string] Status`:(Read Only) The status of the current download.

Device Management Web Methods

- `PagedList<Device> GetAllDevices(string marker, int pageSize)`
- `List<Device> GetSpecifiedDevices(int[] deviceIds)`
- `PagedList<Device> FindDevices(string namePattern, string marker, int pageSize)`
- `Device GetDevice(int deviceId)`
- `Device GetDeviceBySerial(string serial)`
- `PagedList<DeviceError> GetDeviceErrors(int deviceId, string marker, int pageSize)`
- `PagedList<DeviceDownload> GetDeviceDownloads(int deviceId, string marker, int pageSize)`

- `bool RenameDevice(int deviceId, string newName, string newDescription)`
- `bool UpdateDeviceLogsSettings(int deviceId, DeviceLogsSettings settings)`
- `bool ReassignDevices(int[] deviceIds, int newGroupId)`
- `bool ForceDevicesReboot(int[] deviceIds)`
- `bool CancelDevicesReboot(int[] deviceIds)`
- `bool ForceDevicesRecovery(int[] deviceIds, bool reformat)`
- `bool CancelDevicesRecovery(int[] deviceIds)`
- `bool ForceDevicesLogsUpload(int[] deviceIds)`
- `bool CancelDevicesLogsUpload(int[] deviceIds)`
- `PagedList<DeviceLogReport> GetDeviceLogReports(int DeviceId, DeviceLogType logType, string marker, int pageSize)`
- `bool DeleteDevices(int[] deviceIds)`

Note: *Devices can only be added to a BrightSign Network account automatically during the player setup process. Consequently, there is no equivalent* `CreateDevice()` *web method in the API.*

`PagedList<Device> GetAllDevices(string marker, int pageSize)`

Description

Retrieves the next page of the Devices list, sorted by serial number. This method will return no more items than the defined page size.

Required Permissions

Device: View Devices

Description

- `[string] marker`: The `[string] Serial` of the last Device instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is

not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`List<Device> GetSpecifiedDevices(int[] deviceIds)`

Description

Retrieves a list of Device instances matching the specified identifiers. The list is sorted by device serial number. The identifiers of nonexistent Device instances will be ignored.

Required Permissions

Device: View Devices

Parameters

- `[Int[]] deviceIds`: An array of `[int] Id` values for the Device instances being requested. The number of items is limited to 100 by the server. Attempting to request more than the maximum allowed number of objects will cause an error, while passing an empty array will lead to an immediate empty response without an error.

`PagedList<Device> FindDevices(string namePattern, string marker, int pageSize)`

Description

Retrieves the next page of a Device list containing names matched with the specified pattern. The returned list is organized by `[string] Serial` and may not contain more items than the defined page size.

Required Permissions

Device: View Devices

Parameters

- `[string] namePattern`: The exact `[string] Name` of the Device instance (or its wildcard-based pattern). Supported wildcards currently include “*”, “?”, and “[‘and’]”.
- `[string] marker`: The `[string] Serial` of the last Device instance on the previous page. If the value is Null, then the method will retrieve the first page.

- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`Device GetDevice(int deviceId)`

Description

Retrieves a single Device instance with the specified `[int] Id`. This method returns Null if a Device instance with the specified identifier does not exist.

Required Permissions

Device: View Devices

Parameters

- `[int] DeviceId`: The identifier and primary key of the requested Device instance.

`Device GetDeviceBySerial(string serial)`

Description

Retrieves a single Device instance with the specified serial number. This method returns Null if a Device instance with the specified identifier does not exist.

Required Permissions

Device: View Devices

Parameters

- `[string] serial`: The serial number of the device. This is an alternate key for a Device instance.

`PagedList<DeviceError> GetDeviceErrors(int deviceId, string marker, int pageSize)`

Description

Retrieves the next page of the DeviceError list, sorted by Timestamp. This method will return no more items than the defined page size.

Required Permissions

Device: View Device Errors

Parameters

- `[int] deviceId`: The identifier and primary key of the device reporting the requested errors. The method returns Null if a Device instance with the specified identifier does not exist.
- `[string] marker`: The timestamp of the last DeviceError instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`PagedList<DeviceDownload> GetDeviceDownloads(int deviceId, string marker, int pageSize)`

Description

Retrieves the next page of the DeviceDownload list, sorted by timestamp. This method will return no more items than the defined page size.

Required Permissions

Device: View Device Errors

Parameters

- `[int] deviceId`: The identifier and primary key of the device conducting the requested downloads. The method returns Null if a Device instance with the specified identifier does not exist.

- `[string] marker`: The timestamp of the last DeviceDownload instance on the previous page. If the value is Null, then the method will retrieve the first page.
- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

```
bool RenameDevice(int deviceId, string newName, string newDescription)
```

Description

Updates the `[string] Name` and `[string] Description` of the specified Device instance. This method returns True only if the operation was completely successful. Otherwise, it returns False.

Required Permissions

Device: Update Device

Parameters

- `[int] deviceId`: The identifier and primary key of the Device instance to be renamed. If a Device instance with the specified identifier does not exist, the method will immediately return False without an error.
- `[string] newName`: The new name for the specified Device instance. If the string is greater than 128 characters, the method will return a descriptive error.
- `[string] newDescription`: The new description for the specified Device instance. If the string is greater than 128 character, the method will return a descriptive error.

```
bool UpdateDeviceLogsSettings(int deviceId, DeviceLogsSettings settings)
```

Description

Updates the log settings structure of the specified Device instance. This operation returns True only if it is completely successful. Otherwise, it returns False.

Required Permissions

Device: Update Device

Parameters

- `[int] DeviceId`: The identifier and primary key of the Device instance to be updated. If a Device instance with the specified identifier does not exist, the method will immediately return False without an error.
- `[DeviceLogsSettings] settings`: A DeviceLogsSettings structure with updated values for the specified device. If this parameter is set to Null, then the method will immediately return False without an error. A descriptive error will be returned if the specified `[Nullable<TimeSpan>] UploadTime` value is less than 00:00:00 or greater than 23:59:59.

```
bool ReassignDevices(int[] deviceIds, int newGroupId)
```

Description

Reassigns the specified Device instance(s) to the specified Group instance. This operation returns True only if it is completely successful. Otherwise, it returns False.

Required Permissions

Device: Change Target Group – Group: Remove Device, Add Device

Parameters

- `[int[]] deviceIds`: An array of identifiers corresponding to the Device instance(s) that should be reassigned. The number of passed items is limited to 100 by the server. Attempting to reassign more than the allowed number of objects will result in an error. Passing an array containing a Device `[int] Id` value that does not correspond to an existing Device instance will also result in an error. Passing an empty array will lead to an immediate False response without an error.
- `[int] newGroupId`: The identifier of the Group instance that the Device instance(s) should be reassigned to. If this value does not match an existing Group instance, then the Device instance(s) will be reassigned to the default “Unassigned” group.

`bool ForceDevicesReboot(int[] deviceIds)`

Description

Enables the flags indicating that the specified device(s) should reboot during the next check-in event. This operation returns True only if it is completely successful. Otherwise, it returns False.

Required Permissions

Device: Reboot Device

Parameters

- `[int[]] deviceIds`: An array of identifiers corresponding to the Device instance(s) that should be rebooted. The number of passed items is limited to 100 by the server. Attempting to reboot more than the allowed number of objects will result in an error. Passing an array containing a Device `[int] Id` value that does not correspond to an existing Device instance will also result in an error. Passing an empty array will lead to an immediate False response without an error.

`bool CancelDevicesReboot(int[] deviceIds)`

Description

Disables the flags indicating that the specified device(s) should reboot during the next check-in event. This operation returns True only if it is completely successful. Otherwise, it returns False.

Required Permissions

Device: Reboot Device

Parameters

- `[int[]] deviceIds`: An array of identifiers corresponding to the Device instance(s) that should be affected. The number of passed items is limited to 100 by the server. Attempting to target more than the allowed number of objects will result in an error. Passing an array containing a Device `[int] Id` value that does not correspond to an existing Device instance will also result in an error. Passing an empty array will lead to an immediate False response without an error.

`bool ForceDevicesRecovery(int[] deviceIds, bool reformat)`

Description

Enables the flags indicating that the specified device(s) should download and run a recovery script during the next check-in event. This operation returns True only if it is completely successful. Otherwise, it returns False.

Required Permissions

Device: Recover Device

Parameters

- `[int[]] deviceIds`: An array of identifiers corresponding to the Device instance(s) that should be recovered. The number of passed items is limited to 100 by the server. Attempting to recover more than the allowed number of objects will result in an error. Passing an array containing a Device `[int] Id` value that does not correspond to an existing Device instance will also result in an error. Passing an empty array will lead to an immediate False response without an error.
- `[bool] reformat`: A flag indicating whether the specified devices should also reformat their local storage.

`bool CancelDevicesRecovery(int[] deviceIds)`

Description

Disables the flags indicating that the specified device(s) should download and run a recovery script during the next check-in event. This operation only returns True if it is completely successful. Otherwise, it returns False.

Required Permissions

Device: Recover Device

Parameters

- `[int[]] deviceIds`: An array of identifiers corresponding to the Device instance(s) that should be affected. The number of passed items is limited to 100 by the server. Attempting to target more than the allowed number of objects will result in an error. Passing an array containing a Device `[int] Id` value that does not correspond to

an existing Device instance will also result in an error. Passing an empty array will lead to an immediate False response without an error.

`bool ForceDevicesLogsUpload(int[] deviceIds)`

Description

Enables the flags indicating that the specified device(s) should upload their current logs to the BrightSign Network servers during the next check-in event. This operation returns True only if it is completely successful. Otherwise, it returns False.

Required Permissions

Device: Force Logs Upload

Parameters

- `[int[]] deviceIds`: An array of identifiers corresponding to the Device instance(s) that should be affected. The number of passed items is limited to 100 by the server. Attempting to target more than the allowed number of objects will result in an error. Passing an array containing a Device `[int] Id` value that does not correspond to an existing Device instance will also result in an error. Passing an empty array will lead to an immediate False response without an error.

`bool CancelDevicesLogsUpload(int[] deviceIds)`

Description

Disables the flags indicating that the specified device(s) should upload their current logs to the BrightSign Network servers during the next check-in event. This operation returns True only if it is completely successful. Otherwise, it returns False.

Required Permissions

Device: Force Logs Upload

Parameters

- `[int[]] deviceIds`: An array of identifiers corresponding to the Device instance(s) that should be affected. The number of passed items is limited to 100 by the server. Attempting to target more than the allowed number of objects will result in an error. Passing an array containing a Device `[int] Id` value that does not correspond to an existing Device instance will also result in an error. Passing an empty array will lead to an immediate False response without an error.

```
PagedList<DeviceLogReport> GetDeviceLogReports(int deviceId, DeviceLogType logType,
string marker, int pageSize)
```

Description

Retrieves the next page of the DeviceLogReport list associated with the specified Device instance. The list is sorted chronologically by the `[DateTime] StartDate` of the log report. A device log, represented by a DeviceLogReport entity, can be downloaded from persistent storage using the `[string] FilePath` URL of the entity. Note that these URLs point to un-archived files that are in OpenXML format and may be up to 2GB in size each.

Required Permissions

Device: View Device Log Reports

Parameters

- `[int] DeviceId`: The identifier of the Device instance associated with the listed device logs. If a Device instance with the specified identifier does not exist, the server will return a Null response without error.
- `[DeviceLogType] logType`: The type of DeviceLogReport instances to be retrieved. Log types are represented with the DeviceLogType enumeration.
- `[string] marker`: The name of the last DeviceLogReport instance on the previous page. If the passed string is Null or empty, the first page of the list will be returned.
- `[int] pageSize`: The maximum number of objects returned by the method. If the list of objects that match the search criteria exceeds the `[int] pageSize` limit, the returned list will indicate that it is truncated. If the integer is

not positive, then the method will return the maximum allowed number of objects. Attempting to request more objects than is allowed will lead to the same result, but without an error.

`bool DeleteDevices(int[] deviceIds)`

Description

Deletes the specified Device instance(s) and releases the associated DeviceSubscription instance(s), allowing them to be reassigned. This operation returns True only if it is completely successful. Otherwise, it returns False.

Note: *If the physical device associated with a deleted Device instance reconnects to the BrightSign Network, the instance will be restored, but without the DeviceSubscription instance that was previously assigned to it.*

Required Permissions

Device: Delete Device

Parameters

- `[int[]] deviceIds`: An array of identifiers corresponding to the Device instance(s) that should be deleted. The number of passed items is limited to 100 by the server. Attempting to delete more than the allowed number of objects will result in an error. Passing an array containing a Device `[int] Id` value that does not correspond to an existing Device instance will also result in an error. Passing an empty array will lead to an immediate False response without an error.