Table of Contents

Namespace DLCToolkit

Classes

DLC

Main API for interacting with and loading DLC content from an external source. Also provides API's for retrieving DLC from various DRM (Digital Rights Management) services such as Steamworks, Google Play and more. DRM is used to ensure that the current user has access to or owns the requested DLC content if it is paid for or licensed in any way by the providing service.

DLCAsync

An awaitable object that is returned by async operations so you can wait for completion in a coroutine as well as access progress and status information. Used to wait until an async operation has been completed

DLCAsync<T>

An awaitable object that is returned by async operations so you can wait for completion in a coroutine as well as access progress and status information. Used to wait for an async operation to be completed with a result object.

DLCBatchAsync<T>

A batch async awaitable operation that contains multiple sub async operations. Can represent the progress and status of all operations combined and will wait until all operations have finished. Note that all sub operations will run in parallel in most cases, so a batch operation will only usually take as much time as the longest sub operation in most cases, plus some minor overhead for management.

DLCContent

Represents a DLC loaded in memory. Provides access to metadata, icons, plus API's for loading assets and scenes. Will remain in memory until Unload(bool), Dispose() or until the game exits.

DLCDirectory

A utility class for mounting a specific folder path as a designated DLC install folder. Contains useful helper methods to find and access the DLC contents included in the target folder. Note that only valid DLC files/information will be returned using this API (IsDLCFile(string) = true), even if the folder contains additional non-DLC files. There is also support for detecting when DLC files are added or removed from the target folder.

DLCNotAvailableException

An exception thrown when DLC is not available upon request.

DLCNotLoadedException

An exception thrown when DLC content is trying to be accessed but the DLC is not loaded.

DLCStreamProvider

Represents a data stream source containing DLC content.

Interfaces

IDLCAsyncProvider

A host that is able to manage the invocation of an async operation.

IDLCIconProvider

Provides access to various icon content that has been registered with the DLC content. Useful for displaying in UI's or splash screens to indicate that a certain DLC has been detected and activated.

IDLCMetadata

Access all metadata for a given DLC.

IDLCNameInfo

Access name information for a given DLC.

Enums

DLCContentFlags

The content flags that indicate which types of content are included in a given DLC.

DLCIconType

The built in icon size that we need to access.

Class DLC

Main API for interacting with and loading DLC content from an external source. Also provides API's for retrieving DLC from various DRM (Digital Rights Management) services such as Steamworks, Google Play and more. DRM is used to ensure that the current user has access to or owns the requested DLC content if it is paid for or licensed in any way by the providing service.

Inheritance

object

DLC

Namespace: DLCToolkit

Assembly: DLCToolkit.dll

Syntax

```
public static class DLC
```

Fields

ToolkitVersion

Get the current version of DLC Toolkit.

Declaration

```
public static readonly Version ToolkitVersion
```

Field Value

TYPE	DESCRIPTION
Version	

Properties

AllDLCContents

Get all DLC contents that are currently available. This could include contents that are currently loading or loaded.

Declaration

```
public static IEnumerable<DLCContent> AllDLCContents { get; }
```

Property Value

TYPE	DESCRIPTION
IEnumerable<DLCContent>	

DRMServiceProvider

Get the service that can provide a DRM container for the current build configuration.

Declaration

```
public static IDRMServiceProvider DRMServiceProvider { get; }
```

Property Value

TYPE	DESCRIPTION
IDRMServiceProvider	

IsScriptingSupported

Check if scripting is supported on this platform. Scripting is supported only on desktop platforms using the Mono backend.

Declaration

```
public static bool IsScriptingSupported { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

LoadedDLCContents

Get all DLC contents that are currently available with a loaded status. The DLC has been successfully loaded into memory.

Declaration

```
public static IEnumerable<DLCContent> LoadedDLCContents { get; }
```

Property Value

TYPE	DESCRIPTION
IEnumerable<DLCContent>	

LoadingDLCContents

Get all DLC contents that are currently available with a loading status. The DLC is currently in the process of being loaded into memory.

Declaration

```
public static IEnumerable<DLCContent> LoadingDLCContents { get; }
```

Property Value

TYPE	DESCRIPTION
IEnumerable<DLCContent>	

LocalDLCUniqueKeys

Try to get all DLC unique keys that are available locally. Local keys are simply DLC unique keys which were known about at the time of building the game (DLC profiles created before building the game). Note that only unique keys for enabled DLC profiles at the time of building the game will be available. For that reason the array will only list DLC contents that were created during development of the game, whether the DLC content was released or not. As a result it is highly recommended that you check with the current DRM provider for a true reflection of available DLC content using DLCUniqueKeysAsync (If the current platform has DRM support and the DRM provider can support listing unique contents). Alternatively you might use these local keys in combination with IsDLCAvailableAsync(IDLCAsyncProvider, string) to determine whether the DLC is usable (Exists via DRM) and is available (Installed locally).

Declaration

```
public static string[] LocalDLCUniqueKeys { get; }
```

Property Value

TYPE	DESCRIPTION
string[]	

RemoteDLCUniqueKeysAsync

Try to get all DLC unique keys that are available remotely. Remote keys are simply DLC unique keys which have been published to a DRM provider such as Steamworks. Some DRM providers may not support listing DLC contents that are published remotely. Note that only published unique keys will be returned here if the DRM provider supports listing available DLC contents. Note also that all DLC unique keys will be listed here even if the user does not own or subscribed to the downloadable content. For that reason you should use IsDLCAvailableAsync(IDLCAsyncProvider, string) to determine whether the DLC is usable (Exists via DRM) and is available (Installed locally).

Declaration

```
public static DLCAsync<string[]> RemoteDLCUniqueKeysAsync { get; }
```

Property Value

TYPE	DESCRIPTION
DLCAsync<string[]>	

Exceptions

TYPE	CONDITION
NotSupportedException	DRM provider does not support listing published DLC unique keys

Methods

GetDLC(string)

Try to get the DLC with the specified key if it has already started loading or has been loaded.

Declaration

```
public static DLCContent GetDLC(string uniqueKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key for the DLC

Returns

TYPE	DESCRIPTION
DLCContent	The DLC that is already loading or loaded with the provided unique key, or null if the DLC was not found

GetDLC(string, Version)

Try to get the DLC loaded with the specified name and optional version if it has already started loading or has been loaded.

Declaration

```
public static DLCContent GetDLC(string name, Version version = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	The name of the DLC
Version	version	The optional version of the DLC if an explicit match is required

Returns

TYPE	DESCRIPTION
DLCContent	The DLC that is already loading or loaded from the provided path, or null if the DLC was not found

GetDLCFrom(string)

Try to get the DLC loaded from the specified path if it has already started loading or has been loaded.

Declaration

```
public static DLCContent GetDLCFrom(string path)
```

Parameters

TYPE	NAME	DESCRIPTION
string	path	The path of the DLC

Returns

TYPE	DESCRIPTION
DLCContent	The DLC that is already loading or loaded from the provided path, or null if the DLC was not found

GetLoadedDLC(string)

Try to get the DLC with the specified key if it has already been loaded. Note this will only detect successfully loaded DLC's and not DLC's that are currently loading.

Declaration

```
public static DLCContent GetLoadedDLC(string uniqueKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key for the DLC

Returns

TYPE	DESCRIPTION
DLCContent	The DLC that is already loaded with the provided unique key, or null if the DLC was not found

GetLoadedDLC(string, Version)

Try to get the DLC loaded with the specified name and optional version if it has already been loaded. Note this will only detect successfully loaded DLC's and not DLC's that are currently loading.

Declaration

```
public static DLCContent GetLoadedDLC(string name, Version version = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	The name of the DLC
Version	version	The optional version of the DLC if an explicit match is required

Returns

TYPE	DESCRIPTION
DLCContent	The DLC that is already loaded from the provided name and optional version, or null if the DLC was not found

GetLoadingDLC(string)

Try to get the DLC currently loading with the specified key. Note this will only detect successfully loaded DLC's and not DLC's that are currently loaded.

Declaration

```
public static DLCContent GetLoadingDLC(string uniqueKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key for the DLC

Returns

TYPE	DESCRIPTION
DLCContent	The DLC that is loading with the provided unique key, or null if the DLC was not found

GetLoadingDLC(string, Version)

Try to get the DLC currently loading with the specified name and optional version. Note this will only detect DLC's that are currently in the process of being loaded and not DLC's that are currently loaded.

Declaration

```
public static DLCContent GetLoadingDLC(string name, Version version = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	The name of the DLC
Version	version	The optional version of the DLC if an explicit match is required

Returns

TYPE	DESCRIPTION
DLCContent	The DLC that is loading with the provided name and optional version, or null if the DLC was not found

IsAvailable(string)

Check if the specified DLC is purchased and installed. Some providers may need to make a web request to check for purchased DLC, so this operations must be async.

Declaration

```
public static DLCAsync<bool> IsAvailable(string uniqueKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key for the dlc

Returns

TYPE	DESCRIPTION
DLCAsync<bool>	True if the dlc is installed or false if not

Exceptions

TYPE	CONDITION
NotSupportedException	No DRM provider for ths current platform

IsDLCFile(string)

Check if the specified file path is a valid DLC file format. This is intended to be a very quick check and will only load a few bytes from the source file in order to determine validity.

Declaration

```
public static bool IsDLCFile(string path)
```

Parameters

TYPE	NAME	DESCRIPTION
string	path	The path of the file to check

Returns

TYPE	DESCRIPTION
bool	True if the file is a valid DLC format which can be loaded, or false if not

IsDLCLoaded(string)

Check if the DLC with the specified unique key is currently loaded.

Declaration

```
public static bool IsDLCLoaded(string uniqueKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key for the DLC

Returns

TYPE	DESCRIPTION
bool	True if the DLC is loaded or false if not

IsDLCLoaded(string, Version)

Check if the DLC with the specified name and optional version is currently loaded. Note that this method will check the DLC name/version and not the DLC unique key which can be checked by IsDLCLoaded(string) instead.

Declaration

```
public static bool IsDLCLoaded(string name, Version version = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	The name of the DLC
Version	version	An optional version if you want to find a specific version of a loaded DLC

Returns

TYPE	DESCRIPTION
bool	True if the DLC is loaded or false if not

IsDLCLoading(string)

Check if the DLC with the specified unique key is currently being loaded. Note that this method is only of use when DLC async loading is used.

Declaration

```
public static bool IsDLCLoading(string uniqueKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key for the DLC

Returns

TYPE	DESCRIPTION
bool	True if the DLC is loading or false if not

IsDLCLoading(string, Version)

Check if the DLC with the specified name and optional version is currently being loaded. Note that this method will check the DLC name/version and not the DLC unique key which can be checked by IsDLCLoading(string) instead. Note that this method is only of use when DLC async loading is used.

Declaration

```
public static bool IsDLCLoading(string name, Version version = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	The name of the DLC

TYPE	NAME	DESCRIPTION
Version	version	An optional version if you want to find a specific version of a loading DLC

Returns

TYPE	DESCRIPTION
bool	True if the DLC is loading or false if not

LoadDLC(DLCStreamProvider)

Attempt to load DLC content with the specified unique key. Note that the unique key can be different between platforms and you should check the DLC profile to ensure that the correct key is used for the current platform. Alternatively you may be able to query IDRMProvider.DLCUniqueKeys to enumerate all available DLC's, but note that some DRM providers may not implement that property or only partially implement it (May not return all possible DLC's). The DLC will need to be available (Owned and installed) from the current DRM provider in order to succeed. The DLC will be loaded into memory and data may be preloaded according to the preload options set at build time.

Declaration

```
public static DLCContent LoadDLC(DLCStreamProvider streamProvider)
```

Parameters

TYPE	NAME	DESCRIPTION
DLCStreamProvider	streamProvider	The stream provider for the DLC content

Returns

TYPE	DESCRIPTION
DLCContent	A DLCContent containing the loaded DLC or null if the DLC could not be loaded

Exceptions

TYPE	CONDITION
ArgumentNullException	The stream provider is null
NotSupportedException	There is not suitable DRM provider for this platform
TimeoutException	A DRM request timed out and the operation is aborted to avoid infinite waiting or freezing the application
DLCNotAvailableException	The requested DLC is owned but is not currently installed or available locally. You may need to request that the DLC is installed using RequestInstallDLCAsync(IDLCAsyncProvider, string)

TYPE	CONDITION
DirectoryNotFoundException	Part of the file path could not be found
FileNotFoundException	The specified file path does not exist
FormatException	The specified file is not a valid DLC format
InvalidDataException	The specified file has been signed by another game or version - The DLC does not belong to this game
InvalidOperationException	The DLC file is missing required data or is possibly corrupt

LoadDLC(string)

Attempt to load DLC content with the specified unique key. Note that the unique key can be different between platforms and you should check the DLC profile to ensure that the correct key is used for the current platform. Alternatively you may be able to query IDRMProvider.DLCUniqueKeys to enumerate all available DLC's, but note that some DRM providers may not implement that property or only partially implement it (May not return all possible DLC's). The DLC will need to be available (Owned and installed) from the current DRM provider in order to succeed. The DLC will be loaded into memory and data may be preloaded according to the preload options set at build time.

Declaration

```
public static DLCContent LoadDLC(string uniqueKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key for the DLC content

Returns

TYPE	DESCRIPTION
DLCContent	A DLCContent containing the loaded DLC or null if the DLC could not be loaded

Exceptions

TYPE	CONDITION
ArgumentException	The unique key is null or empty
NotSupportedException	There is not suitable DRM provider for this platform

TYPE	CONDITION
TimeoutException	A DRM request timed out and the operation is aborted to avoid infinite waiting or freezing the application
DLCNotAvailableException	The requested DLC is owned but is not currently installed or available locally. You may need to request that the DLC is installed using RequestInstallDLCAsync(IDLCAsyncProvider, string)
DirectoryNotFoundException	Part of the file path could not be found
FileNotFoundException	The specified file path does not exist
FormatException	The specified file is not a valid DLC format
InvalidDataException	The specified file has been signed by another game or version - The DLC does not belong to this game
InvalidOperationException	The DLC file is missing required data or is possibly corrupt

LoadDLCAsync(DLCStreamProvider)

Attempt to load DLC content from the specified stream provider asynchronously. If the target DLC is already being loaded or has been loaded, this method will simply return the current load operation or a completed operation with the already loaded DLCContent. Note that the unique key can be different between platforms and you should check the DLC profile to ensure that the correct key is used for the current platform. Alternatively you may be able to query IDRMProvider.DLCUniqueKeys to enumerate all available DLC's, but note that some DRM providers may not implement that property or only partially implement it (May not return all possible DLC's). The DLC will be loaded on the background thread so it is possible to continue with gameplay or show an animated loading screen. The DLC will be loaded into memory and data may be preloaded according to the preload options set at build time. Note that a DRM provider is required for the current platform, otherwise you should use LoadDLCFromAsync(string).

Declaration

```
public static DLCAsync<DLCContent> LoadDLCAsync(DLCStreamProvider streamProvider)
```

Parameters

TYPE	NAME	DESCRIPTION
DLCStreamProvider	streamProvider	The stream provider for the DLC content

Returns

TYPE	DESCRIPTION
DLCAsync<DLCContent>	A DLCContent containing the loaded DLC or null if the DLC could not be loaded

Exceptions

TYPE	CONDITION
ArgumentNullException	The stream provider is null
NotSupportedException	There is no suitable DRM provider for this platform
DirectoryNotFoundException	Part of the file path could not be found
FileNotFoundException	The specified file path does not exist
FormatException	The specified file is not a valid DLC format
InvalidDataException	The specified file has been signed by another game or version - The DLC does not belong to this game
InvalidOperationException	The DLC file is missing required data or is possibly corrupt

LoadDLCAsync(string, bool)

Attempt to load DLC content with the specified unique key asynchronously. If the target DLC is already being loaded or has been loaded, this method will simply return the current load operation or a completed operation with the already loaded DLCContent. Note that the unique key can be different between platforms and you should check the DLC profile to ensure that the correct key is used for the current platform. Alternatively you may be able to query IDRMProvider.DLCUniqueKeys to enumerate all available DLC's, but note that some DRM providers may not implement that property or only partially implement it (May not return all possible DLC's). The DLC will be loaded on the background thread so it is possible to continue with gameplay or show an animated loading screen. The DLC will need to be available (Owned and installed, or owned with `installOnDemand` enabled) from the current DRM provider in order to succeed. The DLC will be loaded into memory and data may be preloaded according to the preload options set at build time. Note that a DRM provider is required for the current platform, otherwise you should use LoadDLCFromAsync(string).

Declaration

```
public static DLCAsync<DLCContent> LoadDLCAsync(string uniqueKey, bool installOnDemand = false)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key for the DLC content
bool	installOnDemand	Should the DLC be installed if it is owned by the used but not available locally. Only available in async mode

Returns

TYPE	DESCRIPTION
DLCAsync<DLCContent>	A DLCContent containing the loaded DLC or null if the DLC could not be loaded

Exceptions

TYPE	CONDITION
ArgumentException	The unique key is null or empty
NotSupportedException	There is no suitable DRM provider for this platform
DirectoryNotFoundException	Part of the file path could not be found
FileNotFoundException	The specified file path does not exist
FormatException	The specified file is not a valid DLC format
InvalidDataException	The specified file has been signed by another game or version - The DLC does not belong to this game
InvalidOperationException	The DLC file is missing required data or is possibly corrupt

LoadDLCBatchAsync(string[], bool)

Attempt to load multiple DLC content simultaneously with the specified unique keys asynchronously. Note that the unique key can be different between platforms and you should check the DLC profile to ensure that the correct key is used for the current platform. Alternatively you may be able to query IDRMProvider.DLCUniqueKeys to enumerate all available DLC's, but note that some DRM providers may not implement that property or only partially implement it (May not return all possible DLC's). The DLC will be loaded on the background thread so it is possible to continue with gameplay or show an animated loading screen. The DLC will need to be available (Owned and installed, or owned with `installOnDemand` enabled) from the current DRM provider in order to succeed. The DLC will be loaded into memory and data may be preloaded according to the preload options set at build time. Note that a DRM provider is required for the current platform, otherwise you should use LoadDLCFromAsync(string). Note that this method will suppress any exceptions and report them in the resulting async operation if caught.

Declaration

```
public static DLCBatchAsync<DLCContent> LoadDLCBatchAsync(string[] uniqueKeys, bool installOnDemand)
```

Parameters

TYPE	NAME	DESCRIPTION
string[]	uniqueKeys	An array of unique keys for all DLC content to load

TYPE	NAME	DESCRIPTION
bool	installOnDemand	Should the DLC be installed if it is owned by the user but not available locally. Only available in async mode

Returns

TYPE	DESCRIPTION
DLCBatchAsync<DLCContent>	A batch async operation which can report progress for the combined load operation as a whole, or can provide access to each inner load request

Exceptions

TYPE	CONDITION
ArgumentNullException	The unique keys are empty

LoadDLCBatchFromAsync(string[])

Attempt to load multiple DLC content simultaneously from the paths asynchronously. The DLC will be loaded on the background thread so it is possible to continue with gameplay or show an animated loading screen. The DLC will be loaded into memory and data may be preloaded according to the preload options set at build time. Note that a DRM provider is required for the current platform, otherwise you should use LoadDLCFromAsync(string). Note that this method will suppress any exceptions and report them in the resulting async operation if caught.

Declaration

```
public static DLCBatchAsync<DLCContent> LoadDLCBatchFromAsync(string[] paths)
```

Parameters

TYPE	NAME	DESCRIPTION
string[]	paths	An array of paths for all DLC content to load

Returns

TYPE	DESCRIPTION
DLCBatchAsync<DLCContent>	A batch async operation which can report progress for the combined load operation as a whole, or can provide access to each inner load request

Exceptions

TYPE	CONDITION
ArgumentNullException	The paths are empty

LoadDLCFrom(string)

Attempt to load DLC content from the specified file path. The DLC will be loaded into memory and data may be preloaded

according to the preload options set at build time.

Declaration

```
public static DLCContent LoadDLCFrom(string path)
```

Parameters

TYPE	NAME	DESCRIPTION
string	path	The file path containing the DLC content

Returns

TYPE	DESCRIPTION
DLCContent	A DLCContent containing the loaded DLC

Exceptions

TYPE	CONDITION
ArgumentException	The path is null or empty
DirectoryNotFoundException	Part of the file path could not be found
FileNotFoundException	The specified file path does not exist
FormatException	The specified file is not a valid DLC format
InvalidDataException	The specified file has been signed by another game or version - The DLC does not belong to this game
InvalidOperationException	The DLC file is missing required data or is possibly corrupt

LoadDLCFromAsync(string)

Attempt to load DLC content from the specified file path asynchronously. The DLC will be loaded into memory and data may be preloaded according to the preload options set at build time.

Declaration

```
public static DLCAsync<DLCContent> LoadDLCFromAsync(string path)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
string	path	The file path containing the DLC content

Returns

TYPE	DESCRIPTION
DLCAsync<DLCContent>	A DLCAsync operation that can be awaited and provides access to the loaded DLC

Exceptions

TYPE	CONDITION
ArgumentException	The path is null or empty
DirectoryNotFoundException	Part of the file path could not be found
FileNotFoundException	The specified file path does not exist
FormatException	The specified file is not a valid DLC format
InvalidDataException	The specified file has been signed by another game or version - The DLC does not belong to this game
InvalidOperationException	The DLC file is missing required data or is possibly corrupt

LoadDLCMetadata(DLCStreamProvider)

Attempt to load DLC content with the specified unique key. This is a reduced load mode and it will only be possible to access the IDLCMetadata for the DLC and nothing more. Designed to be a quick operation for accessing metadata, but may take some time to complete depending upon DRM provider and availability. Use LoadDLCAsync(string, bool) if you need to load assets from the DLC. Note that the unique key can be different between platforms and you should check the DLC profile to ensure that the correct key is used for the current platform. Alternatively you may be able to query IDRMProvider.DLCUniqueKeys to enumerate all available DLC's, but note that some DRM providers may not implement that property or only partially implement it (May not return all possible DLC's). The DLC will need to be available (Owned and installed) from the current DRM provider in order to succeed. The DLC will be loaded into memory and data may be preloaded according to the preload options set at build time.

Declaration

```
public static IDLCMetadata LoadDLCMetadata(DLCStreamProvider streamProvider)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
DLCStreamProvider	streamProvider	The stream provider for the DLC content

Returns

TYPE	DESCRIPTION
IDLCMetadata	A DLCContent containing the loaded DLC or null if the DLC could not be loaded

Exceptions

TYPE	CONDITION
ArgumentNullException	The stream provider is null
NotSupportedException	There is not suitable DRM provider for this platform
TimeoutException	A DRM request timed out and the operation is aborted to avoid infinite waiting or freezing the application
DLCNotAvailableException	The requested DLC is owned but is not currently installed or available locally. You may need to request that the DLC is installed using RequestInstallDLCAsync(IDLCAsyncProvider, string)
DirectoryNotFoundException	Part of the file path could not be found
FileNotFoundException	The specified file path does not exist
FormatException	The specified file is not a valid DLC format
InvalidDataException	The specified file has been signed by another game or version - The DLC does not belong to this game
InvalidOperationException	The DLC file is missing required data or is possibly corrupt

LoadDLCMetadata(string)

Attempt to load DLC content with the specified unique key. This is a reduced load mode and it will only be possible to access the IDLCMetadata for the DLC and nothing more. Designed to be a quick operation for accessing metadata, but may take some time to complete depending upon DRM provider and availability. Use LoadDLCAsync(string, bool) if you need to load assets from the DLC. Note that the unique key can be different between platforms and you should check the DLC profile to ensure that the correct key is used for the current platform. Alternatively you may be able to query IDRMProvider.DLCUniqueKeys to enumerate all available DLC's, but note that some DRM providers may not implement that property or only partially implement it (May not return all possible DLC's). The DLC will need to be available (Owned and installed) from the current DRM provider in order to

succeed. The DLC will be loaded into memory and data may be preloaded according to the preload options set at build time.

Declaration

```
public static IDLCMetadata LoadDLCMetadata(string uniqueKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key for the DLC content

Returns

TYPE	DESCRIPTION
IDLCMetadata	A DLCContent containing the loaded DLC or null if the DLC could not be loaded

Exceptions

TYPE	CONDITION
ArgumentException	The unique key is null or empty
NotSupportedException	There is not suitable DRM provider for this platform
TimeoutException	A DRM request timed out and the operation is aborted to avoid infinite waiting or freezing the application
DLCNotAvailableException	The requested DLC is owned but is not currently installed or available locally. You may need to request that the DLC is installed using RequestInstallDLCAsync(IDLCAsyncProvider, string)
DirectoryNotFoundException	Part of the file path could not be found
FileNotFoundException	The specified file path does not exist
FormatException	The specified file is not a valid DLC format
InvalidDataException	The specified file has been signed by another game or version - The DLC does not belong to this game
InvalidOperationException	The DLC file is missing required data or is possibly corrupt

LoadDLCMetadataAsync(DLCStreamProvider)

Attempt to load DLC metadata only from the specified stream provider asynchronously in metadata. This is a reduced load mode

and it will only be possible to access the IDLCMetadata for the DLC and nothing more. Designed to be a quick operation for accessing metadata, but may take some time to complete depending upon DRM provider and availability. Use LoadDLCAsync(string, bool) if you need to load assets from the DLC. Note that the unique key can be different between platforms and you should check the DLC profile to ensure that the correct key is used for the current platform. Alternatively you may be able to query IDRMProvider.DLCUniqueKeys to enumerate all available DLC's, but note that some DRM providers may not implement that property or only partially implement it (May not return all possible DLC's). The DLC metadata will be loaded on the background thread so it is possible to continue with gameplay or show an animated loading screen. The DLC will be loaded into memory and data may be preloaded according to the preload options set at build time. Note that a DRM provider is required for the current platform, otherwise you should use LoadDLCMetadataFrom(string).

Declaration

```
public static DLCAsync<IDLCMetadata> LoadDLCMetadataAsync(DLCStreamProvider streamProvider)
```

Parameters

TYPE	NAME	DESCRIPTION
DLCStreamProvider	streamProvider	The stream provider for the DLC content

Returns

TYPE	DESCRIPTION
DLCAsync<IDLCMetadata>	A DLCContent containing the loaded DLC or null if the DLC could not be loaded

Exceptions

TYPE	CONDITION
ArgumentNullException	The stream provider is null
NotSupportedException	There is no suitable DRM provider for this platform
DirectoryNotFoundException	Part of the file path could not be found
FileNotFoundException	The specified file path does not exist
FormatException	The specified file is not a valid DLC format
InvalidDataException	The specified file has been signed by another game or version - The DLC does not belong to this game
InvalidOperationException	The DLC file is missing required data or is possibly corrupt

LoadDLCMetadataAsync(string, bool)

Attempt to load DLC metadata only with the specified unique key asynchronously in metadata with assets mode. This is a reduced load mode and it will only be possible to access the IDLCMetadata for the DLC and nothing more. Designed to be a quick operation for accessing metadata, but may take some time to complete depending upon DRM provider and availability. Use LoadDLCAsync(string, bool) if you need to load assets from the DLC. Note that the unique key can be different between platforms and you should check the DLC profile to ensure that the correct key is used for the current platform. Alternatively you may be able to query IDRMProvider.DLCUniqueKeys to enumerate all available DLC's, but note that some DRM providers may not implement that property or only partially implement it (May not return all possible DLC's). The DLC metadata will be loaded on the background thread so it is possible to continue with gameplay or show an animated loading screen. The DLC will need to be available (Owned and installed, or owned with `installOnDemand` enabled) from the current DRM provider in order to succeed. The DLC will be loaded into memory and data may be preloaded according to the preload options set at build time. Note that a DRM provider is required for the current platform, otherwise you should use LoadDLCMetadataFrom(string).

Declaration

```
public static DLCAsync<IDLCMetadata> LoadDLCMetadataAsync(string uniqueKey, bool installOnDemand = false)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key for the DLC content
bool	installOnDemand	Should the DLC be installed if it is owned by the used but not available locally. Only available in async mode

Returns

TYPE	DESCRIPTION
DLCAsync<IDLCMetadata>	A DLCContent containing the loaded DLC or null if the DLC could not be loaded

Exceptions

TYPE	CONDITION
ArgumentException	The unique key is null or empty
NotSupportedException	There is no suitable DRM provider for this platform
DirectoryNotFoundException	Part of the file path could not be found
FileNotFoundException	The specified file path does not exist
FormatException	The specified file is not a valid DLC format

TYPE	CONDITION
InvalidDataException	The specified file has been signed by another game or version - The DLC does not belong to this game
InvalidOperationException	The DLC file is missing required data or is possibly corrupt

LoadDLCMetadataFrom(string)

Attempt to load DLC metadata only from the specified file path. Intended to be very quick access and will only load the absolute minimum amount of data required to access the meta information for the DLC. Only the DLC metadata will be loaded into memory.

Declaration

```
public static IDLCMetadata LoadDLCMetadataFrom(string path)
```

Parameters

TYPE	NAME	DESCRIPTION
string	path	The file path containing the DLC content

Returns

TYPE	DESCRIPTION
IDLCMetadata	A IDLCMetadata containing the loaded DLC metadata

Exceptions

TYPE	CONDITION
ArgumentException	The path is null or empty
DirectoryNotFoundException	Part of the file path could not be found
FileNotFoundException	The specified file path does not exist
FormatException	The specified file is not a valid DLC format
InvalidDataException	The specified file has been signed by another game or version - The DLC does not belong to this game
InvalidOperationException	The DLC file is missing required data or is possibly corrupt

LoadDLCMetadataFromAsync(string)

Attempt to load DLC metadata only from the specified file path asynchronously. Intended to be very quick access and will only load the absolute minimum amount of data required to access the meta information for the DLC. Only the DLC metadata will be loaded into memory.

Declaration

```
public static DLCAsync<IDLCMetadata> LoadDLCMetadataFromAsync(string path)
```

Parameters

TYPE	NAME	DESCRIPTION
string	path	The file path containing the DLC content

Returns

TYPE	DESCRIPTION
DLCAsync<IDLCMetadata>	A DLCAsync operation that can be awaited and provides access to the DLC metadata

Exceptions

TYPE	CONDITION
ArgumentException	The path is null or empty
DirectoryNotFoundException	Part of the file path could not be found
FileNotFoundException	The specified file path does not exist
FormatException	The specified file is not a valid DLC format
InvalidDataException	The specified file has been signed by another game or version - The DLC does not belong to this game
InvalidOperationException	The DLC file is missing required data or is possibly corrupt

LoadDLCMetadataWithAssets(DLCStreamProvider)

Attempt to load DLC content with the specified unique key. This is a reduced load mode and it will be possible to access extra metadata for assets and scenes, but it will not be possible to load any asset or scene content into the game. Note that the unique key can be different between platforms and you should check the DLC profile to ensure that the correct key is used for the current platform. Alternatively you may be able to query IDRMProvider.DLCUniqueKeys to enumerate all available DLC's, but note that some DRM providers may not implement that property or only partially implement it (May not return all possible DLC's). The DLC will need to be available (Owned and installed) from the current DRM provider in order to succeed. The DLC will be loaded into memory and data may be preloaded according to the preload options set at build time.

Declaration

```
public static DLCContent LoadDLCMetadataWithAssets(DLCStreamProvider streamProvider)
```

Parameters

TYPE	NAME	DESCRIPTION
DLCStreamProvider	streamProvider	The stream provider for the DLC content

Returns

TYPE	DESCRIPTION
DLCContent	A DLCContent containing the loaded DLC or null if the DLC could not be loaded

Exceptions

TYPE	CONDITION
ArgumentNullException	The stream provider is null
NotSupportedException	There is not suitable DRM provider for this platform
TimeoutException	A DRM request timed out and the operation is aborted to avoid infinite waiting or freezing the application
DLCNotAvailableException	The requested DLC is owned but is not currently installed or available locally. You may need to request that the DLC is installed using RequestInstallDLCAsync(IDLCAsyncProvider, string)
DirectoryNotFoundException	Part of the file path could not be found
FileNotFoundException	The specified file path does not exist
FormatException	The specified file is not a valid DLC format
InvalidDataException	The specified file has been signed by another game or version - The DLC does not belong to this game
InvalidOperationException	The DLC file is missing required data or is possibly corrupt

LoadDLCMetadataWithAssets(string)

Attempt to load DLC content with the specified unique key. This is a reduced load mode and it will be possible to access extra metadata for assets and scenes, but it will not be possible to load any asset or scene content into the game. Note that the unique key can be different between platforms and you should check the DLC profile to ensure that the correct key is used for the current

platform. Alternatively you may be able to query IDRMProvider.DLCUniqueKeys to enumerate all available DLC's, but note that some DRM providers may not implement that property or only partially implement it (May not return all possible DLC's). The DLC will need to be available (Owned and installed) from the current DRM provider in order to succeed. The DLC will be loaded into memory and data may be preloaded according to the preload options set at build time.

Declaration

```
public static DLCContent LoadDLCMetadataWithAssets(string uniqueKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key for the DLC content

Returns

TYPE	DESCRIPTION
DLCContent	A DLCContent containing the loaded DLC or null if the DLC could not be loaded

Exceptions

TYPE	CONDITION
ArgumentException	The unique key is null or empty
NotSupportedException	There is not suitable DRM provider for this platform
TimeoutException	A DRM request timed out and the operation is aborted to avoid infinite waiting or freezing the application
DLCNotAvailableException	The requested DLC is owned but is not currently installed or available locally. You may need to request that the DLC is installed using RequestInstallDLCAsync(IDLCAsyncProvider, string)
DirectoryNotFoundException	Part of the file path could not be found
FileNotFoundException	The specified file path does not exist
FormatException	The specified file is not a valid DLC format
InvalidDataException	The specified file has been signed by another game or version - The DLC does not belong to this game
InvalidOperationException	The DLC file is missing required data or is possibly corrupt

LoadDLCMetadataWithAssetsAsync(DLCStreamProvider)

Attempt to load DLC content from the specified stream provider asynchronously in metadata with assets mode. This is a reduced load mode and it will be possible to access extra metadata for assets and scenes, but it will not be possible to load any asset or scene content into the game. Use LoadDLCAsync(string, bool) if you need to load assets from the DLC. Note that the unique key can be different between platforms and you should check the DLC profile to ensure that the correct key is used for the current platform. Alternatively you may be able to query IDRMProvider.DLCUniqueKeys to enumerate all available DLC's, but note that some DRM providers may not implement that property or only partially implement it (May not return all possible DLC's). The DLC will be loaded on the background thread so it is possible to continue with gameplay or show an animated loading screen. The DLC will be loaded into memory and data may be preloaded according to the preload options set at build time. Note that a DRM provider is required for the current platform, otherwise you should use LoadDLCMetadataWithAssetsFrom(string).

Declaration

```
public static DLCAsync<DLCContent> LoadDLCMetadataWithAssetsAsync(DLCStreamProvider streamProvider)
```

Parameters

TYPE	NAME	DESCRIPTION
DLCStreamProvider	streamProvider	The stream provider for the DLC content

Returns

TYPE	DESCRIPTION
DLCAsync<DLCContent>	A DLCContent containing the loaded DLC or null if the DLC could not be loaded

Exceptions

TYPE	CONDITION
ArgumentNullException	The stream provider is null
NotSupportedException	There is no suitable DRM provider for this platform
DirectoryNotFoundException	Part of the file path could not be found
FileNotFoundException	The specified file path does not exist
FormatException	The specified file is not a valid DLC format
InvalidDataException	The specified file has been signed by another game or version - The DLC does not belong to this game
InvalidOperationException	The DLC file is missing required data or is possibly corrupt

LoadDLCMetadataWithAssetsAsync(string, bool)

Attempt to load DLC content with the specified unique key asynchronously in metadata with assets mode. This is a reduced load mode and it will be possible to access extra metadata for assets and scenes, but it will not be possible to load any asset or scene content into the game. Use LoadDLCAsync(string, bool) if you need to load assets from the DLC. Note that the unique key can be different between platforms and you should check the DLC profile to ensure that the correct key is used for the current platform. Alternatively you may be able to query IDRMProvider.DLCUniqueKeys to enumerate all available DLC's, but note that some DRM providers may not implement that property or only partially implement it (May not return all possible DLC's). The DLC will be loaded on the background thread so it is possible to continue with gameplay or show an animated loading screen. The DLC will need to be available (Owned and installed, or owned with `installOnDemand` enabled) from the current DRM provider in order to succeed. The DLC will be loaded into memory and data may be preloaded according to the preload options set at build time. Note that a DRM provider is required for the current platform, otherwise you should use LoadDLCMetadataWithAssetsFrom(string).

Declaration

```
public static DLCAsync<DLCContent> LoadDLCMetadataWithAssetsAsync(string uniqueKey, bool installOnDemand = false)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key for the DLC content
bool	installOnDemand	Should the DLC be installed if it is owned by the used but not available locally. Only available in async mode

Returns

TYPE	DESCRIPTION
DLCAsync<DLCContent>	A DLCContent containing the loaded DLC or null if the DLC could not be loaded

Exceptions

TYPE	CONDITION
ArgumentException	The unique key is null or empty
NotSupportedException	There is no suitable DRM provider for this platform
DirectoryNotFoundException	Part of the file path could not be found
FileNotFoundException	The specified file path does not exist
FormatException	The specified file is not a valid DLC format

TYPE	CONDITION
InvalidDataException	The specified file has been signed by another game or version - The DLC does not belong to this game
InvalidOperationException	The DLC file is missing required data or is possibly corrupt

LoadDLCMetadataWithAssetsFrom(string)

Attempt to load DLC metadata with assets only from the specified file path. Intended to be very quick access and will only load the absolute minimum amount of data required to access the meta information for the DLC. Only the DLC metadata and asset metadata will be loaded into memory (Assets, scenes and scripts will not be loadable). SharedAssets and SceneAssets will be accessible after successful load to discover meta information about included assets.

Declaration

```
public static DLCContent LoadDLCMetadataWithAssetsFrom(string path)
```

Parameters

TYPE	NAME	DESCRIPTION
string	path	The file path containing the DLC content

Returns

TYPE	DESCRIPTION
DLCContent	A DLCContent containing the loaded DLC

Exceptions

TYPE	CONDITION
ArgumentException	The path is null or empty
DirectoryNotFoundException	Part of the file path could not be found
FileNotFoundException	The specified file path does not exist
FormatException	The specified file is not a valid DLC format
InvalidDataException	The specified file has been signed by another game or version - The DLC does not belong to this game
InvalidOperationException	The DLC file is missing required data or is possibly corrupt

LoadDLCMetadataWithAssetsFromAsync(string)

Attempt to load DLC metadata with assets only from the specified file path asynchronously. Intended to be very quick access and will only load the absolute minimum amount of data required to access the meta information for the DLC. Only the DLC metadata and asset metadata will be loaded into memory (Assets, scenes and scripts will not be loadable). SharedAssets and SceneAssets will be accessible after successful load to discover meta information only about included assets.

Declaration

```
public static DLCAsync<DLCContent> LoadDLCMetadataWithAssetsFromAsync(string path)
```

Parameters

TYPE	NAME	DESCRIPTION
string	path	The file path containing the DLC content

Returns

TYPE	DESCRIPTION
DLCAsync<DLCContent>	A DLCAsync operation that can be awaited and provides access to the loaded DLC

Exceptions

TYPE	CONDITION
ArgumentException	The path is null or empty
DirectoryNotFoundException	Part of the file path could not be found
FileNotFoundException	The specified file path does not exist
FormatException	The specified file is not a valid DLC format
InvalidDataException	The specified file has been signed by another game or version - The DLC does not belong to this game
InvalidOperationException	The DLC file is missing required data or is possibly corrupt

RegisterDRMServiceProvider(IDRMServiceProvider)

Register a custom IDRMServiceProvider which is responsible for providing the correct DRM management for the current build configuration.

Declaration

```
public static void RegisterDRMServiceProvider(IDRMServiceProvider serviceProvider)
```

Parameters

TYPE	NAME	DESCRIPTION
IDRMServiceProvider	serviceProvider	The custom service provider or null, in which case the default DRM service provider will be used

RequestInstall(string)

Request that the dlc with the provided unique key is installed onto the system if it is available to the user.

Declaration

```
public static DLCAsync RequestInstall(string uniqueKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The async provider to allow async tasks to be started

Returns

TYPE	DESCRIPTION
DLCAsync	The unique key of the dlc

Exceptions

TYPE	CONDITION
NotSupportedException	No DRM provider for ths current platform

RequestUninstall(string)

Request that the dlc with the provided unique key is uninstalled from the system if it is currently installed.

Declaration

```
public static void RequestUninstall(string uniqueKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key of the dlc

Exceptions

TYPE	CONDITION
NotSupportedException	No DRM provider for ths current platform

Class DLCAsync

An awaitable object that is returned by async operations so you can wait for completion in a coroutine as well as access progress and status information. Used to wait until an async operation has been completed

Inheritance

object

DLCAsync

DLCAsync<T>

Implements

IEnumerator

Namespace: DLCToolkit

Assembly: DLCToolkit.dll

Syntax

```
public class DLCAsync : IEnumerator
```

Fields

isSuccessful

Was the operation successful or did something go wrong.

Declaration

```
protected bool isSuccessful
```

Field Value

TYPE	DESCRIPTION
bool	

status

Get the current status of the async operation.

Declaration

```
protected string status
```

Field Value

TYPE	DESCRIPTION
string	

Properties

Current

IEnumerator.Current implementation.

Declaration

```
public object Current { get; }
```

Property Value

TYPE	DESCRIPTION
object	

IsDone

Returns true if the async operation has finished or false if it is still running.

Declaration

```
public bool IsDone { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

IsSuccessful

Returns true if the async operation completed successfully or false if an error occurred.

Declaration

```
public bool IsSuccessful { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Progress

Get the current progress of the async operation. This is a normalized value between 0-1.

Declaration

```
public float Progress { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
float	

ProgressPercentage

Get the current progress percentage of the async operation.

Declaration

```
public int ProgressPercentage { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Result

Get the UnityEngine.Object result of the async operation.

Declaration

```
public Object Result { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
Object	

Status

Get the current status of the async operation.

Declaration

```
public string Status { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
string	

Methods

Await(long)

Block the main thread until the async operation has completed. Use with caution. This can cause an infinite loop if the async operation never completes, if the operation is not true async, or if the async operation relies on data from the main thread.

Declaration

```
public void Await(long msTimeout = 10000)
```

Parameters

TYPE	NAME	DESCRIPTION
long	msTimeout	

Exceptions

TYPE	CONDITION
TimeoutException	The await operation took longer that the specified timeout milliseconds, so was aborted to avoid infinite waiting

Complete(bool, Object)

Complete the operation with the specified success status. This will cause IsDone to become true and Progress to become 1.

Declaration

```
public void Complete(bool success, Object result = null)
```

Parameters

TYPE	NAME	DESCRIPTION
bool	success	Was the operation completed successfully
Object	result	An optional result object

Completed(bool, Object)

Create a new instance with the specified success status. This will cause IsDone to become true and Progress to become 1.

Declaration

```
public static DLCAsync Completed(bool success, Object result = null)
```

Parameters

TYPE	NAME	DESCRIPTION
bool	success	Was the operation completed successfully
Object	result	An optional result object

Returns

TYPE	DESCRIPTION
DLCAsync	

Error(string)

Create a new instance with an error status. This will cause IsDone to become true and Progress to become 1.

Declaration

```
public static DLCAsync Error(string error)
```

Parameters

TYPE	NAME	DESCRIPTION
string	error	The error message for the failure

Returns

TYPE	DESCRIPTION
DLCAsync	

Error(string, Object)

Complete the operation with an error status. This will cause IsDone to become true and Progress to become 1.

Declaration

```
public void Error(string status, Object result = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	status	The error message for the failure
Object	result	An optional result object

MoveNext()

IEnumerator.MoveNext() implementation.

Declaration

```
public bool MoveNext()
```

Returns

TYPE	DESCRIPTION
bool	True if the enumerator advanced successfully or false if not

Reset()

IEnumerator.Reset() implementation.

Declaration

```
public void Reset()
```

UpdateProgress(int, int)

Update the load progress for this operation. Calculates the progress as a value from 0-1 based on the input values.

Declaration

```
protected void UpdateProgress(int current, int total)
```

Parameters

TYPE	NAME	DESCRIPTION
int	current	The current number of tasks that have been completed
int	total	The total number of tasks that should be completed

UpdateProgress(float)

Update the load progress for this operation. The specified progress value should be in the range of 0-1, and will be clamped if not.

Declaration

```
protected void UpdateProgress(float progress)
```

Parameters

TYPE	NAME	DESCRIPTION
float	progress	The current progress value between 0-1

UpdateStatus(string)

Update the status message for this operation. Useful to show the current status if a failure occurs.

Declaration

```
protected void UpdateStatus(string status)
```

Parameters

TYPE	NAME	DESCRIPTION
string	status	The status message

UpdateTasks()

Called when the async operation can perform some logic.

Declaration

```
protected virtual void UpdateTasks()
```

Implements

IEnumerator

Class DLCAsync<T>

An awaitable object that is returned by async operations so you can wait for completion in a coroutine as well as access progress and status information. Used to wait for an async operation to be completed with a result object.

Inheritance

object
DLCAsync
DLCAsync<T>
DLCBatchAsync<T>

Implements

IEnumerator

Inherited Members

DLCAsync.isSuccessful
DLCAsync.status
DLCAsync.Status
DLCAsync.Progress
DLCAsync.ProgressPercentage
DLCAsync.IsDone
DLCAsync.IsSuccessful
DLCAsync.Current
DLCAsync.UpdateTasks()
DLCAsync.MoveNext()
DLCAsync.Reset()
DLCAsync.Await(long)
DLCAsync.UpdateStatus(string)
DLCAsync.UpdateProgress(int, int)
DLCAsync.UpdateProgress(float)
DLCAsync.Error(string, Object)
DLCAsync.Complete(bool, Object)
DLCAsync.Completed(bool, Object)

Namespace: DLCToolkit

Assembly: DLCToolkit.dll

Syntax

```
public class DLCAsync<T> : DLCAsync, IEnumerator
```

Type Parameters

NAME	DESCRIPTION
T	The generic result type

Constructors

DLCAsync()

Create a new instance.

Declaration

```
public DLCAsync()
```

Properties

Result

Get the generic result of the async operation.

Declaration

```
public T Result { get; }
```

Property Value

TYPE	DESCRIPTION
T	

Methods

Complete(bool, T)

Complete the operation with the specified success status. This will cause IsDone to become true and Progress to become 1.

Declaration

```
public void Complete(bool success, T result = default)
```

Parameters

TYPE	NAME	DESCRIPTION
bool	success	Was the operation completed successfully
T	result	An optional result object

Completed(bool, T)

Create a new instance with the specified success status. This will cause IsDone to become true and Progress to become 1.

Declaration

```
public static DLCAsync<T> Completed(bool success, T result = default)
```

Parameters

TYPE	NAME	DESCRIPTION
bool	success	Was the operation completed successfully
T	result	An optional result object

Returns

TYPE	DESCRIPTION
DLCAsync<T>	

Error(string)

Create a new instance an error status. This will cause IsDone to become true and Progress to become 1.

Declaration

```
public static DLCAsync<T> Error(string error)
```

Parameters

TYPE	NAME	DESCRIPTION
string	error	The error message for the failure

Returns

TYPE	DESCRIPTION
DLCAsync<T>	

Error(string, T)

Complete the operation with an error status. This will cause IsDone to become true and Progress to become 1.

Declaration

```
public void Error(string status, T result = default)
```

Parameters

TYPE	NAME	DESCRIPTION
string	status	The error message for the failure
T	result	An optional result object

Implements

IEnumerator

Class DLCBatchAsync<T>

A batch async awaitable operation that contains multiple sub async operations. Can represent the progress and status of all operations combined and will wait until all operations have finished. Note that all sub operations will run in parallel in most cases, so a batch operation will only usually take as much time as the longest sub operation in most cases, plus some minor overhead for management.

Inheritance

object
DLCAsync
DLCAsync<T[]>
DLCBatchAsync<T>

Implements

IEnumerator

Inherited Members

DLCAsync<T[]>.Result
DLCAsync<T[]>.Error(string, T[])
DLCAsync<T[]>.Complete(bool, T[])
DLCAsync<T[]>.Completed(bool, T[])
DLCAsync<T[]>.Error(string)
DLCAsync.Result
DLCAsync.Status
DLCAsync.Progress
DLCAsync.ProgressPercentage
DLCAsync.IsDone
DLCAsync.IsSuccessful
DLCAsync.Current
DLCAsync.MoveNext()
DLCAsync.Reset()
DLCAsync.Await(long)
DLCAsync.Error(string, Object)
DLCAsync.Complete(bool, Object)
DLCAsync.Completed(bool, Object)
DLCAsync.Error(string)

Namespace: DLCToolkit

Assembly: DLCToolkit.dll

Syntax

```
public sealed class DLCBatchAsync<T> : DLCAsync<T[]>, IEnumerator
```

Type Parameters

NAME	DESCRIPTION
T	The generic type returned by each sub operation

Properties

CompletedCount

Get the total number of async operations in this batch that have completed with or without error.

Declaration

```
public int CompletedCount { get; }
```

Property Value

TYPE	DESCRIPTION
int	

LoadingCount

Get the total number of async operations in this batch that have not yet completed.

Declaration

```
public int LoadingCount { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Tasks

Get access to each individual async task that is contained in this batch. Useful to access progress and status of a specific operation rather that using the total combined Progress.

Declaration

```
public IReadOnlyList<DLCAsync<T>> Tasks { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<DLCAsync<T>>	

TotalCount

Get the total number of async operations in this batch.

Declaration

```
public int TotalCount { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Methods

UpdateTasks()

Called when this async operation can update.

Declaration

```
protected override void UpdateTasks()
```

Overrides

DLCAsync.UpdateTasks()

Implements

IEnumerator

Class DLCContent

Represents a DLC loaded in memory. Provides access to metadata, icons, plus API's for loading assets and scenes. Will remain in memory until Unload(bool), Dispose() or until the game exits.

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
DLCContent

Implements

IDLCAsyncProvider
IDisposable

Inherited Members

Behaviour.enabled
Behaviour.isActiveAndEnabled
Component.GetComponent(Type)
Component.GetComponent<T>()
Component.TryGetComponent(Type, out Component)
Component.TryGetComponent<T>(out T)
Component.GetComponent(string)
Component.GetComponentInChildren(Type, bool)
Component.GetComponentInChildren(Type)
Component.GetComponentInChildren<T>(bool)
Component.GetComponentInChildren<T>()
Component.GetComponentsInChildren(Type, bool)
Component.GetComponentsInChildren(Type)
Component.GetComponentsInChildren<T>(bool)
Component.GetComponentsInChildren<T>(bool, List<T>)
Component.GetComponentsInChildren<T>()
Component.GetComponentsInChildren<T>(List<T>)
Component.GetComponentInParent(Type, bool)
Component.GetComponentInParent(Type)
Component.GetComponentInParent<T>(bool)
Component.GetComponentInParent<T>()
Component.GetComponentsInParent(Type, bool)
Component.GetComponentsInParent(Type)
Component.GetComponentsInParent<T>(bool)
Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)
Component.SendMessageUpwards(string, SendMessageOptions)

Component.SendMessage(string, object)
Component.SendMessage(string)
Component.SendMessage(string, object, SendMessageOptions)
Component.SendMessage(string, SendMessageOptions)
Component.BroadcastMessage(string, object, SendMessageOptions)
Component.BroadcastMessage(string, object)
Component.BroadcastMessage(string)
Component.BroadcastMessage(string, SendMessageOptions)
Component.transform
Component.gameObject
Component.tag
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.FindObjectsByType(Type, FindObjectsSortMode)
Object.FindObjectsByType(Type, FindObjectsInactive, FindObjectsSortMode)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsByType<T>(FindObjectsSortMode)
Object.FindObjectsOfType<T>(bool)
Object.FindObjectsByType<T>(FindObjectsInactive, FindObjectsSortMode)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindFirstObjectByType<T>()
Object.FindAnyObjectByType<T>()
Object.FindFirstObjectByType<T>(FindObjectsInactive)
Object.FindAnyObjectByType<T>(FindObjectsInactive)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindFirstObjectByType(Type)
Object.FindAnyObjectByType(Type)
Object.FindObjectOfType(Type, bool)

Object.FindFirstObjectByType(Type, FindObjectsInactive)
Object.FindAnyObjectByType(Type, FindObjectsInactive)
Object.ToString()
Object.name
Object.hideFlags

Namespace: DLCToolkit

Assembly: DLCToolkit.dll

Syntax

```
public sealed class DLCContent : MonoBehaviour, IDLCAsyncProvider, IDisposable
```

Fields

OnUnloaded

Called when the DLC content has finished unloading. Usually this means an unload request was made of the content was disposed.

Declaration

```
[HideInInspector]
public UnityEvent OnUnloaded
```

Field Value

TYPE	DESCRIPTION
UnityEvent	

OnWillUnload

Called when the DLC content is just about to be unloaded. Usually this means an unload request was made of the content was disposed.

Declaration

```
[HideInInspector]
public UnityEvent OnWillUnload
```

Field Value

TYPE	DESCRIPTION
UnityEvent	

Properties

HintLoadPath

Get the local path where the DLC content was loaded from.

Declaration

```
public string HintLoadPath { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Exceptions

TYPE	CONDITION
DLCNotLoadedException	The DLC is not currently loaded

IconProvider

Get the IDLCIconProvider for the loaded DLC content. Provides access to all icons associated with the DLC if available.

Declaration

```
public IDLCIconProvider IconProvider { get; }
```

Property Value

TYPE	DESCRIPTION
IDLCIconProvider	

Exceptions

TYPE	CONDITION
DLCNotLoadedException	The DLC is not currently loaded

IsLoaded

Check if the DLC content has been successfully loaded.

Declaration

```
public bool IsLoaded { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

IsLoading

Check if the DLC content is currently loading.

Declaration

```
public bool IsLoading { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Metadata

Get the IDLCMetadata for the loaded DLC content. Provides access to useful metadata such as name, version, author information and more.

Declaration

```
public IDLCMetadata Metadata { get; }
```

Property Value

TYPE	DESCRIPTION
IDLCMetadata	

Exceptions

TYPE	CONDITION
DLCNotLoadedException	The DLC is not currently loaded

NameInfo

Get the IDLCNameInfo for the loaded DLC content. Provides access to useful identifying data such as name, version and unique key.

Declaration

```
public IDLCNameInfo NameInfo { get; }
```

Property Value

TYPE	DESCRIPTION
IDLCNameInfo	

Exceptions

TYPE	CONDITION
DLCNotLoadedException	The DLC is not currently loaded

SceneAssets

Get a collection DLCSceneAsset for the loaded DLC content listing each scene asset that was included. Scene assets are simply Unity scenes. Provides access to useful asset metadata such as name, path, extension, type and more.

Declaration

```
public DLCAssetCollection<DLCSceneAsset> SceneAssets { get; }
```

Property Value

TYPE	DESCRIPTION
DLCAssetCollection<DLCSceneAsset>	

Exceptions

TYPE	CONDITION

TYPE	CONDITION
DLCNotLoadedException	The DLC is not currently loaded or is loaded in metadata only mode

SharedAssets

Get a collection DLCSharedAsset for the loaded DLC content listing each asset that was included. Shared assets are all non-scene assets types such as prefab, texture, material, audio clip, etc. Provides access to useful asset metadata such as name, path, extension, type and more.

Declaration

```
public DLCAssetCollection<DLCSharedAsset> SharedAssets { get; }
```

Property Value

TYPE	DESCRIPTION
DLCAssetCollection<DLCSharedAsset>	

Exceptions

TYPE	CONDITION
DLCNotLoadedException	The DLC is not currently loaded or is loaded in metadata only mode

Methods

Dispose()

Unload this DLC and release all associated resources. Will call Unload(bool) with `true` argument internally to cause the DLC to be unloaded from memory. Note that this will cause the DLCContent container object to be recycled and will become available for other load operations. For that reason references should not be kept after dispose has been called.

Declaration

```
public void Dispose()
```

Unload(bool)

Request that the DLC contents be unloaded from memory. This will case DLC metadata, assets and scenes to be unloaded. Note that the DLCContent container object will not be destroyed and will remain in memory until manually destroyed. Use Dispose() to unload the content and to recycle the DLCContent container object for use in other DLC load operations.

Declaration

```
public void Unload(bool withAssets = true)
```

Parameters

TYPE	NAME	DESCRIPTION
bool	withAssets	Should asset instances also be unloaded

Implements

IDLCAsyncProvider
IDisposable

Enum DLCContentFlags

The content flags that indicate which types of content are included in a given DLC.

Namespace: DLCToolkit

Assembly: DLCToolkit.dll

Syntax

```
[Flags]
public enum DLCContentFlags
```

Fields

NAME	DESCRIPTION
Assets	The DLC contains one or more shared assets such as prefabs, textures, materials, audio clip, etc.
Scenes	The DLC contains one or more scenes.
Scripts	The DLC contains one or more script assemblies.

Class DLCDirectory

A utility class for mounting a specific folder path as a designated DLC install folder. Contains useful helper methods to find and access the DLC contents included in the target folder. Note that only valid DLC files/information will be returned using this API (IsDLCFile(string) = true), even if the folder contains additional non-DLC files. There is also support for detecting when DLC files are added or removed from the target folder.

Inheritance

object

DLCDirectory

Namespace: DLCToolkit

Assembly: DLCToolkit.dll

Syntax

```
public sealed class DLCDirectory
```

Constructors

DLCDirectory(string, SearchOption, string, bool)

Create new instance for the target directory. Note that the specified folder path must already exist or an exception will be thrown.

Declaration

```
public DLCDirectory(string dlcDirectoryPath, SearchOption option = SearchOption.TopDirectoryOnly, string
extension = null, bool raiseFileEvents = false)
```

Parameters

TYPE	NAME	DESCRIPTION
string	dlcDirectoryPath	The directory path where DLC files may be installed
SearchOption	option	The search option to use when scanning for DLC content
string	extension	An optional extension for DLC content to narrow the search
bool	raiseFileEvents	Should the directory trigger events when DLC files are added and removed

Exceptions

TYPE	CONDITION
ArgumentException	The specified path is null or empty
DirectoryNotFoundException	The specified directory does not exist
NotSupportedException	DLC directory is not supported on the current platform

Fields

OnDLCFileAdded

Called when a valid DLC format was installed in this directory. Note that events will only be called when `raiseFileEvents` is enabled in the constructor.

Declaration

```
public UnityEvent<string> OnDLCFileAdded
```

Field Value

TYPE	DESCRIPTION
UnityEvent<string>	

OnDLCFileDeleted

Called when a valid DLC format was uninstalled from this directory. Note that events will only be called when `raiseFileEvents` is enabled in the constructor.

Declaration

```
public UnityEvent<string> OnDLCFileDeleted
```

Field Value

TYPE	DESCRIPTION
UnityEvent<string>	

Properties

DLCCount

Get the total number of valid DLC files installed in this directory.

Declaration

```
public int DLCCount { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Methods

EnumerateAllDLC(string)

Enumerate the metadata for all valid installed DLC content.

Declaration

```
public IEnumerable<IDLCMetadata> EnumerateAllDLC(string searchName = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	searchName	An optional search name to narrow the search

Returns

TYPE	DESCRIPTION
IEnumerable<IDLCMetadata>	An enumerable for all installed DLC metadata

EnumerateDLCFiles(string)

Enumerate file paths for all valid installed DLC content.

Declaration

```
public IEnumerable<string> EnumerateDLCFiles(string searchName = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	searchName	An optional search name to narrow the search

Returns

TYPE	DESCRIPTION
IEnumerable<string>	An enumerable for all installed DLC files

EnumerateDLCUniqueKeys(string)

Enumerate the unique keys for all valid installed DLC content.

Declaration

```
public IEnumerable<string> EnumerateDLCUniqueKeys(string searchName = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	searchName	An optional search name to narrow the search

Returns

TYPE	DESCRIPTION
IEnumerable<string>	An enumerable for all installed DLC unique keys

GetAllDLC(string)

Get the metadata for all valid installed DLC content.

Declaration

```
public IDLCMetadata[] GetAllDLC(string searchName = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	searchName	An optional search name to narrow the search

Returns

TYPE	DESCRIPTION
IDLCMetadata[]	An array of metadata for all installed DLC content

GetDLC(string)

Try to get the metadata for the DLC with the specified unique key.

Declaration

```
public IDLCMetadata GetDLC(string uniqueKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key for the DLC

Returns

TYPE	DESCRIPTION
IDLCMetadata	The metadata for the DLC if found or null if the DLC is not installed

GetDLC(string, Version)

Try to get the metadata for the DLC with the specified name and optional version.

Declaration

```
public IDLCMetadata GetDLC(string name, Version version = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	The name of the DLC

TYPE	NAME	DESCRIPTION
Version	version	The optional version of the DLC if an explicit match is required

Returns

TYPE	DESCRIPTION
IDLCMetadata	The metadata for the DLC if found or null if the DLC is not installed

GetDLCFile(string)

Try to get the file path for the DLC with the specified unique key.

Declaration

```
public string GetDLCFile(string uniqueKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key for the DLC

Returns

TYPE	DESCRIPTION
string	The file path for the DLC if found or null if the DLC is not installed

GetDLCFile(string, Version)

Try to get the file path for the DLC with the specified name and optional version.

Declaration

```
public string GetDLCFile(string name, Version version = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	The name of the DLC
Version	version	The optional version of the DLC if an explicit match is required

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
string	The file path for the DLC if found or null if the DLC is not installed

GetDLCFiles(string)

Get file paths for all valid installed DLC content.

Declaration

```
public string[] GetDLCFiles(string searchName = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	searchName	An optional search name to narrow the search

Returns

TYPE	DESCRIPTION
string[]	An array of file paths for all installed DLC content

GetDLCUniqueKeys(string)

Get the unique keys for all valid installed DLC content.

Declaration

```
public string[] GetDLCUniqueKeys(string searchName = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	searchName	An optional search name to narrow the search

Returns

TYPE	DESCRIPTION
string[]	An array of unique keys for all installed DLC content

HasDLC(string)

Check if a valid DLC with the specified unique ey is installed in this directory.

Declaration

```
public bool HasDLC(string uniqueKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key for the DLC

Returns

TYPE	DESCRIPTION
bool	True if the DLC is installed or false if not

HasDLC(string, Version)

Check if a valid DLC with the specified name and optional version is installed in this directory.

Declaration

```
public bool HasDLC(string name, Version version = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	The name of the DLC
Version	version	The optional version of the DLC if an explicit match is required

Returns

TYPE	DESCRIPTION
bool	True if the DLC is installed or false if not

Enum DLCIconType

The built in icon size that we need to access.

Namespace: DLCToolkit

Assembly: DLCToolkit.dll

Syntax

```
public enum DLCIconType
```

Fields

NAME	DESCRIPTION
ExtraLarge	Access the extra large icon for the DLC. Usually a high res icon atleast 512px square or larger.
Large	Access the large icon for the DLC. Usually between 64 - 256px square in size.
Medium	Access the medium icon for the DLC. Usually between 32 - 64px square in size.
Small	Access the small icon for the DLC. Usually between 16 - 32px square in size.

Class DLCNotAvailableException

An exception thrown when DLC is not available upon request.

Inheritance

object
Exception
DLCNotAvailableException

Implements

ISerializable

Inherited Members

Exception.GetBaseException()
Exception.GetObjectData(SerializationInfo, StreamingContext)
Exception.GetType()
Exception.ToString()
Exception.Data
Exception.HelpLink
Exception.HResult
Exception.InnerException
Exception.Message
Exception.Source
Exception.StackTrace
Exception.TargetSite

Namespace: DLCToolkit

Assembly: DLCToolkit.dll

Syntax

```
public sealed class DLCNotAvailableException : Exception, ISerializable
```

Constructors

DLCNotAvailableException()

Create a new instance.

Declaration

```
public DLCNotAvailableException()
```

DLCNotAvailableException(string)

Create a new instance.

Declaration

```
public DLCNotAvailableException(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	The exception message

Implements

ISerializable

Class DLCNotLoadedException

An exception thrown when DLC content is trying to be accessed but the DLC is not loaded.

Inheritance

object
Exception
DLCNotLoadedException

Implements

ISerializable

Inherited Members

Exception.GetBaseException()
Exception.GetObjectData(SerializationInfo, StreamingContext)
Exception.GetType()
Exception.ToString()
Exception.Data
Exception.HelpLink
Exception.HResult
Exception.InnerException
Exception.Message
Exception.Source
Exception.StackTrace
Exception.TargetSite

Namespace: DLCToolkit

Assembly: DLCToolkit.dll

Syntax

```
public sealed class DLCNotLoadedException : Exception, ISerializable
```

Constructors

DLCNotLoadedException()

Create new instance.

Declaration

```
public DLCNotLoadedException()
```

DLCNotLoadedException(string)

Create new instance.

Declaration

```
public DLCNotLoadedException(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
string	message	The exception message

Implements

ISerializable

Class DLCStreamProvider

Represents a data stream source containing DLC content.

Inheritance

object

DLCStreamProvider

Implements

IDisposable

Namespace: DLCToolkit

Assembly: DLCToolkit.dll

Syntax

```
public abstract class DLCStreamProvider : IDisposable
```

Properties

HintPath

Get the path where the DLC content was sourced from.

Declaration

```
public abstract string HintPath { get; }
```

Property Value

TYPE	DESCRIPTION
string	

SupportsMultipleStreams

Does the stream provider support multiple simultaneous open read streams. Can improve performance if enabled but each opened stream must have a unique stream/file handle to avoid seek read issues.

Declaration

```
public abstract bool SupportsMultipleStreams { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Methods

Dispose()

Dispose of any open streams used by this stream provider.

Declaration

```
public abstract void Dispose()
```

FromData(byte[], string)

Create a stream provider from the specified dlc file data. This is recommended over FromStream(Stream, string) as it can support

multiple simultaneous reads for quicker loading.

Declaration

```
public static DLCStreamProvider FromData(byte[] data, string hintPath = null)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	data	The data where the dlc content is stored
string	hintPath	An optional hint path describing where the stream content originated from

Returns

TYPE	DESCRIPTION
DLCStreamProvider	A stream provider that allows access to the DLC content via a standard API

FromFile(string)

Create a stream provider from the specified file path. It is highly recommended to use this option where possible, as reading from file path can support multiple simultaneous read operations to offset quicker load times.

Declaration

```
public static DLCStreamProvider FromFile(string localDLCPath)
```

Parameters

TYPE	NAME	DESCRIPTION
string	localDLCPath	The file path for the DLC content

Returns

TYPE	DESCRIPTION
DLCStreamProvider	A stream provider that allows access to the DLC content via a standard API

FromStream(Stream, string)

Create a stream provider from the specified stream. Use FromData(byte[], string) or FromFile(string) where possible as both options support multiple simultaneous read calls for quicker loading.

Declaration

```
public static DLCStreamProvider FromStream(Stream stream, string hintPath = null)
```

Parameters

TYPE	NAME	DESCRIPTION
Stream	stream	The stream to load the content from
string	hintPath	An optional hint path describing where the stream content originated from

Returns

TYPE	DESCRIPTION
DLCStreamProvider	A stream provider that allows access to the DLC content via a standard API

OpenReadStream()

Try to open the DLC content stream for reading.

Declaration

```
public abstract Stream OpenReadStream()
```

Returns

TYPE	DESCRIPTION
Stream	An open readable stream that supports seeking

OpenReadStream(long, long)

Try to open the DLC content stream for reading with the specified stream offset and size bounds. The returned streams position '0' must represent the given offset for example.

Declaration

```
public abstract Stream OpenReadStream(long offset, long size)
```

Parameters

TYPE	NAME	DESCRIPTION
long	offset	The offset into the base stream
long	size	The size of the data stream

Returns

TYPE	DESCRIPTION
Stream	

Implements

IDisposable

Interface IDLCAsyncProvider

A host that is able to manage the invocation of an async operation.

Namespace: DLCToolkit

Assembly: DLCToolkit.dll

Syntax

```
public interface IDLCAsyncProvider
```

Methods

RunAsync(IEnumerator)

Start a new async operation.

Declaration

```
Coroutine RunAsync(IEnumerator routine)
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerator	routine	The async method to invoke

Returns

TYPE	DESCRIPTION
Coroutine	The coroutine for the async method

Interface IDLCIconProvider

Provides access to various icon content that has been registered with the DLC content. Useful for displaying in UI's or splash screens to indicate that a certain DLC has been detected and activated.

Namespace: DLCToolkit

Assembly: DLCToolkit.dll

Syntax

```
public interface IDLCIconProvider
```

Methods

HasCustomIcon(string)

Check if the custom icon with the specified key is available for this DLC.

Declaration

```
bool HasCustomIcon(string iconKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	iconKey	The string key for the custom icon

Returns

TYPE	DESCRIPTION
bool	True if the icon is present or false if not

HasIcon(DLCIconType)

Check if the specified icon is available for this DLC.

Declaration

```
bool HasIcon(DLCIconType iconType)
```

Parameters

TYPE	NAME	DESCRIPTION
DLCIconType	iconType	The type of icon

Returns

TYPE	DESCRIPTION
bool	True if the icon is present or false if not

LoadCustomIcon(string)

Load the custom icon specified by the unique icon key string that is assigned when setting up your DLC content. Use this method when all the DLCIconType options have been used up or do not properly describe the icon content.

Declaration

```
Texture2D LoadCustomIcon(string iconKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	iconKey	The unique string key of the icon to fetch

Returns

TYPE	DESCRIPTION
Texture2D	The texture for the requested icon

LoadCustomIconAsync(string)

Load the custom icon specified by the unique icon key string asynchronously that is assigned when setting up your DLC content. Use this method when all the DLCIconType options have been used up or do not properly describe the icon content.

Declaration

```
DLCAsync<Texture2D> LoadCustomIconAsync(string iconKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	iconKey	The unique string key of the icon to fetch

Returns

TYPE	DESCRIPTION
DLCAsync<Texture2D>	The texture for the requested icon

LoadIcon(DLCIconType)

Load the icon of the specified type or null if no icon was assigned.

Declaration

```
Texture2D LoadIcon(DLCIconType iconType)
```

Parameters

TYPE	NAME	DESCRIPTION
DLCIconType	iconType	The icon type to fetch

Returns

TYPE	DESCRIPTION
Texture2D	The texture for the requested icon

LoadIconAsync(DLCIconType)

Load the icon of the specified type asynchronously or null if no icon was assigned.

Declaration

```
DLCAsync<Texture2D> LoadIconAsync(DLCIconType iconType)
```

Parameters

TYPE	NAME	DESCRIPTION
DLCIconType	iconType	The icon type to fetch

Returns

TYPE	DESCRIPTION
DLCAsync<Texture2D>	The texture for the requested icon

Interface IDLCMetadata

Access all metadata for a given DLC.

Namespace: DLCToolkit

Assembly: DLCToolkit.dll

Syntax

```
public interface IDLCMetadata
```

Properties

BuildTime

Get the time stamp for when the DLC was built. Can be used as a unique DLC identifier to use in multiplayer scenarios or similar where you need to ensure that the same DLC is used between clients and the name/version combo may not give enough distinction.

Declaration

```
DateTime BuildTime { get; }
```

Property Value

TYPE	DESCRIPTION
DateTime	

ContentFlags

Access the content flags for this DLC content to determine which types of assets are included.

Declaration

```
DLCContentFlags ContentFlags { get; }
```

Property Value

TYPE	DESCRIPTION
DLCContentFlags	

Description

The description for this DLC content.

Declaration

```
string Description { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Developer

The developer name who created this DLC content.

Declaration

```
string Developer { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Guid

The unique Guid for the DLC content assigned at creation.

Declaration

```
string Guid { get; }
```

Property Value

TYPE	DESCRIPTION
string	

NameInfo

Access the name info for the DLC content.

Declaration

```
IDLCNameInfo NameInfo { get; }
```

Property Value

TYPE	DESCRIPTION
IDLCNameInfo	

Publisher

The publisher name who published this DLC content.

Declaration

```
string Publisher { get; }
```

Property Value

TYPE	DESCRIPTION
string	

ToolkitVersion

The DLC toolkit version required to load this DLC.

Declaration

```
Version ToolkitVersion { get; }
```

Property Value

TYPE	DESCRIPTION
Version	

UnityVersion

The Unity version string required to load this DLC.

Declaration

```
string UnityVersion { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Methods

GetNetworkUniqueIdentifier(bool)

Returns a unique string identifier for this DLC including name and version information which may be used between networking clients to compare DLC content in use. Can be used to determine if two network clients are using the same DLC (Or atleast have the same unique key and version) which will usually be a good enough indicator. Be sure to enabled `includeBuildStamp` if you want an explicit version match of DLC's (Exact same build) as it will serialize the build time stamp for when the DLC file was created and will almost certainly identify that the DLC is the exact same version and build. Even if the unique key and version information match, there could be potential for the DLC's to be matched if the version was not updated correctly for example.

Declaration

```
string GetNetworkUniqueIdentifier(bool includeBuildStamp)
```

Parameters

TYPE	NAME	DESCRIPTION
bool	includeBuildStamp	Should the build time stamp be included in the unique string which will guarantee an explicit DLC match

Returns

TYPE	DESCRIPTION
string	A short string identifier to uniquely identify this DLC content and can be shared between network clients to compare DLC's in use

Interface IDLCNameInfo

Access name information for a given DLC.

Namespace: DLCToolkit

Assembly: DLCToolkit.dll

Syntax

```
public interface IDLCNameInfo
```

Properties

Name

The friendly name of the DLC content.

Declaration

```
string Name { get; }
```

Property Value

TYPE	DESCRIPTION
string	

UniqueKey

The unique named key for the DLC content.

Declaration

```
string UniqueKey { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Version

The version information for the DLC content.

Declaration

```
Version Version { get; }
```

Property Value

TYPE	DESCRIPTION
Version	

Namespace DLCToolkit.Assets

Classes

DLCAsset

Represents a specific asset included in the DLC. Provides access to useful metadata such as name, path, extension, type and more which can be quickly accessed without forcing the asset to be loaded into memory. Note that this can represent shared assets and scene assets in the same form.

DLCAssetCollection<T>

Represents a collection of DLCAsset which are stored in a given DLC. Note that this collection can contain either shared assets or scene assets, but they will not be stored together. SharedAssets provides access to all shared assets for the DLC, and SceneAssets provides access to all scene assets.

DLCSceneAsset

Represents a scene asset included in the DLC. Scene assets are simply Unity scenes.

DLCSharedAsset

Represents a shared asset included in the DLC. Shared assets are content assets such as prefabs, textures, materials, audio clips, scriptable objects and more.

Class DLCAsset

Represents a specific asset included in the DLC. Provides access to useful metadata such as name, path, extension, type and more which can be quickly accessed without forcing the asset to be loaded into memory. Note that this can represent shared assets and scene assets in the same form.

Inheritance

object
DLCAsset
DLCSceneAsset
DLCSharedAsset

Namespace: DLCToolkit.Assets

Assembly: DLCToolkit.dll

Syntax

```
public abstract class DLCAsset
```

Fields

assetID

The unique ID for this asset.

Declaration

```
protected int assetID
```

Field Value

TYPE	DESCRIPTION
int	

assetType

The type of this asset.

Declaration

```
protected Type assetType
```

Field Value

TYPE	DESCRIPTION
Type	

extension

The file extension of the asset.

Declaration

```
protected string extension
```

Field Value

TYPE	DESCRIPTION
string	

fullName

The full name relative to the original Unity project assets folder of the asset.

Declaration

```
protected string fullName
```

Field Value

TYPE	DESCRIPTION
string	

name

The name of the asset.

Declaration

```
protected string name
```

Field Value

TYPE	DESCRIPTION
string	

relativeName

The relative name relative to the original DLC content folder of the asset.

Declaration

```
protected string relativeName
```

Field Value

TYPE	DESCRIPTION
string	

Properties

AssetID

The unique id for this asset. This is guaranteed to be unique during the current session. Id's are not persistent.

Declaration

```
public int AssetID { get; }
```

Property Value

TYPE	DESCRIPTION
int	

AssetMainType

Get the type of the main asset. For example this value will be `GameObject` when dealing with a prefab asset, or `Texture2D` when dealing with a texture.

Declaration

```
public Type AssetMainType { get; }
```

Property Value

TYPE	DESCRIPTION
Type	

Extension

Get the extension of the asset, For example: '.prefab'. The extension will always begin with a '.'.

Declaration

```
public string Extension { get; }
```

Property Value

TYPE	DESCRIPTION
string	

FullName

Get the full name of the asset. The full name is defined as the asset path relative the the export project folder including the asset extension, For example: 'assets/mydlc/subfolder/cube.prefab'.

Declaration

```
public string FullName { get; }
```

Property Value

TYPE	DESCRIPTION
string	

IsBundleLoaded

Check if the containing bundle is currently loaded in memory. If false, any load requests issued may take longer than expected as the containing bundle will first need to be loaded.

Declaration

```
public bool IsBundleLoaded { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

IsLoadable

Check if the asset can be loaded into memory. Only possible when the DLC content has been fully loaded as opposed to partially loaded using a method such as LoadDLCMetadataWithAssets(string).

Declaration

```
public bool IsLoadable { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

IsLoaded

Check if the asset is currently loaded in memory.

Declaration

```
public bool IsLoaded { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Name

Get the name of the asset.

Declaration

```
public string Name { get; }
```

Property Value

TYPE	DESCRIPTION
string	

RelativeName

Get the relative name of the asset. The relative name is defined as the asset path relative to the mod folder without the asset extension, For example: 'subfolder/cube'. If the asset is not in a sub folder then the value wil be equal to Name.

Declaration

```
public string RelativeName { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Methods

IsAssetSubType(Type)

Check if this asset is of the specified type or is derived from the specified type. For example: A `Texture2D` asset will return true if `UnityEngine.Object` is passed as the parameter type, since textures derive from the base object type.

Declaration

```
public bool IsAssetSubType(Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	type	The type to check for equality or derivation

Returns

TYPE	DESCRIPTION
bool	True if the asset is of the specified type, or is derived from the specified type otherwise false

IsAssetSubType<T>()

Check if this asset is of the specified generic type or is derived from the specified type. For example: A `Texture2D` asset will return true if `UnityEngine.Object` is passed as the parameter type, since textures derive from the base object type.

Declaration

```
public bool IsAssetSubType<T>()
```

Returns

TYPE	DESCRIPTION
bool	True if the asset is of the specified generic type, or is derived from the specified type otherwise false

Type Parameters

NAME	DESCRIPTION
T	The generic type to check for equality or derivation

IsAssetType(Type)

Check if this asset is of the specified type. Note that this will not check for derived types and you can use IsAssetSubType(Type) instead.

Declaration

```
public bool IsAssetType(Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	type	The type to check for equality

Returns

TYPE	DESCRIPTION
bool	True if the asset is of specified type of false if not

IsAssetType<T>()

Check if this asset is of the specified generic type. Note that this will not check for derived types and you can use IsAssetSubType<T>() instead.

Declaration

```
public bool IsAssetType<T>()
```

Returns

TYPE	DESCRIPTION
bool	True if the asset is of specified generic type of false if not

Type Parameters

NAME	DESCRIPTION
T	The generic type to check for equality

ToString()

Convert to string representation.

Declaration

```
public override string ToString()
```

Returns

TYPE	DESCRIPTION
string	This asset as a string

Overrides

object.ToString()

Class DLCAssetCollection<T>

Represents a collection of DLCAsset which are stored in a given DLC. Note that this collection can contain either shared assets or scene assets, but they will not be stored together. SharedAssets provides access to all shared assets for the DLC, and SceneAssets provides access to all scene assets.

Inheritance

object

DLCAssetCollection<T>

Implements

IEnumerable<T>

IEnumerable

Namespace: DLCToolkit.Assets

Assembly: DLCToolkit.dll

Syntax

```
public sealed class DLCAssetCollection<T> : IEnumerable<T>, IEnumerable where T : DLCAsset
```

Type Parameters

NAME	DESCRIPTION
T	The generic type of DLCAsset to store in this collection

Properties

AssetCount

Get the total number of shared assets that are available in this DLC.

Declaration

```
public int AssetCount { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Methods

EnumerateAll()

Enumerate all assets that are available in this DLC.

Declaration

```
public IEnumerable<T> EnumerateAll()
```

Returns

TYPE	DESCRIPTION
IEnumerable<T>	An enumerable of assets

EnumerateAllInFolder(string)

Enumerate all assets that are available in this DLC and are located in the specified folder. The folder path should either be relative to the DLC content folder, or relative to the Unity assets folder.

Declaration

```
public IEnumerable<T> EnumerateAllInFolder(string path)
```

Parameters

TYPE	NAME	DESCRIPTION
string	path	The folder path to search in

Returns

TYPE	DESCRIPTION
IEnumerable<T>	An enumerable of assets that are in the specified folder

EnumerateAllInFolderWithExtension(string, string)

Enumerate all assets that are located in ths specified folder and has the specified file extension. The folder path should either be relative to the DLC content folder, or relative to the Unity assets folder.

Declaration

```
public IEnumerable<T> EnumerateAllInFolderWithExtension(string path, string extension)
```

Parameters

TYPE	NAME	DESCRIPTION
string	path	The folder path to search in
string	extension	The file extension to find

Returns

TYPE	DESCRIPTION
IEnumerable<T>	An enumerable of assets in the specified folder with the specified extension

Exceptions

TYPE	CONDITION
ArgumentException	The Specified extension was null or empty

TYPE	CONDITION
ArgumentException	The specified extension was incorrectly formatted and did not start with a leading '.' character

EnumerateAllNames()

Enumerate the names of all assets that are available in this DLC.

Declaration

```
public IEnumerable<string> EnumerateAllNames()
```

Returns

TYPE	DESCRIPTION
IEnumerable<string>	An enumerable of asset names

EnumerateAllOfType(Type)

Enumerate all assets that are available in ths DLC of the specified type. Note that this will check for type equality and not sub types. Use EnumerateAllSubTypesOf(Type) if you want to discover derived types too.

Declaration

```
public IEnumerable<T> EnumerateAllOfType(Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	type	The type of asset to find, for example: UnityEngine.Texture2D

Returns

TYPE	DESCRIPTION
IEnumerable<T>	An enumerable of assets of the specified type

EnumerateAllOfTypeInFolder(Type, string)

Enumerate all assets that are available in this DLC and are located in the specified folder and are of the specified type. The folder path should either be relative to the DLC content folder, or relative to the Unity assets folder. Note that this will check for type equality and will not discover derived types. If you want to find derived types in addition, you should use EnumerateAllSubTypesOfInFolder(Type, string).

Declaration

```
public IEnumerable<T> EnumerateAllOfTypeInFolder(Type type, string path)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	type	The type of asset to find
string	path	The folder path to search in

Returns

TYPE	DESCRIPTION
IEnumerable<T>	An enumerable of assets of the specified type that are located in the specified folder

EnumerateAllOfTypeInFolder<TType>(string)

Enumerate all assets that are available in this DLC and are located in the specified folder and are of the specified generic type. The folder path should either be relative to the DLC content folder, or relative to the Unity assets folder. Note that this will check for type equality and will not discover derived types. If you want to find derived types in addition, you should use EnumerateAllOfTypeInFolder<TType>(string).

Declaration

```
public IEnumerable<T> EnumerateAllOfTypeInFolder<TType>(string path)
```

Parameters

TYPE	NAME	DESCRIPTION
string	path	The folder path to search in

Returns

TYPE	DESCRIPTION
IEnumerable<T>	An enumerable of assets of the specified generic type that are located in the specified folder

Type Parameters

NAME	DESCRIPTION
TType	The generic type of asset to find

EnumerateAllOfType<TType>()

Enumerate all assets that are available in ths DLC of the specified generic type. Note that this will check for type equality and not sub types. Use EnumerateAllSubTypesOf<TType>() if you want to discover derived types too.

Declaration

```
public IEnumerable<T> EnumerateAllOfType<TType>()
```

Returns

TYPE	DESCRIPTION
IEnumerable<T>	An enumerable of assets of the specified generic type

Type Parameters

NAME	DESCRIPTION
TType	The generic type of asset to find, for example: UnityEngine.Texture2D

EnumerateAllRelativeNames()

Enumerate the relative names of all assets that are available in this DLC. The relative name is the asset path relative to the DLC content folder.

Declaration

```
public IEnumerable<string> EnumerateAllRelativeNames()
```

Returns

TYPE	DESCRIPTION
IEnumerable<string>	An enumerable of asset relative names

EnumerateAllSubTypesOf(Type)

Enumerate all assets that are available in this DLC which are of the specified type, or derived from the specified type. Supports discovering derived types, for example: Searching for UnityEngine.Texture type will also return derived UnityEngine.Texture2D types.

Declaration

```
public IEnumerable<T> EnumerateAllSubTypesOf(Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	type	The type of asset or derived asset to find

Returns

TYPE	DESCRIPTION
IEnumerable<T>	An enumerable of assets matching or derived from the specified type

EnumerateAllSubTypesOfInFolder(Type, string)

Enumerate all assets that are available in this DLC and are located in the specified folder and are of the specified type or derived from the specified type. The folder path should either be relative to the DLC content folder, or relative to the Unity assets folder.

Declaration

```
public IEnumerable<T> EnumerateAllSubTypesOfInFolder(Type type, string path)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	type	The type of asset to find
string	path	The folder path to search in

Returns

TYPE	DESCRIPTION
IEnumerable<T>	An enumerable of assets of the specified type or derived from the specified type that are located in the specified folder

EnumerateAllSubTypesOfInFolder<TType>(string)

Enumerate all assets that are available in this DLC and are located in the specified folder and are of the specified type or derived from the specified generic type. The folder path should either be relative to the DLC content folder, or relative to the Unity assets folder.

Declaration

```
public IEnumerable<T> EnumerateAllSubTypesOfInFolder<TType>(string path)
```

Parameters

TYPE	NAME	DESCRIPTION
string	path	The folder path to search in

Returns

TYPE	DESCRIPTION
IEnumerable<T>	An enumerable of assets of the specified generic type or derived from the specified type that are located in the specified folder

Type Parameters

NAME	DESCRIPTION
TType	The generic type of asset to find

EnumerateAllSubTypesOf<TType>()

Enumerate all assets that are available in this DLC which are of the specified type, or derived from the specified generic type. Supports discovering derived types, for example: Searching for UnityEngine.Texture type will also return derived UnityEngine.Texture2D types.

Declaration

```
public IEnumerable<T> EnumerateAllSubTypesOf<TType>()
```

Returns

TYPE	DESCRIPTION
IEnumerable<T>	An enumerable of assets matching or derived from the specified generic type

Type Parameters

NAME	DESCRIPTION
TType	The generic type of asset or derived asset to find

EnumerateAllWithExtension(string)

Enumerate all assets that are available in this DLC with the specified file extension.

Declaration

```
public IEnumerable<T> EnumerateAllWithExtension(string extension)
```

Parameters

TYPE	NAME	DESCRIPTION
string	extension	The file extension to find

Returns

TYPE	DESCRIPTION
IEnumerable<T>	An enumerable of assets with the specified extension

Exceptions

TYPE	CONDITION
ArgumentException	The Specified extension was null or empty
ArgumentException	The specified extension was incorrectly formatted and did not start with a leading '.' character

EnumerateAllWithName(string)

Enumerate all assets that are available in this DLC with the specified name. Note that this method may return multiple results with the same name if they are located in different folders.

Declaration

```
public IEnumerable<T> EnumerateAllWithName(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	The name of the asset to find

Returns

TYPE	DESCRIPTION
IEnumerable<T>	An enumerable of assets with the specified name

Exceptions

TYPE	CONDITION
ArgumentException	The specified name was null or empty

Exists(int)

Check if an asset with the specified id exists in the DLC.

Declaration

```
public bool Exists(int assetID)
```

Parameters

TYPE	NAME	DESCRIPTION
int	assetID	The asset ID to check

Returns

TYPE	DESCRIPTION
bool	True if the asset exists or false if not

Exists(string)

Check if an asset with the specified name or path exists in the DLC.

Declaration

```
public bool Exists(string nameOfPath)
```

Parameters

TYPE	NAME	DESCRIPTION
string	nameOfPath	The asset name or path

Returns

TYPE	DESCRIPTION
bool	True if the asset exists or false if not

Find(int)

Try to find the asset with the specified asset ID.

Declaration

```
public T Find(int assetID)
```

Parameters

TYPE	NAME	DESCRIPTION
int	assetID	The id of the asset

Returns

TYPE	DESCRIPTION
T	An asset if found or null if not

Find(string)

Try to find the asset with the specified name or path. Note that if searching by name and multiple assets with the same name exist in the DLC, this method will return the first asset found during the search.

Declaration

```
public T Find(string nameOrPath)
```

Parameters

TYPE	NAME	DESCRIPTION
string	nameOrPath	The name or path of the asset

Returns

TYPE	DESCRIPTION
T	An asset if found or null

Exceptions

TYPE	CONDITION
ArgumentException	The nameOrPath is null or empty

FindAll()

Find all assets that are available in this DLC.

Declaration

```
public T[] FindAll()
```

Returns

TYPE	DESCRIPTION
T[]	An array of assets or an empty array if no assets are available

FindAllInFolder(string)

Find all assets that are available in this DLC and are located in the specified folder. The folder path should either be relative to the DLC content folder, or relative to the Unity assets folder.

Declaration

```
public T[] FindAllInFolder(string path)
```

Parameters

TYPE	NAME	DESCRIPTION
string	path	The folder path to search in

Returns

TYPE	DESCRIPTION
T[]	An array of assets that are in the specified folder or an empty array if no matches were found

FindAllInFolderWithExtension(string, string)

Find all assets that are located in the specified folder and has the specified file extension. The folder path should either be relative to the DLC content folder, or relative to the Unity assets folder.

Declaration

```
public T[] FindAllInFolderWithExtension(string path, string extension)
```

Parameters

TYPE	NAME	DESCRIPTION
string	path	The folder path to search in
string	extension	The file extension to find

Returns

TYPE	DESCRIPTION
T[]	An array of assets in the specified folder with the specified extension or an empty array if no matches were found

Exceptions

TYPE	CONDITION
ArgumentException	The Specified extension was null or empty
ArgumentException	The specified extension was incorrectly formatted and did not start with a leading '.' character

FindAllNames()

Find the names of all assets that are available in this DLC.

Declaration

```
public string[] FindAllNames()
```

Returns

TYPE	DESCRIPTION
string[]	An array of asset names or an empty array if no assets are available

FindAllOfType(Type)

Find all assets that are available in this DLC of the specified type. Note that this will check for type equality and not sub types. Use FindAllSubTypesOf(Type) if you want to discover derived types too.

Declaration

```
public T[] FindAllOfType(Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	type	The type of asset to find, for example: UnityEngine.Texture2D

Returns

TYPE	DESCRIPTION
T[]	An array of assets matching the specified type or an empty array if no matching assets are available

FindAllOfTypeInFolder(Type, string)

Find all assets that are available in this DLC and are located in the specified folder and are of the specified type. The folder path should either be relative to the DLC content folder, or relative to the Unity assets folder. Note that this will check for type equality

and will not discover derived types. If you want to find derived types in addition, you should use FindAllSubTypesOfInFolder(Type, string).

Declaration

```
public T[] FindAllOfTypeInFolder(Type type, string path)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	type	The type of asset to find
string	path	The folder path to search in

Returns

TYPE	DESCRIPTION
T[]	An array of assets of the specified type that are located in the specified folder or an empty array if no matches were found

FindAllOfTypeInFolder<TType>(string)

Find all assets that are available in this DLC and are located in the specified folder and are of the specified generic type. The folder path should either be relative to the DLC content folder, or relative to the Unity assets folder. Note that this will check for type equality and will not discover derived types. If you want to find derived types in addition, you should use FindAllSubTypesOfInFolder<TType>(string).

Declaration

```
public T[] FindAllOfTypeInFolder<TType>(string path)
```

Parameters

TYPE	NAME	DESCRIPTION
string	path	The folder path to search in

Returns

TYPE	DESCRIPTION
T[]	An array of assets of the specified generic type that are located in the specified folder or an empty array if no matches were found

Type Parameters

NAME	DESCRIPTION
TType	The generic type of asset to find

FindAllOfType<TType>()

Find all assets that are available in this DLC of the specified generic type. Note that this will check for type equality and not sub types. Use FindAllSubTypesOf<TType>() if you want to discover derived types too.

Declaration

```
public T[] FindAllOfType<TType>()
```

Returns

TYPE	DESCRIPTION
T[]	An array of assets matching the specified generic type or an empty array if no matching assets are available

Type Parameters

NAME	DESCRIPTION
TType	The generic type of asset to find, for example: UnityEngine.Texture2D

FindAllRelativeNames()

Find the relative names of all assets that are available in this DLC. The relative name is the asset path relative to the DLC content folder.

Declaration

```
public string[] FindAllRelativeNames()
```

Returns

TYPE	DESCRIPTION
string[]	An array of asset relative names or an empty array if no assets are available

FindAllSubTypesOf(Type)

Find all assets that are available in this DLC with are of the specified type, or derived from the specified type. Supports discovering derived types, for example: Searching for UnityEngine.Texture type will also return derived UnityEngine.Texture2D types.

Declaration

```
public T[] FindAllSubTypesOf(Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	type	The type of asset or derived asset to find

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
T[]	An array of assets matching or derived from the specified type or an empty array if no matching assets are available

FindAllSubTypesOfInFolder(Type, string)

Find all assets that are available in this DLC and are located in the specified folder and are of the specified type or derived from the specified type. The folder path should either be relative to the DLC content folder, or relative to the Unity assets folder.

Declaration

```
public T[] FindAllSubTypesOfInFolder(Type type, string path)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	type	The type of asset to find
string	path	The folder path to search in

Returns

TYPE	DESCRIPTION
T[]	An array of assets of the specified type or derived from the specified type that are located in the specified folder or an empty array if no matches were found

FindAllSubTypesOfInFolder<TType>(string)

Find all assets that are available in this DLC and are located in the specified folder and are of the specified generic type or derived from the specified generic type. The folder path should either be relative to the DLC content folder, or relative to the Unity assets folder.

Declaration

```
public T[] FindAllSubTypesOfInFolder<TType>(string path)
```

Parameters

TYPE	NAME	DESCRIPTION
string	path	The folder path to search in

Returns

TYPE	DESCRIPTION
T[]	An array of assets of the specified generic type or derived from the specified generic type that are located in the specified folder or an empty array if no matches were found

Type Parameters

NAME	DESCRIPTION
TType	The generic type of asset to find

FindAllSubTypesOf<TType>()

Find all assets that are available in this DLC with are of the specified generic type, or derived from the specified generic type. Supports discovering derived types, for example: Searching for UnityEngine.Texture type will also return derived UnityEngine.Texture2D types.

Declaration

```
public T[] FindAllSubTypesOf<TType>()
```

Returns

TYPE	DESCRIPTION
T[]	An array of assets matching or derived from the specified generic type or an empty array if no matching assets are available

Type Parameters

NAME	DESCRIPTION
TType	The type of asset or derived asset to find

FindAllWithExtension(string)

Find all assets that are available in this DLC with the specified file extension.

Declaration

```
public T[] FindAllWithExtension(string extension)
```

Parameters

TYPE	NAME	DESCRIPTION
string	extension	The file extension to find

Returns

TYPE	DESCRIPTION
T[]	An array of matching assets or an empty array if no matches are available

Exceptions

TYPE	CONDITION

TYPE	CONDITION
ArgumentException	The Specified extension was null or empty
ArgumentException	The specified extension was incorrectly formatted and did not start with a leading '.' character

FindAllWithName(string)

Find all assets that are available in this DLC with the specified name. Note that this method may return multiple results with the same name if they are located in different folders.

Declaration

```
public T[] FindAllWithName(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	The name of asset to find

Returns

TYPE	DESCRIPTION
T[]	An array of matching assets or an empty array if no matches are available

Exceptions

TYPE	CONDITION
ArgumentException	The specified name was null or empty

Implements

IEnumerable<T>
IEnumerable

Class DLCSceneAsset

Represents a scene asset included in the DLC. Scene assets are simply Unity scenes.

Inheritance

object
DLCAsset
DLCSceneAsset

Inherited Members

DLCAsset.IsLoaded
DLCAsset.IsLoadable
DLCAsset.IsBundleLoaded
DLCAsset.AssetID
DLCAsset.AssetMainType
DLCAsset.FullName
DLCAsset.RelativeName
DLCAsset.Name
DLCAsset.Extension
DLCAsset.ToString()
DLCAsset.IsAssetType(Type)
DLCAsset.IsAssetType<T>()
DLCAsset.IsAssetSubType(Type)
DLCAsset.IsAssetSubType<T>()

Namespace: DLCToolkit.Assets

Assembly: DLCToolkit.dll

Syntax

```
public sealed class DLCSceneAsset : DLCAsset
```

Methods

ActivatePendingScene()

Attempt to switch activation to the current streamed scene loaded during the last async load operation. This is the same as setting `allowSceneActivation` to true and can be used to switch from a loading screen at a suitable time. You must use one of the LoadAsync(LoadSceneMode, bool) methods first with `allowSceneActivation` set to `false`.

Declaration

```
public void ActivatePendingScene()
```

Exceptions

TYPE	CONDITION
InvalidOperationException	There is no pending load scene operation

Load(LoadSceneMode)

Load the DLCSceneAsset.

Declaration

```
public void Load(LoadSceneMode loadSceneMode)
```

Parameters

TYPE	NAME	DESCRIPTION
LoadSceneMode	loadSceneMode	The mode to use when loading the scene

Load(LoadSceneParameters)

Load the DLCSceneAsset.

Declaration

```
public void Load(LoadSceneParameters loadSceneParameters)
```

Parameters

TYPE	NAME	DESCRIPTION
LoadSceneParameters	loadSceneParameters	The load scene parameters to use when loading the scene

LoadAsync(LoadSceneMode, bool)

Load the DLCSceneAsset asynchronously.

Declaration

```
public DLCAsync LoadAsync(LoadSceneMode loadSceneMode, bool allowSceneActivation = true)
```

Parameters

TYPE	NAME	DESCRIPTION
LoadSceneMode	loadSceneMode	Should the scene be additively loaded into the current scene
bool	allowSceneActivation	Should the scene be activated as soon as it is loaded

Returns

TYPE	DESCRIPTION
DLCAsync	

LoadAsync(LoadSceneParameters, bool)

Load the DLCSceneAsset asynchronously.

Declaration

```
public DLCAsync LoadAsync(LoadSceneParameters loadSceneParameters, bool allowSceneActivation = true)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
LoadSceneParameters	loadSceneParameters	The parameters to use when loading the scene
bool	allowSceneActivation	Should the scene be activated as soon as it is loaded

Returns

TYPE	DESCRIPTION
DLCAsync	

Class DLCSharedAsset

Represents a shared asset included in the DLC. Shared assets are content assets such as prefabs, textures, materials, audio clips, scriptable objects and more.

Inheritance

object
DLCAsset
DLCSharedAsset

Inherited Members

DLCAsset.IsLoaded
DLCAsset.IsLoadable
DLCAsset.IsBundleLoaded
DLCAsset.AssetID
DLCAsset.AssetMainType
DLCAsset.FullName
DLCAsset.RelativeName
DLCAsset.Name
DLCAsset.Extension
DLCAsset.ToString()
DLCAsset.IsAssetType(Type)
DLCAsset.IsAssetType<T>()
DLCAsset.IsAssetSubType(Type)
DLCAsset.IsAssetSubType<T>()

Namespace: DLCToolkit.Assets

Assembly: DLCToolkit.dll

Syntax

```
public sealed class DLCSharedAsset : DLCAsset
```

Methods

Load()

Attempts to load this asset from the dlc.

Declaration

```
public Object Load()
```

Returns

TYPE	DESCRIPTION
Object	The loaded asset

LoadAsync()

Attempts to load this asset from the dlc asynchronously. This method returns a DLCAsync object which is yieldable and contains information about the loading progress and status.

Declaration

```
public DLCAsync<Object> LoadAsync()
```

Returns

TYPE	DESCRIPTION
DLCAsync<Object>	A yieldable DLCAsync object

LoadAsync<T>()

Attempts to load this asset from the dlc asynchronously. This method returns a DLCAsync object which is yieldable and contains information about the loading progress and status.

Declaration

```
public DLCAsync<T> LoadAsync<T>() where T : Object
```

Returns

TYPE	DESCRIPTION
DLCAsync<T>	A yieldable DLCAsync object

Type Parameters

NAME	DESCRIPTION
T	The generic type to load the asset as

LoadWithSubAssets()

Attempts to load this asset with all associated sub assets.

Declaration

```
public Object[] LoadWithSubAssets()
```

Returns

TYPE	DESCRIPTION
Object[]	An array of sub assets for this asset

LoadWithSubAssetsAsync()

Attempts to load this asset with all associated sub assets from the dlc asynchronously. This method returns a DLCAsync object which is yieldable and contains information about the loading progress and status.

Declaration

```
public DLCAsync<Object[]> LoadWithSubAssetsAsync()
```

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
DLCAsync<Object[]>	A yieldable DLCAsync object

LoadWithSubAssetsAsync<T>()

Attempts to load this asset with all associated sub assets from the dlc asynchronously. This method returns a DLCAsync object which is yieldable and contains information about the loading progress and status.

Declaration

```
public DLCAsync<T[]> LoadWithSubAssetsAsync<T>() where T : Object
```

Returns

TYPE	DESCRIPTION
DLCAsync<T[]>	A yieldable DLCAsync object

Type Parameters

NAME	DESCRIPTION
T	The generic asset type used to specify which sub asset types to load

LoadWithSubAssets<T>()

Attempts to load this asset with all associated sub assets. This overload will only return assets that are of the specified generic type such as 'Mesh'.

Declaration

```
public T[] LoadWithSubAssets<T>() where T : Object
```

Returns

TYPE	DESCRIPTION
T[]	An array of sub assets for this asset

Type Parameters

NAME	DESCRIPTION
T	The generic asset type to return

Load<T>()

Attempts to load this asset from the dlc as the specified generic type.

Declaration

```
public T Load<T>() where T : Object
```

Returns

TYPE	DESCRIPTION
T	The loaded asset as the generic type

Type Parameters

NAME	DESCRIPTION
T	The generic type to load the asset as

Namespace DLCToolkit.BuildTools

Classes

DLCBuildPipeline

Main API for building DLC content from script.

DLCBuildResult

Contains information about which aspects of the DLC build request were successful and which were not.

DLCBuildTask

Represents an individual build task of a DLC profile for a specific build platform.

DLCPostBuildAttribute

Used to mark a post DLC build event called just after DLC has been built. Must implement DLCBuildEvent base class.

DLCPostBuildPlatformProfileAttribute

Used to mark a post DLC build event called just after a specific DLC profile platform has been built. Must implement DLCBuildPlatformProfileResultEvent base class.

DLCPreBuildAttribute

Used to mark a pre DLC build event called just before DLC will be built. Must implement DLCBuildEvent base class.

DLCPreBuildPlatformProfileAttribute

Used to mark a pre DLC build event called just before a specific DLC profile platform will be built. Must implement DLCBuildPlatformProfileEvent base class.

DLCPreBuildProfileAttribute

Used to mark a pre DLC build event called just before a specific DLC profile will be built. Must implement DLCBuildProfileEvent base class.

Enums

DLCBuildOptions

Custom build options for building DLC content. Multiple options can be combined.

Enum DLCBuildOptions

Custom build options for building DLC content. Multiple options can be combined.

Namespace: DLCToolkit.BuildTools

Assembly: DLCToolkit.BuildTools.dll

Syntax

```
[Flags]
public enum DLCBuildOptions
```

Fields

NAME	DESCRIPTION
DebugScripting	Attempt to build supported scripting DLC in debug mode with appropriate debug symbols so that the script debugger can be attached when loading DLC content at runtime. Only supported on windows platforms.
DebugScripting_UseConfig	Should the script debugging option be taken from the quick select build menu option (Tools -> DLC Toolkit -> Build Config -> Compilation).
Default	Default options.
DisableScripting	Prevent DLC content from being built with scripting support.
ForceRebuild	Should a total rebuild be forced disregarding any cached or incremental build artifacts. Will usually cause the build to take much longer but wil ensure that content is refreshed.
ForceRebuild_UseConfig	Should the force rebuild option be taken from the quick select build menu option (Tools -> DLC Toolkit -> Build Config -> Force Rebuild).
IncludeDisabledDLC	Should DLC profiles which are disabled be included in the build.
None	No options selected.

Class DLCBuildPipeline

Main API for building DLC content from script.

Inheritance

object

DLCBuildPipeline

Namespace: DLCToolkit.BuildTools

Assembly: DLCToolkit.BuildTools.dll

Syntax

```
[InitializeOnLoad]
public static class DLCBuildPipeline
```

Methods

BuildAllDLCContent(string, BuildTarget[], DLCBuildOptions)

Build all DLCProfile in the project with the specified options.

Declaration

```
public static DLCBuildResult BuildAllDLCContent(string outputFolder = null, BuildTarget[] platforms = null,
DLCBuildOptions buildOptions = DLCBuildOptions.Default)
```

Parameters

TYPE	NAME	DESCRIPTION
string	outputFolder	The optional output folder where the DLC content will be built. Use null for the default output folder of 'DLC Content/'
BuildTarget[]	platforms	An optional array of platforms to build for or null if all platforms should be built
DLCBuildOptions	buildOptions	Build option flags to define various build settings

Returns

TYPE	DESCRIPTION
DLCBuildResult	

BuildAllDLCShipWithGameContent(BuildTarget, bool, string, string, DLCBuildOptions)

Build all DLCProfile in the project for the target platform that are marked to be shipped with the game.

Declaration

```
public static DLCBuildResult BuildAllDLCShipWithGameContent(BuildTarget platformTarget, bool install = true,
string buildOutputFolder = null, string outputFolder = null, DLCBuildOptions buildOptions =
DLCBuildOptions.Default)
```

Parameters

TYPE	NAME	DESCRIPTION
BuildTarget	platformTarget	The platform that the DLC should be built for
bool	install	Should the built DLC content be installed in the shipped game output folder
string	buildOutputFolder	The output folder where the shipped game will be built
string	outputFolder	The optional output folder where the DLC content will be built. Use null for the default output folder of 'DLC Content/'
DLCBuildOptions	buildOptions	Build option flags to define various build settings

Returns

TYPE	DESCRIPTION
DLCBuildResult	

BuildDLCContent(DLCProfile, string, BuildTarget[], DLCBuildOptions)

Build the specified DLC content in the project for the provided DLCProfile. Only the profile provided will be build and all other profiles in the project will be ignored.

Declaration

```
public static DLCBuildResult BuildDLCContent(DLCProfile profile, string outputFolder = null, BuildTarget[]
platforms = null, DLCBuildOptions buildOptions = DLCBuildOptions.Default)
```

Parameters

TYPE	NAME	DESCRIPTION
DLCProfile	profile	The DLC profile to build
string	outputFolder	The optional output folder where the DLC content will be built. Use null for the default output folder of 'DLC Content/'
BuildTarget[]	platforms	An optional array of platforms to build for or null if all platforms should be built
DLCBuildOptions	buildOptions	Build option flags to define various build settings

Returns

TYPE	DESCRIPTION
DLCBuildResult	

Exceptions

TYPE	CONDITION
ArgumentNullException	profile is null

BuildDLCContent(DLCProfile[], string, BuildTarget[], DLCBuildOptions)

Build all DLC content in the project for only the specified array of DLCProfile. Only the profiles provided will be built and other profiles in the project will be ignored.

Declaration

```
public static DLCBuildResult BuildDLCContent(DLCProfile[] profiles, string outputFolder = null, BuildTarget[]
platforms = null, DLCBuildOptions buildOptions = DLCBuildOptions.Default)
```

Parameters

TYPE	NAME	DESCRIPTION
DLCProfile[]	profiles	An array of DLC profiles to build in this batch
string	outputFolder	The optional output folder where the DLC content will be built. Use null for the default output folder of 'DLC Content/'
BuildTarget[]	platforms	An optional array of platforms to build for or null if all platforms should be built
DLCBuildOptions	buildOptions	Build option flags to define various build settings

Returns

TYPE	DESCRIPTION
DLCBuildResult	

Exceptions

TYPE	CONDITION
ArgumentNullException	profiles is null

BuildDLCShipWithGameContent(DLCProfile, BuildTarget, bool, string, string, DLCBuildOptions)

Build the provided DLCProfile for the target platform that are marked to be shipped with the game.

Declaration

```
public static DLCBuildResult BuildDLCShipWithGameContent(DLCProfile profile, BuildTarget platformTarget, bool install = true, string buildOutputFolder = null, string outputFolder = null, DLCBuildOptions buildOptions = DLCBuildOptions.Default)
```

Parameters

TYPE	NAME	DESCRIPTION
DLCProfile	profile	The DLC profile to build
BuildTarget	platformTarget	The platform that the DLC should be built for
bool	install	Should the built DLC content be installed in the shipped game output folder
string	buildOutputFolder	The output folder where the shipped game will be built
string	outputFolder	The optional output folder where the DLC content will be built. Use null for the default output folder of 'DLC Content/'
DLCBuildOptions	buildOptions	Build option flags to define various build settings

Returns

TYPE	DESCRIPTION
DLCBuildResult	

BuildDLCShipWithGameContent(DLCProfile[], BuildTarget, bool, string, string, DLCBuildOptions)

Build all DLCProfile provided for the target platform that are marked to be shipped with the game.

Declaration

```
public static DLCBuildResult BuildDLCShipWithGameContent(DLCProfile[] profiles, BuildTarget platformTarget, bool install = true, string buildOutputFolder = null, string outputFolder = null, DLCBuildOptions buildOptions = DLCBuildOptions.Default)
```

Parameters

TYPE	NAME	DESCRIPTION
DLCProfile[]	profiles	A collection of DLC profiles that should be built
BuildTarget	platformTarget	The platform that the DLC should be built for
bool	install	Should the built DLC content be installed in the shipped game output folder

TYPE	NAME	DESCRIPTION
string	buildOutputFolder	The output folder where the shipped game will be built
string	outputFolder	The optional output folder where the DLC content will be built. Use null for the default output folder of 'DLC Content/'
DLCBuildOptions	buildOptions	Build option flags to define various build settings

Returns

TYPE	DESCRIPTION
DLCBuildResult	

GetAllDLCProfiles(BuildTarget[], bool)

Get an array of all DLCProfile in the project which is enabled for the specified platforms and is enabled unless `includeDisabledDlc` is disabled. Platforms can be null if you want to get all DLC for all platforms.

Declaration

```
public static DLCProfile[] GetAllDLCProfiles(BuildTarget[] platforms = null, bool includeDisabledDlc = false)
```

Parameters

TYPE	NAME	DESCRIPTION
BuildTarget[]	platforms	An optional array of platforms that should be found or null for all platforms
bool	includeDisabledDlc	Should the result include DLC content that is not enabled for build

Returns

TYPE	DESCRIPTION
DLCProfile[]	An array of DLC profiles matching the search criteria or an empty array if no results are found

GetDLCProfile(string)

Try to get the DLCProfile from the project with the specified unique key. This will check all platform profiles in the DLC for a matching unique key.

Declaration

```
public static DLCProfile GetDLCProfile(string uniqueKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key for the DLC

Returns

TYPE	DESCRIPTION
DLCProfile	A matching profile if found or null

GetDLCProfile(string, Version)

Try to get the DLCProfile from the project with the specified name and optional version.

Declaration

```
public static DLCProfile GetDLCProfile(string name, Version version = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	name	The name of the DLC to find
Version	version	The optional version of the DLC to find if an exact match is required

Returns

TYPE	DESCRIPTION
DLCProfile	A matching profile if found or null

Class DLCBuildResult

Contains information about which aspects of the DLC build request were successful and which were not.

Inheritance

object

DLCBuildResult

Namespace: DLCToolkit.BuildTools

Assembly: DLCToolkit.BuildTools.dll

Syntax

```
public sealed class DLCBuildResult
```

Properties

AllSuccessful

Were all build tasks that were started completed successfully. No errors no failed builds.

Declaration

```
public bool AllSuccessful { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

BuildFailedCount

The total number of build tasks that failed.

Declaration

```
public int BuildFailedCount { get; }
```

Property Value

TYPE	DESCRIPTION
int	

BuildStartTime

The time that the DLC build request started.

Declaration

```
public DateTime BuildStartTime { get; }
```

Property Value

TYPE	DESCRIPTION
DateTime	

BuildSuccessCount

The total number of build taks that completed successfully.

Declaration

```
public int BuildSuccessCount { get; }
```

Property Value

TYPE	DESCRIPTION
int	

BuildTaskCount

The total number of build tasks that were run.

Declaration

```
public int BuildTaskCount { get; }
```

Property Value

TYPE	DESCRIPTION
int	

BuildTasks

All build tasks that were included in the request whether they were successful or not.

Declaration

```
public IReadOnlyList<DLCBuildTask> BuildTasks { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<DLCBuildTask>	

ElapsedBuildTime

The amount of time that the entire build batch took to complete.

Declaration

```
public TimeSpan ElapsedBuildTime { get; }
```

Property Value

TYPE	DESCRIPTION
TimeSpan	

Methods

GetBuildTasksForPlatform(BuildTarget)

Get all build tasks for the target platform.

Declaration

```
public IEnumerable<DLCBuildTask> GetBuildTasksForPlatform(BuildTarget target)
```

Parameters

TYPE	NAME	DESCRIPTION
BuildTarget	target	The platform of interest

Returns

TYPE	DESCRIPTION
IEnumerable<DLCBuildTask>	An enumerable of build tasks for the target platform

GetFailedBuildTasks()

Get all failed build tasks.

Declaration

```
public IEnumerable<DLCBuildTask> GetFailedBuildTasks()
```

Returns

TYPE	DESCRIPTION
IEnumerable<DLCBuildTask>	An enumerable of failed build tasks

GetFailedBuildTasksForPlatform(BuildTarget)

Get all build tasks that failed for the target platform.

Declaration

```
public IEnumerable<DLCBuildTask> GetFailedBuildTasksForPlatform(BuildTarget target)
```

Parameters

TYPE	NAME	DESCRIPTION
BuildTarget	target	The platform of interest

Returns

TYPE	DESCRIPTION
IEnumerable<DLCBuildTask>	An enumerable of failed build tasks for the target platform

GetSuccessfulBuildTasks()

Get all successful build tasks.

Declaration

```
public IEnumerable<DLCBuildTask> GetSuccessfulBuildTasks()
```

Returns

TYPE	DESCRIPTION
IEnumerable<DLCBuildTask>	An enumerable of successful build tasks

GetSuccessfulBuildTasksForPlatform(BuildTarget)

Get all build tasks that completed successfully for the target platform.

Declaration

```
public IEnumerable<DLCBuildTask> GetSuccessfulBuildTasksForPlatform(BuildTarget target)
```

Parameters

TYPE	NAME	DESCRIPTION
BuildTarget	target	The platform of interest

Returns

TYPE	DESCRIPTION
IEnumerable<DLCBuildTask>	An enumerable of successful build tasks for the target platform

HasBuildTasksForPlatform(BuildTarget)

Check if any build tasks were run for the specified platform.

Declaration

```
public bool HasBuildTasksForPlatform(BuildTarget target)
```

Parameters

TYPE	NAME	DESCRIPTION
BuildTarget	target	The platform to check

Returns

TYPE	DESCRIPTION
bool	True if one or more build tasks were run for the target platform whether successful or not

Class DLCBuildTask

Represents an individual build task of a DLC profile for a specific build platform.

Inheritance

object

DLCBuildTask

Namespace: DLCToolkit.BuildTools

Assembly: DLCToolkit.BuildTools.dll

Syntax

```
public sealed class DLCBuildTask
```

Properties

OutputPath

The output path for this build if successful or empty string if not.

Declaration

```
public string OutputPath { get; }
```

Property Value

TYPE	DESCRIPTION
string	

PlatformProfile

The DLC platform profile for this build.

Declaration

```
public DLCPlatformProfile PlatformProfile { get; }
```

Property Value

TYPE	DESCRIPTION
DLCPlatformProfile	

Profile

The DLC profile for this build.

Declaration

```
public DLCProfile Profile { get; }
```

Property Value

TYPE	DESCRIPTION
DLCProfile	

Success

Was the build successful.

Declaration

```
public bool Success { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Class DLCPostBuildAttribute

Used to mark a post DLC build event called just after DLC has been built. Must implement DLCBuildEvent base class.

Inheritance

object
Attribute
DLCPostBuildAttribute

Implements

_Attribute

Inherited Members

Attribute.Equals(object)
Attribute.GetCustomAttribute(Assembly, Type)
Attribute.GetCustomAttribute(Assembly, Type, bool)
Attribute.GetCustomAttribute(MemberInfo, Type)
Attribute.GetCustomAttribute(MemberInfo, Type, bool)
Attribute.GetCustomAttribute(Module, Type)
Attribute.GetCustomAttribute(Module, Type, bool)
Attribute.GetCustomAttribute(ParameterInfo, Type)
Attribute.GetCustomAttribute(ParameterInfo, Type, bool)
Attribute.GetCustomAttributes(Assembly)
Attribute.GetCustomAttributes(Assembly, bool)
Attribute.GetCustomAttributes(Assembly, Type)
Attribute.GetCustomAttributes(Assembly, Type, bool)
Attribute.GetCustomAttributes(MemberInfo)
Attribute.GetCustomAttributes(MemberInfo, bool)
Attribute.GetCustomAttributes(MemberInfo, Type)
Attribute.GetCustomAttributes(MemberInfo, Type, bool)
Attribute.GetCustomAttributes(Module)
Attribute.GetCustomAttributes(Module, bool)
Attribute.GetCustomAttributes(Module, Type)
Attribute.GetCustomAttributes(Module, Type, bool)
Attribute.GetCustomAttributes(ParameterInfo)
Attribute.GetCustomAttributes(ParameterInfo, bool)
Attribute.GetCustomAttributes(ParameterInfo, Type)
Attribute.GetCustomAttributes(ParameterInfo, Type, bool)
Attribute.GetHashCode()
Attribute.IsDefaultAttribute()
Attribute.IsDefined(Assembly, Type)
Attribute.IsDefined(Assembly, Type, bool)
Attribute.IsDefined(MemberInfo, Type)
Attribute.IsDefined(MemberInfo, Type, bool)
Attribute.IsDefined(Module, Type)
Attribute.IsDefined(Module, Type, bool)
Attribute.IsDefined(ParameterInfo, Type)
Attribute.IsDefined(ParameterInfo, Type, bool)
Attribute.Match(object)
Attribute.TypeId

Namespace: DLCToolkit.BuildTools

Assembly: DLCToolkit.BuildTools.dll

Syntax

```
[AttributeUsage(AttributeTargets.Class)]
public sealed class DLCPostBuildAttribute : Attribute, _Attribute
```

Implements

_Attribute

Class DLCPostBuildPlatformProfileAttribute

Used to mark a post DLC build event called just after a specific DLC profile platform has been built. Must implement DLCBuildPlatformProfileResultEvent base class.

Inheritance

object
Attribute
DLCPostBuildPlatformProfileAttribute

Implements

_Attribute

Inherited Members

Attribute.Equals(object)
Attribute.GetCustomAttribute(Assembly, Type)
Attribute.GetCustomAttribute(Assembly, Type, bool)
Attribute.GetCustomAttribute(MemberInfo, Type)
Attribute.GetCustomAttribute(MemberInfo, Type, bool)
Attribute.GetCustomAttribute(Module, Type)
Attribute.GetCustomAttribute(Module, Type, bool)
Attribute.GetCustomAttribute(ParameterInfo, Type)
Attribute.GetCustomAttribute(ParameterInfo, Type, bool)
Attribute.GetCustomAttributes(Assembly)
Attribute.GetCustomAttributes(Assembly, bool)
Attribute.GetCustomAttributes(Assembly, Type)
Attribute.GetCustomAttributes(Assembly, Type, bool)
Attribute.GetCustomAttributes(MemberInfo)
Attribute.GetCustomAttributes(MemberInfo, bool)
Attribute.GetCustomAttributes(MemberInfo, Type)
Attribute.GetCustomAttributes(MemberInfo, Type, bool)
Attribute.GetCustomAttributes(Module)
Attribute.GetCustomAttributes(Module, bool)
Attribute.GetCustomAttributes(Module, Type)
Attribute.GetCustomAttributes(Module, Type, bool)
Attribute.GetCustomAttributes(ParameterInfo)
Attribute.GetCustomAttributes(ParameterInfo, bool)
Attribute.GetCustomAttributes(ParameterInfo, Type)
Attribute.GetCustomAttributes(ParameterInfo, Type, bool)
Attribute.GetHashCode()
Attribute.IsDefaultAttribute()
Attribute.IsDefined(Assembly, Type)
Attribute.IsDefined(Assembly, Type, bool)
Attribute.IsDefined(MemberInfo, Type)
Attribute.IsDefined(MemberInfo, Type, bool)
Attribute.IsDefined(Module, Type)
Attribute.IsDefined(Module, Type, bool)
Attribute.IsDefined(ParameterInfo, Type)
Attribute.IsDefined(ParameterInfo, Type, bool)
Attribute.Match(object)
Attribute.TypeId

Namespace: DLCToolkit.BuildTools

Assembly: DLCToolkit.BuildTools.dll

Syntax

```
[AttributeUsage(AttributeTargets.Class)]
public sealed class DLCPostBuildPlatformProfileAttribute : Attribute, _Attribute
```

Implements

_Attribute

Class DLCPreBuildAttribute

Used to mark a pre DLC build event called just before DLC will be built. Must implement DLCBuildEvent base class.

Inheritance

object
Attribute
DLCPreBuildAttribute

Implements

_Attribute

Inherited Members

Attribute.Equals(object)
Attribute.GetCustomAttribute(Assembly, Type)
Attribute.GetCustomAttribute(Assembly, Type, bool)
Attribute.GetCustomAttribute(MemberInfo, Type)
Attribute.GetCustomAttribute(MemberInfo, Type, bool)
Attribute.GetCustomAttribute(Module, Type)
Attribute.GetCustomAttribute(Module, Type, bool)
Attribute.GetCustomAttribute(ParameterInfo, Type)
Attribute.GetCustomAttribute(ParameterInfo, Type, bool)
Attribute.GetCustomAttributes(Assembly)
Attribute.GetCustomAttributes(Assembly, bool)
Attribute.GetCustomAttributes(Assembly, Type)
Attribute.GetCustomAttributes(Assembly, Type, bool)
Attribute.GetCustomAttributes(MemberInfo)
Attribute.GetCustomAttributes(MemberInfo, bool)
Attribute.GetCustomAttributes(MemberInfo, Type)
Attribute.GetCustomAttributes(MemberInfo, Type, bool)
Attribute.GetCustomAttributes(Module)
Attribute.GetCustomAttributes(Module, bool)
Attribute.GetCustomAttributes(Module, Type)
Attribute.GetCustomAttributes(Module, Type, bool)
Attribute.GetCustomAttributes(ParameterInfo)
Attribute.GetCustomAttributes(ParameterInfo, bool)
Attribute.GetCustomAttributes(ParameterInfo, Type)
Attribute.GetCustomAttributes(ParameterInfo, Type, bool)
Attribute.GetHashCode()
Attribute.IsDefaultAttribute()
Attribute.IsDefined(Assembly, Type)
Attribute.IsDefined(Assembly, Type, bool)
Attribute.IsDefined(MemberInfo, Type)
Attribute.IsDefined(MemberInfo, Type, bool)
Attribute.IsDefined(Module, Type)
Attribute.IsDefined(Module, Type, bool)
Attribute.IsDefined(ParameterInfo, Type)
Attribute.IsDefined(ParameterInfo, Type, bool)
Attribute.Match(object)
Attribute.TypeId

Namespace: DLCToolkit.BuildTools

Assembly: DLCToolkit.BuildTools.dll

Syntax

```
[AttributeUsage(AttributeTargets.Class)]
public sealed class DLCPreBuildAttribute : Attribute, _Attribute
```

Implements

_Attribute

Class DLCPreBuildPlatformProfileAttribute

Used to mark a pre DLC build event called just before a specific DLC profile platform will be built. Must implement DLCBuildPlatformProfileEvent base class.

Inheritance

object
Attribute
DLCPreBuildPlatformProfileAttribute

Implements

_Attribute

Inherited Members

Attribute.Equals(object)
Attribute.GetCustomAttribute(Assembly, Type)
Attribute.GetCustomAttribute(Assembly, Type, bool)
Attribute.GetCustomAttribute(MemberInfo, Type)
Attribute.GetCustomAttribute(MemberInfo, Type, bool)
Attribute.GetCustomAttribute(Module, Type)
Attribute.GetCustomAttribute(Module, Type, bool)
Attribute.GetCustomAttribute(ParameterInfo, Type)
Attribute.GetCustomAttribute(ParameterInfo, Type, bool)
Attribute.GetCustomAttributes(Assembly)
Attribute.GetCustomAttributes(Assembly, bool)
Attribute.GetCustomAttributes(Assembly, Type)
Attribute.GetCustomAttributes(Assembly, Type, bool)
Attribute.GetCustomAttributes(MemberInfo)
Attribute.GetCustomAttributes(MemberInfo, bool)
Attribute.GetCustomAttributes(MemberInfo, Type)
Attribute.GetCustomAttributes(MemberInfo, Type, bool)
Attribute.GetCustomAttributes(Module)
Attribute.GetCustomAttributes(Module, bool)
Attribute.GetCustomAttributes(Module, Type)
Attribute.GetCustomAttributes(Module, Type, bool)
Attribute.GetCustomAttributes(ParameterInfo)
Attribute.GetCustomAttributes(ParameterInfo, bool)
Attribute.GetCustomAttributes(ParameterInfo, Type)
Attribute.GetCustomAttributes(ParameterInfo, Type, bool)
Attribute.GetHashCode()
Attribute.IsDefaultAttribute()
Attribute.IsDefined(Assembly, Type)
Attribute.IsDefined(Assembly, Type, bool)
Attribute.IsDefined(MemberInfo, Type)
Attribute.IsDefined(MemberInfo, Type, bool)
Attribute.IsDefined(Module, Type)
Attribute.IsDefined(Module, Type, bool)
Attribute.IsDefined(ParameterInfo, Type)
Attribute.IsDefined(ParameterInfo, Type, bool)
Attribute.Match(object)
Attribute.TypeId

Namespace: DLCToolkit.BuildTools

Assembly: DLCToolkit.BuildTools.dll

Syntax

```
[AttributeUsage(AttributeTargets.Class)]
public sealed class DLCPreBuildPlatformProfileAttribute : Attribute, _Attribute
```

Implements

_Attribute

Class DLCPreBuildProfileAttribute

Used to mark a pre DLC build event called just before a specific DLC profile will be built. Must implement DLCBuildProfileEvent base class.

Inheritance

object
Attribute
DLCPreBuildProfileAttribute

Implements

_Attribute

Inherited Members

Attribute.Equals(object)
Attribute.GetCustomAttribute(Assembly, Type)
Attribute.GetCustomAttribute(Assembly, Type, bool)
Attribute.GetCustomAttribute(MemberInfo, Type)
Attribute.GetCustomAttribute(MemberInfo, Type, bool)
Attribute.GetCustomAttribute(Module, Type)
Attribute.GetCustomAttribute(Module, Type, bool)
Attribute.GetCustomAttribute(ParameterInfo, Type)
Attribute.GetCustomAttribute(ParameterInfo, Type, bool)
Attribute.GetCustomAttributes(Assembly)
Attribute.GetCustomAttributes(Assembly, bool)
Attribute.GetCustomAttributes(Assembly, Type)
Attribute.GetCustomAttributes(Assembly, Type, bool)
Attribute.GetCustomAttributes(MemberInfo)
Attribute.GetCustomAttributes(MemberInfo, bool)
Attribute.GetCustomAttributes(MemberInfo, Type)
Attribute.GetCustomAttributes(MemberInfo, Type, bool)
Attribute.GetCustomAttributes(Module)
Attribute.GetCustomAttributes(Module, bool)
Attribute.GetCustomAttributes(Module, Type)
Attribute.GetCustomAttributes(Module, Type, bool)
Attribute.GetCustomAttributes(ParameterInfo)
Attribute.GetCustomAttributes(ParameterInfo, bool)
Attribute.GetCustomAttributes(ParameterInfo, Type)
Attribute.GetCustomAttributes(ParameterInfo, Type, bool)
Attribute.GetHashCode()
Attribute.IsDefaultAttribute()
Attribute.IsDefined(Assembly, Type)
Attribute.IsDefined(Assembly, Type, bool)
Attribute.IsDefined(MemberInfo, Type)
Attribute.IsDefined(MemberInfo, Type, bool)
Attribute.IsDefined(Module, Type)
Attribute.IsDefined(Module, Type, bool)
Attribute.IsDefined(ParameterInfo, Type)
Attribute.IsDefined(ParameterInfo, Type, bool)
Attribute.Match(object)
Attribute.TypeId

Namespace: DLCToolkit.BuildTools

Assembly: DLCToolkit.BuildTools.dll

Syntax

```
[AttributeUsage(AttributeTargets.Class)]
public sealed class DLCPreBuildProfileAttribute : Attribute, _Attribute
```

Implements

_Attribute

Namespace DLCToolkit.BuildTools.Events

Classes

DLCBuildEvent

Implement this event base when you want to hook into DLC pre or post build events. Use with DLCPreBuildAttribute or DLCPostBuildAttribute.

DLCBuildPlatformProfileEvent

Implement this event base when you want to hook into DLC build profile platform event. Use with DLCPreBuildPlatformProfileAttribute.

DLCBuildPlatformProfileResultEvent

Implement this event base when you want to hook into the DLC build profile platform completed event. Use with DLCPostBuildPlatformProfileAttribute

DLCBuildProfileEvent

Implement this event base when you want to hook into DLC build profile event. Use with DLCPreBuildProfileAttribute.

Class DLCBuildEvent

Implement this event base when you want to hook into DLC pre or post build events. Use with DLCPreBuildAttribute or DLCPostBuildAttribute.

Inheritance

object

DLCBuildEvent

Namespace: DLCToolkit.BuildTools.Events

Assembly: DLCToolkit.BuildTools.dll

Syntax

```
public abstract class DLCBuildEvent
```

Methods

OnBuildEvent()

Called while building DLC content.

Declaration

```
public abstract void OnBuildEvent()
```

Class DLCBuildPlatformProfileEvent

Implement this event base when you want to hook into DLC build profile platform event. Use with DLCPreBuildPlatformProfileAttribute.

Inheritance

object

DLCBuildPlatformProfileEvent

Namespace: DLCToolkit.BuildTools.Events

Assembly: DLCToolkit.BuildTools.dll

Syntax

```
public abstract class DLCBuildPlatformProfileEvent
```

Methods

OnBuildProfileEvent(DLCProfile, DLCPlatformProfile)

Called while building DLC content for the specified profile and platform.

Declaration

```
public abstract void OnBuildProfileEvent(DLCProfile profile, DLCPlatformProfile platformProfile)
```

Parameters

TYPE	NAME	DESCRIPTION
DLCProfile	profile	The profile that is currently being built
DLCPlatformProfile	platformProfile	The platform that is currently being build

Class DLCBuildPlatformProfileResultEvent

Implement this event base when you want to hook into the DLC build profile platform completed event. Use with DLCPostBuildPlatformProfileAttribute

Inheritance

object

DLCBuildPlatformProfileResultEvent

Namespace: DLCToolkit.BuildTools.Events

Assembly: DLCToolkit.BuildTools.dll

Syntax

```
public abstract class DLCBuildPlatformProfileResultEvent
```

Methods

OnBuildProfileEvent(DLCProfile, DLCPlatformProfile, bool, string)

Called while building DLC content for the specified profile and platform with result information.

Declaration

```
public abstract void OnBuildProfileEvent(DLCProfile profile, DLCPlatformProfile platformProfile, bool success,
string output)
```

Parameters

TYPE	NAME	DESCRIPTION
DLCProfile	profile	The profile that was built
DLCPlatformProfile	platformProfile	The platform that was built
bool	success	Did the build succeed
string	output	The output path of the DLC content file

Class DLCBuildProfileEvent

Implement this event base when you want to hook into DLC build profile event. Use with DLCPreBuildProfileAttribute.

Inheritance

object

DLCBuildProfileEvent

Namespace: DLCToolkit.BuildTools.Events

Assembly: DLCToolkit.BuildTools.dll

Syntax

```
public abstract class DLCBuildProfileEvent
```

Methods

OnBuildProfileEvent(DLCProfile)

Called while building DLC content for the specified profile.

Declaration

```
public abstract void OnBuildProfileEvent(DLCProfile profile)
```

Parameters

TYPE	NAME	DESCRIPTION
DLCProfile	profile	The profile that is currently being built

Namespace DLCToolkit.DRM

Classes

DefaultDRMServiceProvider

A DRM service provider that supports editor local DRM, steamworks, and google play DRM services.

LocalDirectoryDRM

A virtual DRM provider mounted to a local directory which offers DRM like ability without any verification. Supports listing all DLC unique keys that are available in the specified directory. Does not support ownership verification and will simply report that any DLC in the specified folder is owned by the user and can be loaded. Supported on any platform with IO support (Not WebGL for example) and may be useful for free DLC or expansion packs that can be simply dropped into a game folder and be auto-detected. You can also manually handle this sort of setup using the DLCDirectory instead, which is what this DRM provider is using internally, but it may offer you more control for things like install/uninstall events.

Interfaces

IDRMProvider

Main API to interact with a DLC DRM service independent of the build platform.

IDRMServiceProvider

Provides access to the IDRMProvider for the current platform.

Class DefaultDRMServiceProvider

A DRM service provider that supports editor local DRM, steamworks, and google play DRM services.

Inheritance

object

DefaultDRMServiceProvider

Implements

IDRMServiceProvider

Namespace: DLCToolkit.DRM

Assembly: DLCToolkit.dll

Syntax

```
public sealed class DefaultDRMServiceProvider : IDRMServiceProvider
```

Fields

editorContentPath

The path where DLC content will be stored in editor mode. The editor will scan this folder for DLC content automatically.

Declaration

```
public string editorContentPath
```

Field Value

TYPE	DESCRIPTION
string	

Methods

GetDRMProvider()

Get the DRM provider for the current platform.

Declaration

```
public IDRMProvider GetDRMProvider()
```

Returns

TYPE	DESCRIPTION
IDRMProvider	

Exceptions

TYPE	CONDITION
NotSupportedException	

Implements

IDRMServiceProvider

Interface IDRMProvider

Main API to interact with a DLC DRM service independent of the build platform.

Namespace: DLCToolkit.DRM

Assembly: DLCToolkit.dll

Syntax

```
public interface IDRMProvider
```

Properties

DLCUniqueKeysAsync

Get all unique id keys for all DLC published via the DRM platform (Steamworks DLC for example). It is allowed for this property to return an empty array or only a partial array of the potentially available DLC Contents. IsDLCAvailableAsync(IDLCAsyncProvider, string) will be used to determine truly whether DLC content is valid and available at any given time, even if this property does not list it. Can throw a NotSupportedException if the DRM platform does not support listing contents.

Declaration

```
DLCAsync<string[]> DLCUniqueKeysAsync { get; }
```

Property Value

TYPE	DESCRIPTION
DLCAsync<string[]>	

Methods

GetDLCStream(string)

Attempt to get the stream provider for the DLC to allow loading.

Declaration

```
DLCStreamProvider GetDLCStream(string uniqueKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key of the dlc

Returns

TYPE	DESCRIPTION
DLCStreamProvider	The stream provider for the dlc if installed or null if it is not available

IsDLCAvailableAsync(IDLCAsyncProvider, string)

Check if the specified DLC is purchased and installed. Some providers may need to make a web request to check for purchased DLC, so this operations must be async.

Declaration

```
DLCAsync<bool> IsDLCAvailableAsync(IDLCAsyncProvider asyncProvider, string uniqueKey)
```

Parameters

TYPE	NAME	DESCRIPTION
IDLCAsyncProvider	asyncProvider	The async provider to allow async tasks to be started
string	uniqueKey	The unique key of the dlc

Returns

TYPE	DESCRIPTION
DLCAsync<bool>	True if the dlc is installed or false if not

RequestInstallDLCAsync(IDLCAsyncProvider, string)

Request that the dlc with the provided unique key is installed onto the system if it is available to the user.

Declaration

```
DLCAsync RequestInstallDLCAsync(IDLCAsyncProvider asyncProvider, string uniqueKey)
```

Parameters

TYPE	NAME	DESCRIPTION
IDLCAsyncProvider	asyncProvider	The async provider to allow async tasks to be started
string	uniqueKey	The unique key of the dlc

Returns

TYPE	DESCRIPTION
DLCAsync	

RequestUninstallDLC(string)

Request that the dlc with the provided unique key is uninstalled from the system if it is currently installed.

Declaration

```
void RequestUninstallDLC(string uniqueKey)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key of the dlc

TrackDLCUsage(string, bool)

Allow for progress tracking/hours tracking from the drm service by setting which dlc is currently being used.

Declaration

```
void TrackDLCUsage(string uniqueKey, bool isInUse)
```

Parameters

TYPE	NAME	DESCRIPTION
string	uniqueKey	The unique key of the dlc
bool	isInUse	True if the game is currently using the dlc or false if not. Used to enable or disable progress tracking

Interface IDRMServiceProvider

Provides access to the IDRMProvider for the current platform.

Namespace: DLCToolkit.DRM

Assembly: DLCToolkit.dll

Syntax

```
public interface IDRMServiceProvider
```

Methods

GetDRMProvider()

Try to get the IDRMProvider for this platform.

Declaration

```
IDRMProvider GetDRMProvider()
```

Returns

TYPE	DESCRIPTION
IDRMProvider	A DRM provider or null if DRM is not supported on this platform

Class LocalDirectoryDRM

A virtual DRM provider mounted to a local directory which offers DRM like ability without any verification. Supports listing all DLC unique keys that are available in the specified directory. Does not support ownership verification and will simply report that any DLC in the specified folder is owned by the user and can be loaded. Supported on any platform with IO support (Not WebGL for example) and may be useful for free DLC or expansion packs that can be simply dropped into a game folder and be auto-detected. You can also manually handle this sort of setup using the DLCDirectory instead, which is what this DRM provider is using internally, but it may offer you more control for things like install/uninstall events.

Inheritance

object

LocalDirectoryDRM

Implements

IDRMProvider

Namespace: DLCToolkit.DRM

Assembly: DLCToolkit.dll

Syntax

```
public sealed class LocalDirectoryDRM : IDRMProvider
```

Constructors

LocalDirectoryDRM(string)

Create a new instance from the target folder. Note that the specified folder must already exist or an exception will be thrown.

Declaration

```
public LocalDirectoryDRM(string localFolderPath)
```

Parameters

TYPE	NAME	DESCRIPTION
string	localFolderPath	The path where DLC content may be stored

Exceptions

TYPE	CONDITION
ArgumentException	The folder path is null or empty
DirectoryNotFoundException	The folder path does not exist

Implements

IDRMProvider

Namespace DLCToolkit.EditorTools

Classes

DLCPlatformProfileInspector

Class DLCPlatformProfileInspector

Inheritance

object

Object

ScriptableObject

Editor

DLCPlatformProfileInspector

Inherited Members

Editor.SaveChanges()

Editor.DiscardChanges()

Editor.CreateEditorWithContext(Object[], Object, Type)

Editor.CreateEditorWithContext(Object[], Object)

Editor.CreateCachedEditorWithContext(Object, Object, Type, ref Editor)

Editor.CreateCachedEditorWithContext(Object[], Object, Type, ref Editor)

Editor.CreateCachedEditor(Object, Type, ref Editor)

Editor.CreateCachedEditor(Object[], Type, ref Editor)

Editor.CreateEditor(Object)

Editor.CreateEditor(Object, Type)

Editor.CreateEditor(Object[])

Editor.CreateEditor(Object[], Type)

Editor.DrawDefaultInspector()

Editor.Repaint()

Editor.CreateInspectorGUI()

Editor.RequiresConstantRepaint()

Editor.DrawHeader()

Editor.DrawFoldoutInspector(Object, ref Editor)

Editor.HasPreviewGUI()

Editor.GetPreviewTitle()

Editor.RenderStaticPreview(string, Object[], int, int)

Editor.OnPreviewGUI(Rect, GUIStyle)

Editor.OnInteractivePreviewGUI(Rect, GUIStyle)

Editor.OnPreviewSettings()

Editor.GetInfoString()

Editor.DrawPreview(Rect)

Editor.ReloadPreviewInstances()

Editor.UseDefaultMargins()

Editor.MoveNextTarget()

Editor.ResetTarget()

Editor.hasUnsavedChanges

Editor.saveChangesMessage

Editor.target

Editor.targets

Editor.serializedObject

Editor.finishedDefaultHeaderGUI

ScriptableObject.SetDirty()

ScriptableObject.CreateInstance(string)

ScriptableObject.CreateInstance(Type)

ScriptableObject.CreateInstance<T>()

Object.GetInstanceID()

Object.GetHashCode()

Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.FindObjectsByType(Type, FindObjectsSortMode)
Object.FindObjectsByType(Type, FindObjectsInactive, FindObjectsSortMode)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsByType<T>(FindObjectsSortMode)
Object.FindObjectsOfType<T>(bool)
Object.FindObjectsByType<T>(FindObjectsInactive, FindObjectsSortMode)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindFirstObjectByType<T>()
Object.FindAnyObjectByType<T>()
Object.FindFirstObjectByType<T>(FindObjectsInactive)
Object.FindAnyObjectByType<T>(FindObjectsInactive)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindFirstObjectByType(Type)
Object.FindAnyObjectByType(Type)
Object.FindObjectOfType(Type, bool)
Object.FindFirstObjectByType(Type, FindObjectsInactive)
Object.FindAnyObjectByType(Type, FindObjectsInactive)
Object.ToString()
Object.name
Object.hideFlags

Namespace: DLCToolkit.EditorTools

Assembly: DLCToolkit.EditorTools.dll

Syntax

```
[CustomEditor(typeof(DLCProfile))]
public sealed class DLCPlatformProfileInspector : Editor
```

Methods

OnInspectorGUI()

Implement this function to make a custom inspector.

Declaration

```
public override void OnInspectorGUI()
```

Overrides

UnityEditor.Editor.OnInspectorGUI()

Namespace DLCToolkit.Profile

Classes

DLCCustomIcon

Represents a custom icon entry for a DLC profile.

DLCPlatformProfile

Represents a DLC build profile for a specific build target platform.

DLCPlatformProfileAndroid

DLCProfile

Represents a build profile for a specified DLC containing most preferences and options relating to the build process.

Enums

ShipWithGameDirectory

Class DLCCustomIcon

Represents a custom icon entry for a DLC profile.

Inheritance

object

DLCCustomIcon

Namespace: DLCToolkit.Profile

Assembly: DLCToolkit.Profile.dll

Syntax

```
[Serializable]
public sealed class DLCCustomIcon
```

Properties

CustomIcon

The assigned texture for the custom icon.

Declaration

```
public Texture2D CustomIcon { get; }
```

Property Value

TYPE	DESCRIPTION
Texture2D	

CustomKey

The unique key for the custom icon.

Declaration

```
public string CustomKey { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Class DLCPlatformProfile

Represents a DLC build profile for a specific build target platform.

Inheritance

object

DLCPlatformProfile

DLCPlatformProfileAndroid

Namespace: DLCToolkit.Profile

Assembly: DLCToolkit.Profile.dll

Syntax

```
[Serializable]
public class DLCPlatformProfile
```

Fields

friendlyPlatformNames

An array of supported platform names in an easily readable format.

Declaration

```
public static readonly string[] friendlyPlatformNames
```

Field Value

TYPE	DESCRIPTION
string[]	

Properties

DLCExtension

The file extension for this platform DLC.

Declaration

```
public string DLCExtension { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

DlcUniqueKey

The unique key for this platform DLC.

Declaration

```
public string DlcUniqueKey { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

Enabled

Is this platform enabled for build.

Declaration

```
public bool Enabled { get; set; }
```

Property Value

TYPE	DESCRIPTION
bool	

Platform

Get the build target for this platform.

Declaration

```
public BuildTarget Platform { get; }
```

Property Value

TYPE	DESCRIPTION
BuildTarget	

PlatformDefines

Get an array of define symbols specific to this platform when compiling scripts.

Declaration

```
public string[] PlatformDefines { get; }
```

Property Value

TYPE	DESCRIPTION
string[]	

PlatformFriendlyName

Get the easily readable platform name for this platform profile.

Declaration

```
public string PlatformFriendlyName { get; }
```

Property Value

TYPE	DESCRIPTION
string	

PlatformName

Get the name of this platform.

Declaration

```
public string PlatformName { get; }
```

Property Value

TYPE	DESCRIPTION
string	

PreloadSceneAssets

Should the DLC content preload scene assets at load time for this platform.

Declaration

```
public bool PreloadSceneAssets { get; set; }
```

Property Value

TYPE	DESCRIPTION
bool	

PreloadSharedAssets

Should the DLC content preload shared assets at load time for this platform.

Declaration

```
public bool PreloadSharedAssets { get; set; }
```

Property Value

TYPE	DESCRIPTION
bool	

RuntimePlatform

Get the runtime target for this platform.

Declaration

```
public RuntimePlatform RuntimePlatform { get; }
```

Property Value

TYPE	DESCRIPTION
RuntimePlatform	

ShipWithGame

Should the DLC content be included in the built game. Note that it is possible to include DLC that is not usable until the user has purchased via DRM.

Declaration

```
public bool ShipWithGame { get; set; }
```

Property Value

TYPE	DESCRIPTION
bool	

ShipWithGameDirectory

The directory relative where the DLC content should be placed.

Declaration

```
public ShipWithGameDirectory ShipWithGameDirectory { get; set; }
```

Property Value

TYPE	DESCRIPTION
ShipWithGameDirectory	

ShipWithGamePath

The path where the DLC content should be placed.

Declaration

```
public string ShipWithGamePath { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

StrictBuild

Should the DLC content be built in strict mode for this platform.

Declaration

```
public bool StrictBuild { get; set; }
```

Property Value

TYPE	DESCRIPTION
bool	

UseCompression

Should the DLC content be built with compression for this platform.

Declaration

```
public bool UseCompression { get; set; }
```

Property Value

TYPE	DESCRIPTION
bool	

Methods

GetFriendlyPlatformName(BuildTarget)

Get the platform friendly name from the specified build target.

Declaration

```
public static string GetFriendlyPlatformName(BuildTarget platform)
```

Parameters

TYPE	NAME	DESCRIPTION
BuildTarget	platform	The build target platform

Returns

TYPE	DESCRIPTION
string	The friendly string name for the build target

GetRuntimePlatform(BuildTarget)

Get the runtime platform from the specified build target.

Declaration

```
public static RuntimePlatform GetRuntimePlatform(BuildTarget platform)
```

Parameters

TYPE	NAME	DESCRIPTION
BuildTarget	platform	The build target platform

Returns

TYPE	DESCRIPTION
RuntimePlatform	The runtime platform for the build target

Exceptions

TYPE	CONDITION
NotSupportedException	The build target is not supported

IsDLCBuildTargetAvailable(BuildTarget)

Check if build support for the target platform is installed.

Declaration

```
public static bool IsDLCBuildTargetAvailable(BuildTarget buildTarget)
```

Parameters

TYPE	NAME	DESCRIPTION
BuildTarget	buildTarget	The build target to check

Returns

TYPE	DESCRIPTION
bool	True if build support is available or false if not

Class DLCPlatformProfileAndroid

Inheritance

object
DLCPlatformProfile
DLCPlatformProfileAndroid

Inherited Members

DLCPlatformProfile.friendlyPlatformNames
DLCPlatformProfile.PlatformFriendlyName
DLCPlatformProfile.PlatformName
DLCPlatformProfile.Platform
DLCPlatformProfile.RuntimePlatform
DLCPlatformProfile.Enabled
DLCPlatformProfile.DlcUniqueKey
DLCPlatformProfile.DLCExtension
DLCPlatformProfile.UseCompression
DLCPlatformProfile.StrictBuild
DLCPlatformProfile.PreloadSharedAssets
DLCPlatformProfile.PreloadSceneAssets
DLCPlatformProfile.ShipWithGame
DLCPlatformProfile.ShipWithGameDirectory
DLCPlatformProfile.ShipWithGamePath
DLCPlatformProfile.PlatformDefines
DLCPlatformProfile.GetFriendlyPlatformName(BuildTarget)
DLCPlatformProfile.GetRuntimePlatform(BuildTarget)
DLCPlatformProfile.IsDLCBuildTargetAvailable(BuildTarget)

Namespace: DLCToolkit.Profile

Assembly: DLCToolkit.Profile.dll

Syntax

```
public sealed class DLCPlatformProfileAndroid : DLCPlatformProfile
```

Constructors

DLCPlatformProfileAndroid(params string[])

Create a new instance.

Declaration

```
public DLCPlatformProfileAndroid(params string[] platformDefines)
```

Parameters

TYPE	NAME	DESCRIPTION
string[]	platformDefines	An array of platform defines

Properties

DLCAssetPackDirectory

The DLC custom asset pack directory.

Declaration

```
public string DLCAssetPackDirectory { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

DeliveryType

The DLC delivery type.

Declaration

```
public string DeliveryType { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

Methods

GetDLCCustomAssetPackGradlePath(string)

Get the Android gradle build file path for the DLC platform.

Declaration

```
public string GetDLCCustomAssetPackGradlePath(string dlcName)
```

Parameters

TYPE	NAME	DESCRIPTION
string	dlcName	The dlc name

Returns

TYPE	DESCRIPTION
string	The gradle file path

GetDLCCustomAssetPackPath(string)

Get the Android custom asset pack path for the DLC platform.

Declaration

```
public string GetDLCCustomAssetPackPath(string dlcName)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
string	dlcName	The dlc name

Returns

TYPE	DESCRIPTION
string	The asset pack path

GetDLCustomAssetGradleContents(string)

Get the Android gradle build file contents for the DLC platform.

Declaration

```
public string GetDLCustomAssetGradleContents(string dlcName)
```

Parameters

TYPE	NAME	DESCRIPTION
string	dlcName	The dlc name

Returns

TYPE	DESCRIPTION
string	The gradle file contents

Class DLCProfile

Represents a build profile for a specified DLC containing most preferences and options relating to the build process.

Inheritance

object
Object
ScriptableObject
DLCProfile

Inherited Members

ScriptableObject.SetDirty()
ScriptableObject.CreateInstance(string)
ScriptableObject.CreateInstance(Type)
ScriptableObject.CreateInstance<T>()
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.FindObjectsByType(Type, FindObjectsSortMode)
Object.FindObjectsByType(Type, FindObjectsInactive, FindObjectsSortMode)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsByType<T>(FindObjectsSortMode)
Object.FindObjectsOfType<T>(bool)
Object.FindObjectsByType<T>(FindObjectsInactive, FindObjectsSortMode)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindFirstObjectByType<T>()
Object.FindAnyObjectByType<T>()
Object.FindFirstObjectByType<T>(FindObjectsInactive)
Object.FindAnyObjectByType<T>(FindObjectsInactive)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)

Object.FindFirstObjectByType(Type)
Object.FindAnyObjectByType(Type)
Object.FindObjectOfType(Type, bool)
Object.FindFirstObjectByType(Type, FindObjectsInactive)
Object.FindAnyObjectByType(Type, FindObjectsInactive)
Object.ToString()
Object.name
Object.hideFlags

Namespace: DLCToolkit.Profile

Assembly: DLCToolkit.Profile.dll

Syntax

```
[CreateAssetMenu]
public sealed class DLCProfile : ScriptableObject
```

Properties

CustomIcons

The custom icons list for the DLC content.

Declaration

```
public IList<DLCCustomIcon> CustomIcons { get; }
```

Property Value

TYPE	DESCRIPTION
IList<DLCCustomIcon>	

DLCBuildPath

The build path for this DLC profile. The build path is the optional output directory where built DLC wil be stored. Default is 'DLC Content/'.

Declaration

```
public string DLCBuildPath { get; }
```

Property Value

TYPE	DESCRIPTION
string	

DLCContentPath

Get the content path for this DLC profile. The content path is the project asset location where DLC assets should be created.

Declaration

```
public string DLCContentPath { get; }
```

Property Value

TYPE	DESCRIPTION
string	

DLCGuid

Get the guid for this DLC profile.

Declaration

```
public string DLCGuid { get; }
```

Property Value

TYPE	DESCRIPTION
string	

DLCName

The name for this DLC content.

Declaration

```
public string DLCName { get; }
```

Property Value

TYPE	DESCRIPTION
string	

DLCProfileName

Get the name of this DLC profile.

Declaration

```
public string DLCProfileName { get; }
```

Property Value

TYPE	DESCRIPTION
string	

DLCVersion

The version for this DLC content.

Declaration

```
public Version DLCVersion { get; }
```

Property Value

TYPE	DESCRIPTION
Version	

DLCVersionString

The version string for this DLC content in the format X.X.X

Declaration

```
public string DLCVersionString { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Description

A short description for this DLC content.

Declaration

```
public string Description { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Developer

The developer or studio name that created this DLC content.

Declaration

```
public string Developer { get; }
```

Property Value

TYPE	DESCRIPTION
string	

EnabledForBuild

Is this DLC enabled for build.

Declaration

```
public bool EnabledForBuild { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

ExtraLargeIcon

The extra large icon for the DLC content.

Declaration

```
public Texture2D ExtraLargeIcon { get; set; }
```

Property Value

TYPE	DESCRIPTION
Texture2D	

LargeIcon

The large icon for the DLC content.

Declaration

```
public Texture2D LargeIcon { get; set; }
```

Property Value

TYPE	DESCRIPTION
Texture2D	

MediumIcon

The medium icon for the DLC content.

Declaration

```
public Texture2D MediumIcon { get; set; }
```

Property Value

TYPE	DESCRIPTION
Texture2D	

Platforms

The available platform profiles for this DLC content.

Declaration

```
public DLCPlatformProfile[] Platforms { get; }
```

Property Value

TYPE	DESCRIPTION
DLCPlatformProfile[]	

Publisher

The publisher name that distributed this DLC content.

Declaration

```
public string Publisher { get; }
```

Property Value

TYPE	DESCRIPTION
string	

SignDLC

Should the DLC content be signed to this game project. Signing means that the DLC will be encoded with data making it loadable only by this game project.

Declaration

```
public bool SignDLC { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

SignDLCVersion

Should the DLC content be signed with version information to this game project. Version signing means that the DLC will signed to and loadable only by a specific version of this game project.

Declaration

```
public bool SignDLCVersion { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

SmallIcon

The small icon for the DLC content.

Declaration

```
public Texture2D SmallIcon { get; set; }
```

Property Value

TYPE	DESCRIPTION
Texture2D	

Methods

GetPlatform(BuildTarget)

Declaration

```
public DLCPlatformProfile GetPlatform(BuildTarget buildTarget)
```

Parameters

TYPE	NAME	DESCRIPTION
BuildTarget	buildTarget	

Returns

TYPE	DESCRIPTION
DLCPlatformProfile	

IsAnyPlatformEnabled(BuildTarget[])

Check if this DLC profile has any of the provided build targets enabled for build.

Declaration

```
public bool IsAnyPlatformEnabled(BuildTarget[] buildTargets)
```

Parameters

TYPE	NAME	DESCRIPTION
BuildTarget[]	buildTargets	An array of build targets to check

Returns

TYPE	DESCRIPTION
bool	True if any of the provided build targets are enabled for build or false if not

IsPlatformEnabled(BuildTarget)

Check if this DLC profile has the specified build target enabled for build.

Declaration

```
public bool IsPlatformEnabled(BuildTarget buildTarget)
```

Parameters

TYPE	NAME	DESCRIPTION
BuildTarget	buildTarget	The build target to check

Returns

TYPE	DESCRIPTION
bool	True if the build target is enabled for build or false if not

Enum ShipWithGameDirectory

Namespace: DLCToolkit.Profile

Assembly: DLCToolkit.Profile.dll

Syntax

```
public enum ShipWithGameDirectory
```

Fields

NAME	DESCRIPTION
BuildDirectory	
StreamingAssets	