

Derived Data Cache (DDC)

Learn about caching data to save your team's time and disk space.



Many **Unreal Engine (UE)** Assets require additional **derived data** before you can use them.

For example, suppose you make a [material](#) with a shader before UE can render the material. In that case, it must compile the shader for the platform Unreal Editor is running on.

Derived data is large, and UE may need to regenerate it throughout development. Therefore, UE keeps it in a **Derived Data Cache (DDC)** instead of checking it into source control.

This document covers:

- An overview of how DDCs are structured.

- The types of DDCs that UE supports.
- Instructions on how to configure, use, and distribute DDCs.
- FAQs for:
 - General DDC use.
 - Switching from a Filesystem to Unreal Zen Storage (ZenServer).

How the DDC is Structured

Depending on how you configure your project and system, you can have several DDC caches in a hierarchy that range from fast to slow. When assessing derived data, your system will do the following to determine how quickly it can access it:

1. When an asset needs derived data, your system checks the fastest cache first, then the next fastest, and so on, until it finds the data.
2. When the data is found, your system copies it into the fastest local cache so it can access it more quickly next time.
3. If your system does not find the data, it generates new derived data for the asset, then asynchronously copies it into the caches so it is available for you and your team in the future.

Content stored in the DDC is disposable, and UE can regenerate it anytime using the data stored in `.uasset` files. Storing these derived formats externally makes it possible to easily add or change the formats UE uses without modifying the source Asset file.

Types of DDC

UE has multiple storage types available for DDC, and you can use each for different DDC locations. The following list covers the persistent storage types:

- **Filesystem**
 - Read/write capable.
 - Suitable for local disk and LAN use.

- Stores data as many files within a root folder, either on your local machine or a file share on the private local network. Unreal Editor directly handles the file read, write, and delete operations, including cleanup or garbage collection of unused data.

- **PAK**

- Read-only.
- Suitable for local disk and LAN use.
- Stores data as a single archive file on disk, usually named with a `.ddp` extension. UE expects the archive to be read-only. A periodic process separate from Unreal Editor writes the contents of the archive.

- **S3**

- Read-only.
- Suitable for cloud use.
- Stores data as a single archive file fetched from a cloud-hosted **simple storage service (S3)** bucket. A periodic process separate from Unreal Editor writes the contents of the archive.



Using the S3 DDC storage type is no longer recommended, and users should migrate to the Unreal Cloud DDC Client instead.

- **Unreal Zen Store Client**

- Read/write capable.
- Suitable for local disk and LAN use.
- Stores data in an [Unreal Zen Store](#) server hosted on the local machine or the private local network, which handles disk persistence. This is an unauthenticated service and should be limited to local connections or a trusted network/VPN. Unreal Editor makes asynchronous requests to the server over HTTP. The server manages the cleanup and garbage collection of any unused data.

- **Unreal Cloud DDC Client**

- Read/write capable.
- Suitable for cloud use.

- Data is stored in [Unreal Cloud DDC server](#), which handles how it persists in cloud storage (for example, AWS, Azure, or others) and how it replicates between different regions. You can deploy this authenticated service on the public internet if combined with an authentication service and appropriate security policies. The server handles cleanup and garbage collection of unused data instead of Unreal Editor.

UE 5.3 and older default to using a **Filesystem DDC as a Local DDC** to represent derived data for your project. For example, you would find your Local DDC in your UE install directory in the `DerivedDataCache` folder.

UE 5.4 defaults to using an **Unreal Zen Store DDC as a Local DDC**, with a default storage location of `AppSettingsDir/Zen/Data`. The legacy Filesystem type Local DDC is set to a delete-only mode and configured to delete data after an 8-day cleanup period. However, you can also set up your project to use a **DDC Pak** or **Shared DDC**.

DDC Pak

If you download UE from the Epic Games Store, it will come with a DDC Pak (`.ddp`). The DDC Pak contains derived data for all engine content, so you can start working without compiling shaders and other engine assets that use derived data. Some samples also ship with a DDC Pak for the same reason.

You can find the DDC Pak for the:

- Engine at `<YourEngineDir>/DerivedDataCache/Compressed.ddp`
- Project at `<YourProjectDir>/DerivedDataCache/Compressed.ddp`

Shared DDC

A Shared DDC is either a network or mapped drive. An example of a shared drive would be: `\epicgames.net\root\DDC-Global-GameTitle`

We strongly recommend that teams in a common location, such as a centralized office or company HQ, set up a Shared DDC. The most common arrangement for a Shared DDC is to use Filesystem-type DDC storage on a network drive that all team members and build machines can read/write to. This then writes off

the cost of creating any required DDC data across the entire team. For example, when an artist edits a shader, the DDC data is written straight to the DDC share.

Experimental: Local Zen Storage Server

As an alternative to using a Filesystem-type DDC on a shared network drive, you can host an Unreal Zen Storage Server instance on a machine on your local network and use it as a Shared DDC. If you host it on a Windows machine, you can install the server as a Windows service using the following command line:

Command Line

```
zenserver.exe -install
```

 Copy full snippet

The `zenserver` process can also accept a configuration file using the `-config [path_to_config_file]` argument. The configuration file controls behavior such as:

- The default data directory.
- Garbage collection parameters.
- Network configuration.

You can find an example of this configuration in the EpicGames GitHub repository [here](#). If you need access to the repository, follow the instructions on the [UE on GitHub](#) page.

Cloud DDC

If all of your team members are not located in one common location, there is significant value in setting up a Cloud DDC for your team to share DDC data beyond a local network or VPN.

With a Cloud DDC, you can add individual pieces of DDC data to the archive as they are written. This is in contrast to PAK or S3 DDCs, which create and publish a fixed, read-only archive that contains commonly needed data at specific intervals.

You can set up a Cloud DDC service using the instructions in the EpicGames GitHub repository [here](#). If you need access to the repository, follow the instructions on the [UE on GitHub](#) page.

How to Use a Shared DDC

Your Shared DDC should be accessible to all users in a particular location. This way, only one person at any given time needs to build the derived data format(s) for an asset, after which the derived data is automatically available to all other users.

Users' systems occasionally stall when they process assets, but the results are stored in the DDC and shared with other users. Even on a small team, sharing asset processing work in this way eliminates most processing time for the team as a whole.



We advise against copying an entire DDC across the internet, creating backups of your DDC, or restoring a DDC from a remote backup, as it takes longer to transfer the amount of data stored in the DDC than it does to generate it from scratch locally. If you have a large project and want to distribute pre-built DDC data, you should generate a DDC Pak.

How to Set Up a Shared DDC

Your UE installation's `BaseEngine.ini` is already set up so Unreal Editor can use a Filesystem-type Shared DDC.

You can enable the Shared DDC in one of three ways, which are listed below in order from most to least recommended:

1. Add an override in your project's `DefaultEngine.ini` that sets the path to a valid location for your team. For example:

DefaultEngine.ini

```
[DerivedDataBackendGraph] Shared=(Type=FileSystem, ReadOnly=false, Clean=false, Flush=false, DeleteUnused=true, UnusedFileAge=10, FoldersTo
```

 Copy full snippet

1. Create an environment variable mapped to the file path of the folder you want to use:
 - **Windows:** `UE-SharedDataCachePath=[path to folder]`
 - **Mac/Linux:** `UE_SharedDataCahcePath=[path to folder]`
2. Open Editor Preferences, navigate to **General > Global > Derived Data Cache**, then set the **Global Shared DDC Path**.

How to Disable a Shared DDC

Developers working from remote locations may experience poor performance because of delays between DDC data access and generation.

To temporarily disable a shared DCC, use one of the methods below:

- For Filesystem or Unreal Zen Storage DDCs, use a `DeactivateAt` configuration parameter in your `DefaultEngine.ini` to set a threshold (in milliseconds) at which the DDC storage layer will disable itself due to poor performance.
- Pass `-ddc=noshared` on the command line.
- Set the environment variable to a local drive hard drive:
 - **Windows:** `UE-SharedDataCachePath=None`
 - **Mac/Linux:** `UE_SharedDataCachePath=None`

How to Use a Cloud DDC

If your team works from different locations, a Cloud DDC ensures you can share your DDC data efficiently.

To use a Cloud DDC, you need to set up its services on a cloud service provider. You can learn more about this on the [Cloud DDC Setup Guide](#). Once this service is in place, you can configure the DDC client as described below.

How to Set Up a Cloud DDC

Your UE installation's `BaseEngine.ini` is already set up so Unreal Editor can use a Cloud DDC.

You can enable the Cloud DDC by adding an override in your project's `DefaultEngine.ini`:

DefaultEngine.ini

```
1 [StorageServers]
2 Cloud=(Host="https://cloud-ddc.example.com", Namespace="myproject.sddc", OAuthProviderIdentifier=MyOrg-MyService,
   OAuthProvider="https://domain.oauthprovider.com/oauthuripath", OAuthClientId="ClientIdString")
```

 Copy full snippet

This override includes:

- A path to a valid host.
- A namespace.
- Authentication information for your team.

For more information on how to set up interactive login for your identity provider, see the [Oidc Token page](#).

Non-Interactive Login to Cloud DDC for Automated Jobs

As part of the authentication flow, Cloud DDCs typically involve a periodic interactive login step.

For automated jobs on build farms or elsewhere, you can bypass the interactive login using a token passed to the Unreal Editor process. Your authentication service (either static or cycling) can generate this token.

The token must have a long enough validity window to encompass your longest-running automation jobs. The token should be a secret, redacted from build system logs, and only held in memory – not on disk – on the machine executing the automated job.

Once you have such a token, you can pass it to Unreal Editor via a configured environment variable named `UE-CloudDataCacheAccessToken`.

How to Build Derived Data for a Cloud DDC

The user who imports an Asset is the one who builds the derived data since they will most likely be using and testing that Asset in UE.


However, there may be times when a new Asset needs to be processed. This happens automatically on an as-needed basis and should not result in much of an impact when running on fast hardware, though there may be occasional stalls.

You can fill your DDCs at any time by running the following command in your UE installation folder:

Command Line

```
Engine\Binaries\Win64\UnrealEditor.exe ProjectName -run=DerivedDataCache -fill
```

 Copy full snippet

 Epic Games does this nightly to ensure that the DDC is always primed, but it is unnecessary, as the general automatic caching feature should suffice.

How to Package and Distribute a DDC

Cooking is the preferred method for packaging games since cooked builds do not need or use a DDC. However, DDCs can be packaged for distribution if the need arises.

To package a DDC:

1. Run `UnrealEditor.exe` from your UE installation's `Engine/Binaries/Win64` directory, passing the arguments shown below: `UnrealEditor.exe ProjectName -run=DerivedDataCache -fill -DDC=CreatePak`
2. This creates a `DDC.ddp` file in the `Project\DerivedDataCache` directory.
3. UE automatically detects and uses the `.ddp` file.

How to Configure Your DDC

You can configure the DDC settings under the `[DerivedDataBackendGraph]` section in the following `.ini` files:

- Project Settings: `<YourProjectDirectory>/Config/DefaultEngine.ini`
- Engine Defaults: `<YourUEDirectory>/Config/BaseEngine.ini`

Unreal Zen Store-type and Unreal Cloud-type DDCs can use a `ServerID` parameter to reference a shared server configuration stored in the `[StorageServers]` section of the `.ini` files. This makes it possible to specify a collection of parameters (such as a hostname and namespace) once instead of using multiple DDC graph sections.

For more information on each of the DDC configuration options, see the comments under the `[DerivedDataBackendGraph]` section of [BaseEngine.ini](#).

For more information on config files in UE, see `[Configuration Files]` (`programming-and-scripting\unreal-architecture\configuration-files`).

Frequently Asked Questions (FAQ)

General Questions

Q: Can I have multiple DDC settings?

Yes, you can create a new `[YourDDCSecrets]` entry in `DefaultEngine.ini` and then run Unreal Editor with `-ddc=YourDDCSecrets`.

At Epic, we do this for three reasons:

1. Teams in some offices want to use their own DDC instead of the global one.
2. When we create DDC Paks, we use one of several options that control what goes into the `.pak` file, such as `[CreateInstalledEnginePak]`.
3. People working from home don't always benefit from a Shared DDC due to slower internet connections or VPN issues.

`BaseEngine.ini` contains several DDC entries by default, including `NoShared`, used for the work-from-home use-case above (`-ddc=NoShared`).

Q: Can I move the Local DDC elsewhere if I am short on disk space?

Yes, you can adjust this in your project's Editor Preferences. You can also set an environment variable of `UE-LocalDataCachePath` to a path of your choice. For example: `UE-LocalDataCachePath=D:\DDC`.

1. In the command prompt, type `setx UE-LocalDataCachePath D:\DDC`.
2. Restart Unreal Editor and any application you launch it from, such as Epic Launcher, UGS, or Visual Studio.

You can use this to avoid duplicating data when you have limited space on your primary drive or are working on multiple branches.

Q: Can I turn off the Shared DDC if I have a slow network connection?

Yes, there are three options:

- (preferred, automatic) Open your `DefaultEngine.ini` and add the parameter `DeactivateAt=40` to your Shared DDC configuration. The Shared DDC deactivates itself if the latency to the Shared DDC is greater than 40 milliseconds. Substitute the value of 40 for any other value that is more appropriate for your use case. This is available for Filesystem-type or Unreal Zen Store-type Shared DDCs.
- Setting the environment variable for a DDC to `None` disables the DDC. In this case, you would set `UE-SharedDataCachePath=None`.
- Launch Unreal Editor with `-ddc=noshared`.

Q: Can I change the Shared DDC path?

Yes, you can change it by setting the `UE-SharedDataCachePath` mentioned above.

However, you should never set `UE-SharedDataCachePath` to a local path. Doing this means you would have both Local and Shared caches on your machine—twice the disk space for zero gain!



Experimental: If using an Unreal Zen Storage type DDC, you can change the host (not path) with the `UE-ZenSharedDataCacheHost=hostname` environment variable (or `UE_ZenSharedDataCacheHost=hostname` environment variable on Mac).

Q: How can I diagnose DDC issues?

If you think your Unreal Editor is not correctly reading DDC data, search your log file for `LogDerivedDataCache`.

Log File

```
1 LogDerivedDataCache: Display: Max Cache Size: 512 MB
2 LogDerivedDataCache: FDerivedDataBackendGraph: Pak pak cache file ../../../../EngineTest/DerivedDataCache/DDC.ddp **not** found, will
**not** **use** a pak cache.
```

```
3 LogDerivedDataCache: Unable to find inner node Pak **for** hierarchical cache Hierarchy.
4 LogDerivedDataCache: FDerivedDataBackendGraph: EnginePak pak cache file ../../../../Engine/DerivedDataCache/DDC.ddp **not** found, will
  **not** **use** a pak cache.
5 LogDerivedDataCache: Unable to find inner node EnginePak **for** hierarchical cache Hierarchy.
6 LogDerivedDataCache: Found environment variable UE-LocalDataCachePath=D:\DDC
7 LogDerivedDataCache: Using Local data cache path D:\DDC: Writable
8 LogDerivedDataCache: Using Shared data cache path \\epicgames.net\root\DDC-Global-UE: Writable
```

 Copy full snippet

In this example, you can see:

- There's no project or Engine Pak cache. This build was compiled from Perforce, so this is expected.
- The local data cache path is writable and mapped to `D:\DDC`.
- It is using Epic's writable shared cache.



To turn on verbose logging, run with `-logcmds="LogDerivedDataCache Verbose"`.

Q: How do I create a DDC Pak?

To create a project-specific DDC, run: `UnrealEditor.exe ProjectName -run=DerivedDataCache -fill -DDC=CreatePak`

This command will create a DDC for all content in the project. Alternatively, you can supply `-DDC=CreatePak` while running automation or even a user session to generate a more targeted set of content.

To create an Engine DDC, run: `UnrealEditor.exe -run=DerivedDataCache -fill -DDC=CreatePak`

Transitioning Filesystem Local DDC to Unreal Zen Storage Local DDC

Q: What is Unreal Zen Storage (ZenServer)?

Unreal Zen Storage, also known as ZenServer, is an Epic-made program that manages multiple kinds of data outside of Unreal Editor. We use it to manage Local DDC data.

It automatically opens and closes with Unreal Editor, but you can configure it to stay running even when it is closed.

Q: What happens when I transition from Local DDC to Unreal Zen Storage Local DDC?

The Local DDC is changed to use Unreal Zen Storage to manage the cached data on your computer instead of just folders of files. This is done for Windows, Mac, and Linux environments.

Q: Will this take up disk space on my computer?

Yes, we designed this change to keep your old folder of Local DDC files and use them to feed the new Unreal Zen Storage Local DDC. This may cause duplicate data, but the old Local DDC folder will delete unused data after eight days.

Q: Do I have to change DDC configuration settings in Unreal Editor?

No, Unreal Zen Storage uses the existing configuration settings, and the guidance for configuring DDC is unchanged.

Q: Where will Unreal Zen Storage store the Local DDC data?


Unreal Zen Storage respects your existing Local DDC configuration if it has a path set outside your UE workspace root directory and one of the following:

- A Global Local DDC Path set in Unreal Editor DDC settings.
- A Local DDC Path override via an environment variable.
- A Local DDC Path override via a command line argument.

If so, Unreal Zen Storage will store its data in a Zen subfolder in the specified path.

If not, Unreal Zen Storage uses a default path different from the previous Filesystem DCC used. The default path on Windows is:

```
C:\ProgramData\Epic\Zen\Data\
```

 We default to the system drive to support the common use case where a user's system drives are faster than their other drives. If you'd like to use another location, you can use one of the configuration options above to specify an alternative path.


Q: Will Unreal Zen Storage running interfere with my ability to sync using any tooling?

No, your tooling syncs the ZenServer executable, but it is not run from its synced location. Instead, the executable is copied to an install location,

```
C:\ProgramData\Epic\Zen\Install\
```

 on Windows, before execution.

Unreal Editor automatically detects when a new version of ZenServer is synced into your workspace and will copy the new executable into the install location before starting it.

 We designed ZenServer to be backward compatible, so if you have multiple workspaces with different versions of ZenServer, Unreal Editor will only use the newest version among them.