

Filtering

Filter object replication to certain network connections with Iris.



! Learn to use this **Experimental** feature, but use caution when shipping with it.

PREREQUISITE TOPICS



In order to understand and use the content on this page, make sure you are familiar with the following topics:

- [Introduction to Iris](#)

The **Iris Filtering System** determines what objects are replicated to which connections. There might be actors or objects in your game that you only want replicated to certain connections. To save time and bandwidth, the filtering system filters which connections an actor or object can replicate. The filtering system supports four different filtering types:

- [Owner](#): Object replicates to the same connections as its owner.
- [Connection](#): Object replicates to specified, allowed connections and does not replicate to specified, disallowed connections.

- [Group](#): Object replicates to the same connections as all other objects in its group.
- [Dynamic](#): Object replicates based on custom, dynamic filtering.

Owner

Actors with the `bOnlyRelevantToOwner` flag set to `true` automatically enable owner filtering. You can also enable owner filtering for standalone objects other than actors.

Set Owner Filters For Objects

To enable owner filtering for objects other than actors, follow these steps:

1. Include the necessary Iris files in order to access required Iris functionality.

```
1 #if UE_WITH_IRIS
2 #include "Net/Iris/ReplicationSystem/ReplicationSystemUtil.h"
3 #include "Iris/ReplicationSystem/ReplicationSystem.h"
4 #include "Net/Iris/ReplicationSystem/ActorReplicationBridge.h"
5 #include "Net/Iris/ReplicationSystem/Filtering/NetObjectFilter.h"
6 #endif UE_WITH_IRIS
```

 Copy full snippet

2. In your gameplay code, retrieve the replication system and replication bridge for your replicated object.

```
1 // The actor for which you want to control filtering
2 AActor* RepActorPtr;
3
4 UReplicationSystem* ReplicationSystem =
    UE::Net::FReplicationSystemUtil::GetReplicationSystem(RepActorPtr);
5 UActorReplicationBridge* ReplicationBridge =
    UE::Net::FReplicationSystemUtil::GetActorReplicationBridge(RepActorPtr);
```

 Copy full snippet

3. Retrieve the `FNetRefHandle` for your replicated object. Iris uses this identifier to locate your object within the replication system.

```
UE::Net::FNetRefHandle RepActorNetRefHandle = ReplicationBridge->Ge
```

 Copy full snippet

4. Set the owner filter handle for your replicated object.

```
ReplicationSystem->SetFilter(RepActorNetHandle, UE::Net::ToOwnerFilter)
```

 Copy full snippet



You should assign the owner filter just after an object's owner is assigned.

Clear All Filters

To clear all filters on a replicated object, first follow Step 1 and 2 from [Set Owner Filters for Objects](#), then set the filter to use the following special filter handle:

```
#if UE_WITH_IRIS
#include "Net/Iris/ReplicationSystem/ReplicationSystemUtil.h"
#include "Iris/ReplicationSystem/ReplicationSystem.h"
#include "Net/Iris/ReplicationSystem/ActorReplicationBridge.h"
#include "Net/Iris/ReplicationSystem/Filtering/NetObjectFilter.h"
#endif UE_WITH_IRIS

UReplicationSystem* ReplicationSystem = UE::Net::FReplicationSystemUtil::GetRepl:
UActorReplicationBridge* ReplicationBridge = UE::Net::FReplicationSystemUtil::Ge

UE::Net::FNetRefHandle RepActorNetRefHandle = ReplicationBridge->GetReplicatedRe:

// Set filter to special invalid handle
ReplicationSystem->SetFilter(RepActorNetRefHandle, UE::Net::InvalidNetObjectFilter)
```

 Copy full snippet

Connection

You can set the connections an object replicates to by specifying connection IDs. We advise using this option when:

- The allowed connections for the object in question are static, and...
- Only a few objects are affected if the connection IDs change for some reason.

If a large number of objects might be affected, consider using group filtering instead.

Connection filtering enables you to add either allowed or disallowed connections to an object.

Set Connection Filters

To add allowed connections:

1. Set the bits for all connections for which the object is allowed to be replicated
2. Initialize the connection with a maximum number of allowed connections before setting the connection filter.

```
#if UE_WITH_IRIS
#include "Net/Iris/ReplicationSystem/ReplicationSystemUtil.h"
#include "Iris/ReplicationSystem/ReplicationSystem.h"
#include "Net/Iris/ReplicationSystem/ActorReplicationBridge.h"
#include "Net/Iris/ReplicationSystem/Filtering/NetObjectFilter.h"
#endif UE_WITH_IRIS

UReplicationSystem* ReplicationSystem = UE::Net::FReplicationSystemUtil::GetReplicationSystem();
UActorReplicationBridge* ReplicationBridge = UE::Net::FReplicationSystem::GetReplicationSystem()->GetActorReplicationBridge();

UE::Net::FNetRefHandle RepActorNetRefHandle = ReplicationBridge->GetRepActorNetRefHandle();

int32 MaxConnections = 100;

TBitArray<> Connections;
Connections.Init(false, MaxConnections);

// PlayerControllerPtr is the player controller for the connection you want to add
uint32 ConnectionId = PlayerControllerPtr->GetNetConnection()->GetConnectionId();
Connections.Insert(true, ConnectionId);
```

```
ReplicationSystem->SetConnectionFilter(RepActorNetRefHandle, Connectio
```

 Copy full snippet

Similar logic applies to add disallowed connections. Instead of using the

`ENetFilterStatus::Allow`, use the `ENetFilterStatus::Disallow` value. In this case, the provided connections are disallowed connections for the object to replicate to.

Clear Connection Filters

Connection filtering for an object is cleared differently than owner filters. To clear connection filters for an object, set a disallowed connection filter, but specify no provided connections to disallow:

```
#if UE_WITH_IRIS
#include "Net/Iris/ReplicationSystem/ReplicationSystemUtil.h"
#include "Iris/ReplicationSystem/ReplicationSystem.h"
#include "Net/Iris/ReplicationSystem/ActorReplicationBridge.h"
#include "Net/Iris/ReplicationSystem/Filtering/NetObjectFilter.h"
#endif UE_WITH_IRIS

UReplicationSystem* ReplicationSystem = UE::Net::FReplicationSystemUtil::GetRepl:
UActorReplicationBridge* ReplicationBridge = UE::Net::FReplicationSystemUtil::Ge

UE::Net::FNetRefHandle RepActorNetRefHandle = ReplicationBridge->GetReplicatedRe:

TBitArray<> NoConnections;

ReplicationSystem->SetConnectionFilter(ObjectNetHandle, NoConnections, UE::Net::I
```

 Copy full snippet

Group

Iris includes an API for creating groups and managing the objects those groups contain. You can also use groups as a filtering mechanism. This is a flexible way of changing which

connections a set of objects can replicate to without requiring you to manually keep track of which objects belong to which groups. Example use cases include filtering based on:

- Team
- Squad
- Streaming Level

An object can belong to more than one group. You must add groups to the filtering system for it to consider them for replication. Once you add them, you can modify which connections members of the group are allowed to replicate. You can set groups as allowed or disallowed for replication to:

- A single connection
- A set of connections
- All connections

Create a Group

To create a filter group, call the `CreateGroup` function, which returns a unique group handle:

```
#if UE_WITH_IRIS
#include "Net/Iris/ReplicationSystem/ReplicationSystemUtil.h"
#include "Iris/ReplicationSystem/ReplicationSystem.h"
#endif UE_WITH_IRIS

UReplicationSystem* ReplicationSystem = UE::Net::FReplicationSystemUtil::GetRepl:
UE::Net::FNetObjectGroupHandle GroupHandle = ReplicationSystem->CreateGroup();
```

 Copy full snippet


Add Group Filter

You must add the group filter must to the filtering system before setting the filter status and adding objects:

```
#if UE_WITH_IRIS
#include "Iris/ReplicationSystem/ReplicationSystem.h"
```

```
#endif UE_WITH_IRIS
```

```
// Add a valid FNetObjectGroupHandle  
ReplicationSystem->AddGroupFilter(GroupHandle);
```

 Copy full snippet

Set Group Filtering

Once you create a group and add the group filter to the replication system, you can set the group filtering. In this example, any object belonging to the group with handle `GroupHandle` is allowed to replicate only to the connection with ID `SpecialConnectionId`:

```
#if UE_WITH_IRIS  
#include "Iris/ReplicationSystem/ReplicationSystem.h"  
#include "Net/Iris/ReplicationSystem/Filtering/NetObjectFilter.h"  
#endif UE_WITH_IRIS  
  
// With a valid FNetObjectGroupHandle and uint32 connection id  
ReplicationSystem->SetGroupFilterStatus(GroupHandle, SpecialConnectionId, UE::Ne
```

 Copy full snippet

Add Object to Group

You can now add your desired object to the group:

```
#if UE_WITH_IRIS  
#include "Net/Iris/ReplicationSystem/ActorReplicationBridge.h"  
#include "Iris/ReplicationSystem/ReplicationSystem.h"  
#endif UE_WITH_IRIS  
  
UE::Net::FNetRefHandle RepActorNetRefHandle = ReplicationBridge->GetReplicatedRe:  
  
// With a valid FNetObjectGroupHandle and FNetHandle
```

```
ReplicationSystem->AddToGroup(GroupHandle, RepActorNetRefHandle);
```

 Copy full snippet

Dynamic

Iris's filtering system can also implement custom, dynamic filtering with a `UNetObjectFilter`-derived class. Dynamic filters can restrict objects from replicating based on custom logic. Dynamic filters are applied after connection and group filters. This means that a dynamic filter cannot enable replication for an object that has already been filtered out by connection or group filters. An object can only have one dynamic filter set at a time.

Dynamic filters always run on an object, whereas connection and group filtering only run when the system is informed of changes. This is due to the fact that a dynamic filter can be implemented in any way, so there is no way for the system to know whether the dynamic filter needs to run or not. There are situations where a dynamic filter provides the best solution. If you modify groups often, such as moving actors between groups or changing which connections the group may be replicated to, a dynamic filter may provide the best solution.



You should only set a dynamic filter for an object as a last resort. This adds significant CPU overhead versus connection and group filters.

UE provides a few dynamic filters in the directory

```
..\Engine\Source\Runtime\Experimental\Iris\Core\Public\Iris\ReplicationSystem\Filters\Filtering\:
```

- `UFilterOutNetObjectFilter`: Disallow replication of any objects added to it.
- `UNetObjectConnectionFilter`: Pre-poll filter that supports per-connection filtering for dependent objects.
- `UNetObjectGridFilter`: Divide the game world into cells and only replicate objects in cells near the player's view.

To use a custom dynamic filter, you must implement the `UNetObjectFilter` interface and it must be configured with filter definitions in order to be used at runtime.

Create a Dynamic Filter

To use a custom, dynamic filter, implement the `UNetObjectFilter` interface and configure its filter definitions to use the filter at runtime. The base class your custom filter must inherit from is located at the following file path:

- `..\Engine\Source\Runtime\Experimental\Iris\Core\Public\Iris\ReplicationSystem\Filtering\NetObjectFilter.h`


A minimal, working example of a custom, dynamic filter is provided in:

- `..\Engine\Source\Runtime\Experimental\Iris\Core\Public\Iris\ReplicationSystem\Filtering\NopNetObjectFilter.h`

Dynamic Filter Configuration

Below is the syntax for engine configuration to use a custom dynamic filter:

```
[/Script/IrisCore.NetObjectFilterDefinitions]
+NetObjectFilterDefintions=(FilterName=<FILTER_NAME>, ClassName=/Script/<PROJECT_
```

 Copy full snippet

For example, a custom, dynamic filter named `MyCustomFilter` in a project named `MyProject` is configured as follows in the engine configuration hierarchy, such as your project's `DefaultEngine.ini` file:

```
[/Script/IrisCore.NetObjectFilterDefinitions]
+NetObjectFilterDefintions=(FilterName=MyCustomFilter, ClassName=/Script/MyProje
```

 Copy full snippet

Once your filter is registered with Iris, you can configure your filter as follows in an engine configuration file such as your project's `DefaultEngine.ini` file:

```
[/Script/MyProject.MyCustomFilterConfig]
MyCustomFilterVar=100
```

 Copy full snippet

Assign Object to Dynamic Filter

To assign a dynamic filter to a replicated object, use the following:

```
const UE::Net::FNetObjectFilterHandle FilterHandle = ReplicationSystem->GetFilterHandle(ObjectNetHandle);
if (FilterHandle != UE::Net::InvalidNetObjectFilterHandle)
{
    const bool bSuccess = ReplicationSystem->SetFilter(ObjectNetHandle, FilterHandle);
}
```

 Copy full snippet

Remove Dynamic Filter

Dynamic filters are removed in the same way as owner filters. To remove any filter, see the [Clear All Filters](#) section of this page.