Migrate to Iris

Learn what has changed between the existing replication systems and 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

Iris maintains backward compatibility with Unreal Engine's (UE) generic replication system as much as possible. However, you might need to make changes to your gameplay code to accommodate key differences between the two systems.

One of Iris's key design principles is to minimize the number of interactions between the replication system and gameplay code. To accomplish this, Iris reduces the number of virtual function calls. These virtual function calls from the generic system are replaced with explicit calls to API functions through the Iris replication system.

The following table lists:

- Features of the current replication system.
- Whether they have changed in Iris.
- What the corresponding Iris feature is if the system has changed.
- Links to the documentation page for more information.

Existing Replication Feature	Changed in Iris	Iris Replication Feature
Dormancy		
Priority	✓	Iris Prioritization
Property Replication		
Relevancy	✓	Iris Filtering
Remote Procedure Calls		
Subobject Replication	✓	Iris Subobject Replication

Anything in the above table that is not checked as Changed in Iris works as it does in the generic replication system.

Push Model

Iris aims to be fully push-based. It is possible to use Iris without push model replication. If push model replication is not enabled for a particular object, Iris automatically falls back on polling the object based on its `NetUpdateFrequency`. By default, Iris honors the standard push model settings.

Replicated Properties

Replicated properties work as they do in the generic replication system. For more information about how replicated properties work in UE, see the Replicate Actor Properties documentation.

Remote Procedure Calls

Remote Procedure Call (RPC) declaration and execution in Iris works as it does in the generic replication system and replication graph. For more information about how RPC and replicated property updates are executed on receiving machines, see the Replication Execution Order documentation.

Subobject Replication

Iris requires you enable the *registered subobject list*. To use the registered subobject list for your actor class, add the following to your replicated actor's constructor:

```
bReplicateUsingRegisteredSubObjectList = true;
```

Copy full snippet

If you are using Iris, replicated subobjects not derived from `AActor` or `UActorComponent` must also implement the virtual function `RegisterReplicationFragments` to register their replicated properties and functions. To implement `RegisterReplicationFragments` for your `UObject`-derived class `UMyDerivedObject`, add the following code to your `MyDerivedObject.h` and `MyDerivedObject.cpp` files, respectively:

MyDerivedObject.h

```
#if UE_WITH_IRIS
// Register replication fragments
virtual void
RegisterReplicationFragments(UE::Net::FFragmentRegistrationContext& Context,
UE::Net::EFragmentRegistrationFlags RegistrationFlags) override;
#endif // UE_WITH_IRIS
```

Copy full snippet

MyDerivedObject.cpp

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

#if UE_WITH_IRIS
void
UMyDerivedObject::RegisterReplicationFragments(UE::Net::FFragmentRegistration
Context, UE::Net::EFragmentRegistrationFlags RegistrationFlags)
{
// Build descriptors and allocate PropertyReplicaitonFragments for this objec
UE::Net::FReplicationFragmentUtil::CreateAndRegisterFragmentsForObject(this,
RegistrationFlags);
}
#endif // UE_WITH_IRIS
```

Copy full snippet

For more information about replicated subobjects, see the Replicated Subobjects documentation.

Custom Network Serializers

Iris supports all Unreal Engine primitive types that can be used as replicated properties for network serialization. If a struct uses a custom `NetSerialize` method and is missing an Iris-specific implementation, the following warning is logged:

```
 Warning: Generating descriptor for struct STRUCT_NAME that has custom serializ
```

Copy full snippet

If the data in the `NetSerialize` method can replicate using only property network serialize methods, you can silence the warning by adding an entry to your project's `DefaultEngine.ini` file:

DefaultEngine.ini

```
[/Script/IrisCore.ReplicationStateDescriptorConfig]
; Declarate structs that are vetted to work using reflection based struct serialization even though there exists a custom NetSerialize function for the struct
+SupportsStructNetSerializerList=(StructName=STRUCT_NAME)
```

Copy full snippet

Fast Array Replication

Iris supports existing fast array definitions. Iris also provides a specialized fast array serializer, `FIrisFastArraySerializer`, located in `IrisFastArraySerializer.h`.

Peer-to-Peer

Iris supports Unreal Engine listen servers. Using a listen server, a game instance can act as the host of a multiplayer game session while also supporting its own, local players.

Differences Specific to Replication Graph

The Replication Graph plugin is a network replication system built upon nodes containing lists of persistent objects to replicate. Iris does not support replication graph. Iris and Replication Graph are separate systems, and the network driver can only use one or the other. Iris does not have a concept of a node in the same sense of replication graph to control when and where actors are replicated. Instead, the new network object filters and prioritizers are intended as a replacement for replication graph's functionality. For more information, see the Filtering and Prioritization documentation.