Glossary of Iris Terms

Glossary page for Iris terminology.

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

Iris introduces several new concepts to Unreal Engine's (UE) replication system. This page is a glossary of terms that are used frequently when referring to the organization and operation of the Iris replication system.

Glossary

Actor Replication Bridge

The actor replication bridge implements the replication bridge interface. The actor replication bridge constructs replication protocols and replication instance protocols from actors and actor components. It also implements code to poll state data from properties and issue required legacy callbacks.

Changemask

A changemask is a bit array used to track dirtiness of replicated data. Iris uses changemasks in two ways:

- All replication states have inlined storage used for local, cache-friendly tracking of changed members.
- The replication system maintains combined changemasks for all replication states in replication protocols.

Data Stream

A data stream is the Iris equivalent of a data channel in the generic replication system. Data streams are responsible for serializing data. You can also use a data stream to implement specific data delivery guarantees for different types of data.

Fast Array Replication Fragment

A fast array replication fragment is the owning object for a fast array replication state. This is a specific implementation of a replication fragment for fast arrays.

Net Bit Stream Reader

A net bit stream reader facilitates efficient deserialization of bits from a memory buffer.

Net Blob

Net blob is the base class for something that can be replicated with an object. There is currently nothing outside of Iris itself that makes use of it, but you can use them to send most types of data or events, for example, debug data or RPCs.

Net Object

A net object is the internal, networking representation of a replicated object instantiated in gameplay code. A net object is associated with a unique net ref handle and an internal index.

Net Ref Handle

A net ref handler is a unique identifier for an object on both the server and clients similar in concept to a NetGUID (Network Globally Unique Identifier). Net handles are used for object references to:

- Statically Addressable Objects
- Replicated Objects

Net Ref Handle Manager

The net ref handle manager assigns internal IDs and state storage for all net objects.

Net RPC

A net blob implemented specifically to handle standard RPCs.

Net RPC Handler

The net RPC handler processes the logic involved in creating and handling received RPCs. For example, the net RPC handler determines whether an actor is still valid to call an RPC on.

Net Token

A net token is a network object for various types of data that are not necessarily directly related to networking, but you may still want to be able to communicate between server and client.

- Examples of what can be expressed as a net token:
- Strings
- Data blobs
- Packages

You can communicate token data to clients either explicitly using a net token data stream, loaded at startup, or exported during serialization. You can query the net token manager for the status of all tokens on the remote end.

Net Token Data Stream

The net token data stream serializes net token data.

Object Reference Cache

The object reference cache stores references to replicated and statically named `UObject`s. This works in a similar way as the NetGUID cache and client package map.

Property Replication Fragment

A property replication fragment is the owning object for a property replication state.

Property Replication State

A property replication state is a dynamic replication state used to interact with the current reflection system.

Replication Bridge

The replication bridge transfers data between gameplay code and networking code. The replication bridge provides the type specific interface for the game and has functionality to create replication protocols and replication instance protocols from the data. The replication bridge is also responsible for instantiating gameplay objects, such as actors and other supported types, upon request from the replication system.

Replication Data Stream

The replication data stream is responsible for serializing and deserializing replication data.

The implementation is split into two parts:

1. Replication Reader
2. Replication Writer

Replication Fragment

A replication fragment is an owning object for a replication state. To transfer data back and forth between the replication system and gameplay code, replication states must be bound to an owning object. Replication fragments serve as this owning object for replication states.

Replication Instance Protocol

A replication instance protocol is created for each object instance. It is an array of all replication fragments for an instance and is used whenever state data is exchanged between gameplay code and the replication system.

Replication Protocol

A replication protocol defines how to replicate data for an object. A replication protocol is an

Replication Reader

A replication reader is an internal system responsible for incoming data. It drives the deserialization and state application using the replication bridge.

Replication Record

A replication record is a list of in-flight data from the replication writer. A key aspect of the replication writer is to track all in-flight data. This is done so that steps can be taken to act accordingly and efficiently if data is lost or delivered. The replication record maintains linked lists for all objects with data in-flight that is still pending delivery notification.

Replication State

A replication state is a struct containing data that should be replicated or transmitted over the network.

Replication State Descriptor

A replication state descriptor describes all aspects of a replication state including:

- Memory layout for internal and external representations.
- How to serialize each member of the state.
- Required operations.

Replication State Descriptor Builder

A replication state descriptor builder builds replication state descriptors for all supported types based on reflection data.

Replication System

The replication system is the primary interface for interacting with Iris in gameplay code.

Replication System Internal

The replication system internal is the internal interface for the replication system and is only exposed to internal Iris code.

Replication Writer

A replication writer is responsible for tracking and serializing the state of all replicated objects for all connections.

This includes:

- Scheduling replication data based on:
    - Priority.

    - Scheduling priority.
    - Replication status of replicated objects.
- Dependencies
- Subobjects
- Serialization

Typed Replication State

A typed replication state is a replication state based on a struct with accessors for all members and internal members to store the ID of the owning net object along with a bit field that tracks the dirtiness for each member.