ScarNet

Multiplayer Automation & Debugger

BLUEPRINT DEVELOPER GUIDE

Make any actor multiplayer-ready in one click, replicate variables and fire network events without touching an RPC, and find out *why* your netcode is broken with the in-game ScarNet Debugger.

Unreal Engine 5.2 • Blueprint-first, C++ compatible • Works with Dedicated / Listen / LAN / Steam / EOS
Black Scar Studio • Version 1.0

1 What ScarNet does

Multiplayer in Unreal is 10% gameplay and 90% “why isn’t this replicating.” ScarNet does two things about that:

- **Automates the boring, error-prone setup.** Add one component, pick a profile, and your actor is replicated with sane update rates, priority, dormancy, relevancy and ownership. Replicate variables and fire network events straight from Blueprint.
- **Tells you what is broken and how to fix it.** Press F9 in-game for a live debugger that inspects every networked actor and, most importantly, an **Issues** tab that names the problem in plain English with a suggested fix.

It does not replace Unreal networking. It sits on top of the engine’s replication layer, so it works with any Online Subsystem and any server type with no backend-specific code.

TIP The golden rule of this plugin

When something doesn’t sync, don’t guess. Press **F9**, read the top red line on the Issues tab, and do what it says. That loop is the whole point of ScarNet.

2 Install & enable

- Copy the **ScarNet** folder into your project’s `Plugins/` directory.
- Open the project. If prompted to rebuild modules, click **Yes**.
- Confirm it is on under **Edit** → **Plugins** → **Networking** → **ScarNet** (it is enabled by default).
- Tune anything you like under **Project Settings** → **Plugins** → **ScarNet**.

TIP C++ not required

Everything in this guide is done with Blueprint nodes and the Details panel. The plugin is fully C++ compatible too, but you never have to open a .cpp file.

3 Quick start in 4 steps

#	Do this	Result
1	Open your actor Blueprint. **Add Component** → **ScarNet Ready**.	The actor can now be made multiplayer-ready.
2	In the component Details, pick a **Replication Profile** (Character, Weapon, Projectile, Pickup, …).	On Begin Play the actor is replicated with tuned settings for that kind of actor.

#	Do this	Result
3	Use the ScarNet nodes: **Register / Set Replicated Variable**, **Broadcast / Send To Server / Send To Owner**, **MP Spawn / Interact**.	Variables sync and events fire on the right machines automatically.
4	Play with 2+ players and press **F9**.	The ScarNet Debugger opens and shows any problems with fixes.

WATCH OUT Call setup on the server, at runtime

Add the component / apply profiles / register variables from **Begin Play** (or later) on the **server**, not in a Construction Script and not on a client. Components added on a client don’t replicate. ScarNet warns you in the log if you get this wrong.

4 Replication profiles

A profile is a curated bundle of networking settings for a *kind* of actor. Pick the closest one and ScarNet applies proven defaults; choose **Custom** to set them by hand. All values are editable per project in **Project Settings** → **ScarNet**.

Profile	Use it for	What it tunes
`Character`	Player pawns	High update rate, replicated movement, always awake
`Projectile`	Bullets, rockets, grenades	High rate + replicated movement while alive
`Weapon`	Equipped weapons	Medium rate, relevant through the owner
`Interactable`	Doors, buttons, levers, switches	Dormant until used, then flushed
`Pickup`	World pickups	Low rate, dormant until grabbed
`Inventory`	Private per-player state	Only relevant to the owning player
`AI`	AI pawns	Medium rate, replicated movement
`Custom`	Anything else	You set every value on the component

TIP Movement replication is on by default

The ScarNet Ready component forces **Replicate Movement** on (and fixes a Static root that would silently block it). Moving actors just work. Turn it off per component only for truly static objects.

5 The Blueprint nodes

All nodes handle authority checks and server routing for you. They work on any machine and any net mode - call them where it reads naturally and ScarNet does the right thing.

Setup

Node	What it does
`Make Multiplayer Ready`	Adds a ScarNet Ready component (if missing) and applies a profile. Great for actors you spawn at runtime.

Replicated variables

Node	What it does
`Register Replicated Variable`	Creates a synced variable with a default value. Replicates to everyone, including late joiners.

Node	What it does
`Set Replicated Variable`	Writes it. Automatically routed to the server when called on a client. Fires the change event.
`Get Replicated Variable`	Reads the local copy anywhere.
`Unregister Replicated Variable`	Removes it everywhere (server only).

To react to a change, select the **ScarNet Ready** component and bind **On Replicated Variable Changed**. It fires on the server and every client. Use the **As Bool / As Int / As Float / As Vector…** nodes to unpack the value.

Network events

Node	Runs on
`Send To Server`	The server only (client asks the server to do something).
`Broadcast Event`	The server and every client (explosions, round start, global FX).
`Send To Owner`	The client that owns the target actor (ammo, private messages).
`Network Event`	Any of the above - you choose the scope on a dropdown, incl. Run Locally.

Receive events by binding **On Network Event** on the target's ScarNet Ready component, then branch on the event name.

Smart actors & interaction

Node	What it does
`MP Spawn Actor`	Spawns on the server from anywhere and replicates. On the server it returns the actor; on a client it returns None and the actor arrives via replication a moment later.
`MP Destroy Actor`	Destroys on the server from anywhere (client requests obey the security settings).
`MP Interact`	Interacts with an actor that has a **ScarNet Interactable** component. Works from any machine and is validated (distance-checked) on the server.

TIP MP Spawn returns None on clients - that's correct

The actor is created on the **server** and replicated down; it doesn't exist on the calling client yet. Don't try to use the return value on a client - react to the actor in its own Begin Play instead.

6 Auto variables & the "my RepNotify never fires" trap

ScarNet only sends a variable when its value actually **changes** (this saves bandwidth). That also means: if you set a variable to the value it already has, nothing replicates and the change event does **not** fire. This is the single most common multiplayer head-scratcher, so ScarNet makes it visible.

- The **Variables** tab shows a **No-Change Sets** counter climbing.
- The **Issues** tab warns: *"Variable X was set N times to the value it already had, so it never replicated and its change event never fired."*
- If you genuinely need the callback to fire anyway, tick **Force Notify** on Set Replicated Variable.

WATCH OUT If a client callback "never runs"

99% of the time the value simply isn't changing on the server. Check the No-Change Sets column first before suspecting replication itself.

7 The ScarNet Debugger (F9)

Press **F9** in any play session (or run the `ScarNet.Debugger` console command). It opens straight to a status banner and the Issues tab - the answer, not a wall of data. The panel is draggable (grab the header) and resizable (orange corner).

Read the banner first

Banner colour	Means	Do
AMBER	You are in **Standalone** - no networking is running.	Set Number of Players to 2+ (or Net Mode: Play As Client) in the Play dropdown.
RED	Problems were found.	Open the Issues tab (it's already in front) and read the top line.
GREEN	No replication problems detected.	Carry on - move and interact; issues appear the moment they happen.

The five tabs

Tab	Shows
Issues	Auto-detected problems, most-severe first, each with a suggested fix. Start here.
Actors	Per actor: Replicates, Movement, Activity (really moving?), Last Sent, owner, authority, dormancy, update rate, relevancy.
Variables	Every registered variable: value, updates, and No-Change Sets.
Events	Server / Client / Multicast RPC history with Sent / Received / Failed / Dropped.
Health	Ping, packet loss, bandwidth, connections, active vs dormant actors, saturation.

TIP Debug from any client

Most problems are only visible on the server. When you open the debugger on a **client**, ScarNet streams the server's findings down to you, so the Issues, Events and Variables tabs match the server. Text is clipped to fit - **hover any cell** to read the full line.

Bottom bar: **Ownership Overlay** draws colour-coded boxes in the world (green = server, blue = client, yellow = shared, red = problem). **Save Report** writes a full diagnostics text file to `Saved/ScarNet/` (the on-screen message shows the exact path). **Clear Events** wipes the RPC history.

8 Troubleshooting: symptom → fix

The Issues tab detects all of these automatically. This table is the same knowledge for offline reference.

Symptom	Likely cause	Fix
Actor says Replicates: Yes but is frozen on clients	Movement replication is off (Movement: No).	Enable Replicate Movement (ScarNet forces it on by default).
Nothing replicates at all; only 1 window	You're in Standalone / 1 player.	Play with 2+ players or Net Mode: Play As Client.
Client RepNotify / change event never fires	The value was set to what it already was.	Change the value, or tick Force Notify. Check No-Change Sets.
Spawned actor only appears in one window	It was spawned on a client (client authority).	Use MP Spawn Actor so it spawns on the server.
Send To Owner does nothing / logs a failure	The actor has no owning connection.	Set the actor's Owner to a player controller/pawn.
Actor exists but its state never updates	Runtime-spawned with dormancy Initial, or moving while Dormant All.	Use an Awake profile, or Flush Net Dormancy on change.
Interactable state doesn't sync to others	Only Relevant To Owner is on.	Turn it off (the Interactable profile already sets relevancy).
Values not set on a client take effect	Called Set on a client with client writes disabled.	Server-authoritative by design; enable client writes in Settings if intended.

9 Testing multiplayer on one PC

- **Quick (PIE):** in the Play dropdown set **Number of Players = 2** and **Net Mode = Play As Client**. You get two windows with a hidden dedicated server underneath.
- **Realistic:** open **Tools** → **ScarNet** → **ScarNet Debugger** and use **Dedicated Server Testing** - one click launches a dedicated server (or listen host) plus N client processes of your current map, each with its own log in `Saved/Logs`.

TIP Put movement behind Switch Has Authority

When you test replication, only the **server** should drive the change (move the actor, set the variable). If both machines do it locally, both windows look right and you've proven nothing.

10 Settings worth knowing

Project Settings → **Plugins** → **ScarNet**:

- **Debugger:** hotkey (F9 by default), refresh interval, overlay interval, max rows.
- **Security:** allow/deny client spawn, destroy and variable-write requests - all validated server-side.
- **Profiles:** every value each built-in profile applies, editable per project.
- **Server Testing:** default client count, dedicated vs listen, extra launch parameters.

Quick FAQ

Question	Answer
Does it work with Steam / EOS?	Yes - any Online Subsystem. ScarNet has no backend-specific code.
Does it run in packaged / shipping builds?	Yes. The debugger uses runtime Slate; F9 works in packaged games (not dedicated servers - use ScarNet.Report there).
Is it heavy?	No. Nothing ticks while idle; the debugger costs nothing when closed and streams to clients only while open.
Can I still write my own replication?	Absolutely. ScarNet sits on top of Unreal's system - mix and match freely.

ScarNet - Multiplayer Automation & Debugger • Black Scar Studio • Unreal Engine 5.2. Support: discord.gg/a3QdMX9JZj