Setup Guide

```
# GONet v1.5 Setup Guide

To get setup with GONet v1.5, follow the steps below. This is only
required once per project.

## Prerequisites

- Unity 2022.3.62f3 LTS or later (minimum required version)
- Basic understanding of Unity GameObject architecture
- (Optional) Watch a complete sample project tutorial video:
https://www.youtube.com/watch?v=fs1flIi35JM

## Instructions (Pre-Import)

### 1. Configure Unity Project Settings

Before importing GONet, configure these critical Unity project settings:

**Enable Unsafe Code** (Required)
- Edit → Project Settings → Player → Allow 'unsafe' Code ✓
- Required for GONet's high-performance bit manipulation and serialization

**Set API Compatibility Level** (Required)
- Edit → Project Settings → Player → Api Compatibility Level → .NET
Framework or .NET Standard 2.1 (preferred!)
- Ensures compatibility with GONet's networking libraries

**Configure Scripting Define Symbols** (Optional, Recommended)
- Edit → Project Settings → Player → Scripting Define Symbols
- Add: `LOG_DEBUG;LOG_INFO;LOG_WARNING;LOG_ERROR;LOG_FATAL`
- Enables GONet's comprehensive logging system for debugging
```

```
**Enable Incremental GC** (Optional, Recommended)
- Edit → Project Settings → Player → Use incremental GC ✓
- Helps manage memory pressure from networking events (reduces frame
hitches)

### 2. Import GONet Unity Package

Import the GONet Unity package into your project:
- Assets → Import Package → Custom Package → Select GONet .unitypackage
- **NOTE:** If you're reading this document, you've likely already
imported GONet. If you see compilation errors, ensure the above steps are
completed first.

## Instructions (Post-Import)

### 3. Compile and Verify

- Unity should auto-compile after import
- Check Console for any errors
- If errors occur, verify pre-import steps were completed correctly

### 4. Add GONet to Your Scene

**Option A: Use the provided sample scene** (Recommended for first-time
users)
- Open `Assets/GONet/Sample/GONetSampleScene.unity`
- This scene is pre-configured with all required components

**Option B: Add GONet to your existing scene**
- Drag `Assets/GONet/Resources/GONet/GONet_GlobalContext` prefab into your
startup scene
- This prefab contains the GONetGlobal singleton and all required
components

### 5. First Run - Test Basic Functionality

**Start the Scene**
- Click Run/Play in Unity Editor to play the scene
```

```markdown
- Code generation will occur automatically (first run may take 5-10 seconds)
- Scene should start without errors/exceptions in Console

**Verify Auto-Detection** (New in v1.5)
- GONet v1.5 includes automatic server/client role detection
- First instance automatically starts as SERVER (port 40000 free)
- Additional instances automatically connect as CLIENTS (port 40000 occupied)
- No manual key combinations needed!

**Manual Server Start** (Optional - only if auto-detection disabled)
- With scene running and Game window focused
- Press: Left CTRL/CMD + Left ALT + S
- GONetServer(Clone) appears in Hierarchy
- Server is now listening for client connections

**Stop the Scene**
- Click Run/Play in Unity Editor again to stop

### 6. Test in Builds

**Create a Build**
- File → Build Settings → Build
- Example output: `gonet_sample.exe` (Windows), `gonet_sample.app` (Mac)

**Windows Quick Start (Batch Files)**
- Open: `Assets/StreamingAssets/GONet/`
- Copy: `Start_CLIENT.bat` and `Start_SERVER.bat`
- Paste into build folder (where gonet_sample.exe exists)
- Edit both files: Change `GONetSandbox.exe` to `gonet_sample.exe` (or your build name)
- Save files
- Run `Start_SERVER.bat` first (server must start before clients)
- Run `Start_CLIENT.bat` (connects to localhost server)

**Manual Build Start (All Platforms)**
- Run first instance
- Focus window, press: Left CTRL/CMD + Left ALT + S (server)
- Wait 2-3 seconds for server initialization
- Run second instance
- Focus window, press: Left CTRL/CMD + Left ALT + C (client)
- Client connects to server automatically

**Auto-Detection in Builds** (New in v1.5)
- Auto-detection works in builds too!
- First instance (port free) → Starts as SERVER
- Additional instances (port occupied) → Start as CLIENTS
- Command line args (`-server` / `-client`) always override auto-detection

## Understanding Auto-Detection (New in v1.5)

GONet v1.5 introduces **automatic client/server role detection** to
streamline local development:

**How It Works**
- First instance checks if port 40000 is available
- Port free → Start as SERVER
- Port occupied → Start as CLIENT (connect to localhost)

**Benefits**
- No manual key combinations needed
- Faster iteration during development
- Multiple editor/build instances "just work"

**Disabling Auto-Detection**
- Select GONet_GlobalContext in Hierarchy
- Inspector → GONetGlobal component
- Uncheck "Enable Auto Role Detection"
- Falls back to manual server/client startup (keyboard shortcuts or
command line args)

## Next Steps - Add Network Functionality

### Basic Networking (Dead Simple)

**Sync GameObject Transform**
- Add `GONetParticipant` component to any GameObject
- Transform position/rotation/scale automatically synchronize across
network
```

- That's it! No additional configuration needed.

Sync Custom Fields
- Add your MonoBehaviour script to a GameObject with GONetParticipant
- Mark fields with `[GONetAutoMagicalSync]` attribute:
```csharp
public class PlayerStats : MonoBehaviour
{
    [GONetAutoMagicalSync] public float health = 100f;
    [GONetAutoMagicalSync] public int score = 0;
    [GONetAutoMagicalSync] public Vector3 velocity;
}
```
- Fields automatically sync to all clients
- Code generation happens automatically on save/compile

Network Spawning
- Use `GameObject.Instantiate(prefab)` as normal
- Objects with GONetParticipant automatically spawn across network
- Server/owner authority assigned automatically

### Remote Procedure Calls (RPCs)

GONet v1.5 includes a robust RPC system with async/await support:

```csharp
public class MyNetworkedScript : GONetBehaviour
{
    [ServerRpc]
    void RequestAction(int param)
    {
        // Runs on server when called by client
        Debug.Log($"Server received action request: {param}");
    }

    [ClientRpc]
    void NotifyAllClients(string message)
    {
        // Runs on all clients when called by server
        Debug.Log($"All clients notified: {message}");
```

~~~
    }

    [TargetRpc]
    void SendToSpecificClient(ushort targetClientId, int data)
    {
        // Targeted delivery to specific client
        Debug.Log($"Received targeted message: {data}");
    }
}
```

### Scene Management (New in v1.5)

Server-authoritative networked scene loading:

```csharp
// SERVER: Load scene directly
GONetMain.SceneManager.LoadSceneFromBuildSettings("NextLevel",
LoadSceneMode.Single);

// CLIENT: Request scene change (requires server approval)
GONetMain.SceneManager.RequestLoadScene("NextLevel");

// Optional validation hook (server-side)
GONetMain.SceneManager.OnValidateSceneLoad += (sceneName, mode,
requestingClient) => {
    // Return false to deny request
    return requestingClient == expectedClientId;
};
```

### Unity Addressables Support (New in v1.5)

GONet v1.5 adds full support for Unity Addressables - for both scenes AND
runtime prefab spawning!

**Addressables Scenes:**
```csharp
// Load Addressables scene (server)
~~~

~~~markdown
GONetMain.SceneManager.LoadSceneFromAddressables("DynamicArena",
LoadSceneMode.Additive);
```

**Addressables Prefabs:**
```csharp
// No code changes needed!
// GONet automatically detects addressables:// paths in metadata and loads
from Addressables
GameObject.Instantiate(weaponPrefab);
```

**Benefits:**
- **No Resources folder restrictions** - Organize prefabs anywhere in your
project
- **Efficient asset bundles** - Use Addressables groups for optimization
- **Platform-specific variants** - Different assets per platform
- **Cleaner project organization** - Better asset management

**Setup:**
1. Install Unity Addressables package (Package Manager)
2. Mark scenes/prefabs as Addressable (right-click → Addressables → Make
Addressable)
3. GONet automatically detects `#if ADDRESSABLES_AVAILABLE` and uses
Addressables loading
4. Mix Resources and Addressables freely (backward compatible)

## Configuration (Advanced)

Most features work with default settings, but GONet v1.5 offers extensive
configuration in the GONetGlobal component:

**GONetId Batch System** (New in v1.5)
- Pre-allocated ID ranges for client spawning (eliminates spawn round-trip
latency)
- Default: 200 IDs per batch
- Adjust: `client_GONetIdBatchSize` (100-1000)

**Congestion Management** (New in v1.5)
- Adaptive pool scaling automatically adjusts to network demand
~~~

```
- Default: Enabled (recommended)
- Configure: `enableAdaptivePoolScaling`, `maxPacketsPerTick`

**Value Blending Buffer**
- Controls interpolation/extrapolation smoothness
- Default: 100ms buffer lead time
- Adjust: `valueBlendingBufferLeadTimeMilliseconds` (0-1000ms)

**Sync Bundle Deferral** (New in v1.5)
- Handles race conditions during rapid spawning
- Default: Disabled (industry standard - drop-first approach)
- Enable for turn-based games: `deferSyncBundlesWaitingForGONetReady`

## Troubleshooting

### Compilation Errors After Import
**Symptom:** Red errors in Console
**Solution:** Verify "Allow unsafe Code" is enabled and API Compatibility
Level is set correctly

### No Server/Client Starts Automatically
**Symptom:** Scene runs but no network activity
**Solution:**
- Check Console for errors
- Verify GONet_GlobalContext prefab is in scene
- Check "Enable Auto Role Detection" setting in GONetGlobal

### Objects Not Syncing
**Symptom:** GameObject changes on one machine don't appear on others
**Solution:**
- Verify GameObject has GONetParticipant component
- Check authority (IsMine property - only owner can modify)
- Review Console logs for sync errors

### Code Generation Errors
**Symptom:** "CodeGenerationId mismatch" or "Key not present in
dictionary"
**Solution:**
- Right-click affected prefab → Reimport
- Verify `Assets/StreamingAssets/GONet/DesignTimeMetadata.json` exists
- Check `Assets/GONet/Code/GONet/Generation/` folder has generated files

### Late-Joining Client Issues
**Symptom:** Client connecting after game started doesn't see correct state
**Solution:**
- GONet v1.5 includes automatic late-joiner synchronization
- Check Console for "SceneLoadEvent" and "GONetId assignment" messages
- Verify persistent events are enabled for critical RPCs (IsPersistent = true)

## Support
If any issues arise, please reach out for assistance via:

- **Discord:** https://discord.gg/NMeheRHQgd
- **Email:** contactus@galoreinteractive.com
- **Website:** https://galoreinteractive.com/gonet

## What's Next?
**Explore Sample Code**
- Open `Assets/GONet/Sample/` folder
- `GONetSampleScene.unity` - Complete working example
- `GONetSampleInputSync.cs` - Input synchronization
- `GONetSampleSpawner.cs` - Network spawning
- `GONetSampleChatSystem.cs` - RPC validation example

**Read the Manual**
- Comprehensive API documentation available on website
- Discord community for questions and discussions
- Active development with regular updates

**Join the Community**
- Share your multiplayer game progress
- Get help from experienced GONet developers
- Contribute to the growing ecosystem

**Congratulations!** You're now ready to build networked multiplayer games with GONet v1.5!
```

Visit the product website: https://galoreinteractive.com/gonet