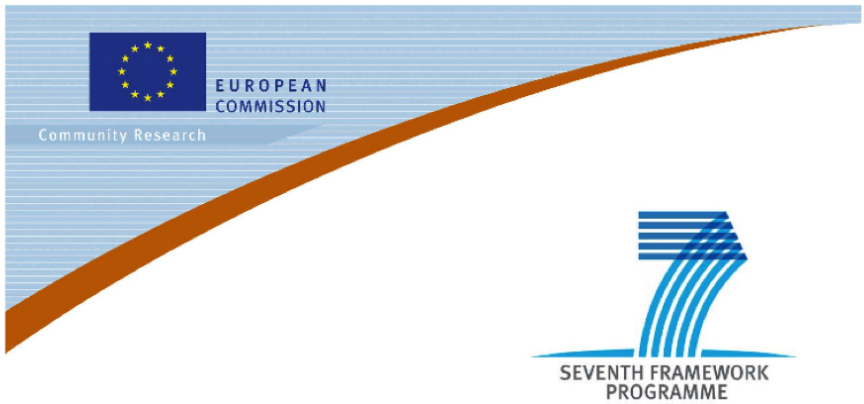

Private Public Partnership Project (PPP)

Large-scale Integrated Project (IP)

Synchronization (Tundra) Installation and Administration Guide

Project full title: Future Internet Core
Project acronym: FI-Core
Contract No.: 632893

Contents

Synchronization - Installation and Administration Guide

Introduction

This document describes installing and running the server and client parts of the Synchronization GE.

System Requirements

The server part of the Synchronization GE is based on the realXtend Tundra SDK, therefore its hardware requirements are the same as of the Tundra SDK.

Hardware Requirements

Server requirements

- Minimum of 1GB memory
- About 40GB free hard disk space when building Tundra, 1GB free when running
- Intel-compatible CPU with SSE instruction support, minimum processor speed 1.66 GHz
- If running graphics, a GPU with minimum of 256MB memory

Operating System Support

Server requirements

- Windows XP or newer
- Ubuntu Linux 14.04 or newer. Other distributions may work but are not guaranteed.

The client code is operating system independent, as it's JavaScript code running in a browser.

Software Requirements

The client part of the Synchronization GE requires a browser that has WebSocket support, such as

- Internet Explorer 10.0+
- Firefox 6.0+
- Chrome 14.0+
- Safari 6.0+
- Opera 12.1+
- iOS Safari 6.0+
- Android Browser 4.4+
- Opera Mobile 12.1+
- Chrome for android 31.0+
- Firefox for Android 25.0+
- Internet Explorer Mobile 10.0+

For serving the synchronization client HTML and JavaScript code, an HTTP server such as Apache can be used. For testing use the browser can also be pointed to local files.

Software Installation and Configuration

Server installation

There are two options, to download a prebuilt binary package, or build from source.

Prebuilt binaries (Linux)

A prebuilt binary is provided for Ubuntu 14.04 64bit.

To install a Tundra package and its dependencies, use these commands. You need to have root privileges and an active Internet connection for downloading dependencies.

- apt-get update
- dpkg -i PackageFileName.deb
- apt-get -f install

After successful installation, the Tundra binary and examples scenes are placed in the /opt/realxtend-tundra directory.

Note! "dpkg -i PackageFileName.deb" will print missing dependencies error, but "apt-get -f install" should find & install them.

The server is also available as a Docker image:

- https://hub.docker.com/r/loorni/synchronization-tundra-urho3d//

Building from source

Clone the Tundra git repository at https://github.com/realXtend/tundra-urho3d.git/ Then follow the README.md file in the root directory of the clone for build instructions.

After a successful build of Tundra, it can be run from the bin subdirectory. On Windows there are Tundra.exe and TundraConsole.exe executables, where TundraConsole will display a console window and is therefore recommended for server usage. On Linux there is just an executable called Tundra.

Testing the installation (all methods)

To load an example scene with server mode enabled for testing, run the following command. On Windows, replace ./Tundra with TundraConsole:

If your using prebuild binaries on Linux, Tundra executable and example scenes can be found at /opt/realxtend-tundra-urho3d directory

Client installation

The client part of the Synchronization GE is supplied as part of the WebTundra JavaScript libraries. There are two options for getting it:

- Download a packaged release
- Clone the git repository at https://github.com/realXtend/WebTundra/ (dev2 branch)

To test connecting to a localhost synchronization server, open the file html/client.html in your browser from the source code clone or extracted package, then click "Connect" in the displayed login screen. You may need to serve the client source directory in a web server such as Apache or node.js server due to permission problems with the local file:// protocol.

Sanity check procedures

End to End testing

For both the server and the client, it is recommended to have at least 1 GB free RAM and 1 GB free hard disk space, a broadband network connection, and to ensure that the system is not unnecessarily loaded with other processes.

Real-time Synchronization

When the Tundra server is running using the command line above, and the client HTML page html/client.html is open, you should be able to test connecting to the localhost server by clicking the "Connect" button. At this point the JavaScript synchronization code should connect to the Tundra server. You should see the following in Tundra server console:

The connection ID may not be exactly the same.

Also, when inspecting the client page using a web developer console, you should see the log message "Server connection established" after clicking "Connect".

Seeing these prints confirms that a WebSocket connection between the Tundra server and the JavaScript client code is functioning, and synchronization can be used.

List of Running Processes

Server: Tundra or TundraConsole

Network interfaces Up & Open

Port 2345, both TCP & UDP. This is the default Tundra port for serving scenes using the real-time synchronization and can be changed with the –port startup parameter.

Databases

N/A

Diagnosis Procedures

If connection fails between the server and the client, verify the following:

- that the server was started with the –server command line parameter
- that the WebSocketServerModule is loaded on the server; the row "Loading plugin WebSocketServerModule" should be printed to the server console during server startup, with no error messages following it
- that firewall is not blocking port 2345
- if the client and server are not on the same machine, or the server port is not the default 2345, make sure that you supply the correct address and port number in the client login screen.

Resource availability

The server memory consumption should be around 50 MB when running a simple scene without rendering. With rendering enabled, it may be between 100 MB and 200 MB. A typical desktop web browser typically uses around 100 MB of memory when showing a Synchronization client application. CPU usage depends on the amount of activity in the networked scene, but should be when there is nothing happening.

Remote Service Access

N/A

Resource consumption

As resource consumption is dependent on the scene complexity, the scripts that are running in it, the amount of client connections to the server and their activity, it is hard to give directions to what kind of eg. a memory consumption on the server would be considered abnormal. One guideline would be to note the amount of memory in use once a server has started up and loaded the scene: if memory use for example rapidly doubles

from that figure and continues to grow unbounded, then it's likely that an application script is misbehaving and leaking memory.

In a demanding application or scene that uses physics and heavy scripting, it is not uncommon to see high CPU use figures by the server (up to full utilization of one processor core) with high numbers of active clients. For an example, consider a high number (50-100) of clients connected to a scene running an "avatar application": each client gets an avatar to move around, which is physically simulated and uses scripts for movement, collision response etc. Rectifying high CPU utilization in such case will require either using more powerful hardware, simplifying the application (for example not using actual physics simulation, or simplifying the scripts) or limiting the amount of users that can simultaneously connect.

Note that this is considering the processing on the Tundra server process as a whole. The network synchronization itself is fairly lightweight in its processing requirements.

I/O flows

The Synchronization GE will by default use TCP (WebSocket) traffic on the port 2345. Additionally native (Tundra) clients will use UDP traffic on the same port. The amount of traffic on the server will depend the scene being served and the amount of connected clients; typical would be in the order of tens to hundred kilobytes per second.