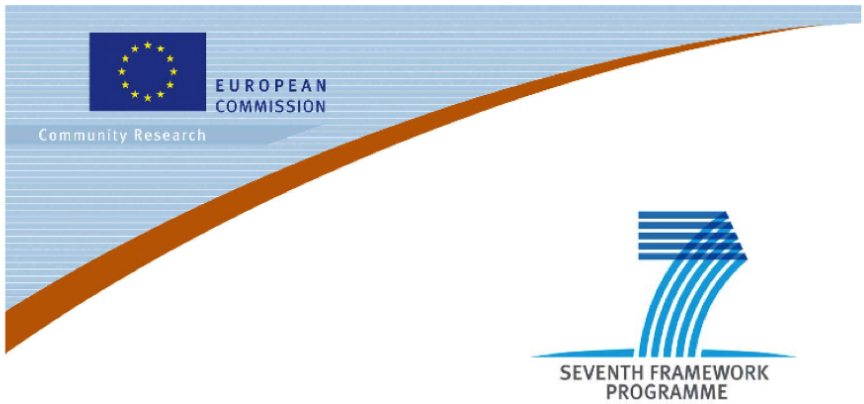

Private Public Partnership Project (PPP)

Large-scale Integrated Project (IP)

Synchronization (FiVES) Installation and Administration Guide

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

Contents

Introduction

This document describes installing and running the server and client parts of the FiVES Synchronization GE implementation. The main purpose of this Generic Enabler is the real time synchronization of data between 3D web clients. Popular use case scenarios are virtual 3D environments and multi user scenarios, with users being represented as 3D avatars. FiVES is designed to simplify creation and deployment of such virtual worlds by a flexible and modular architecture.

System Requirements

The server part of FiVES is provided as C# implementation that can be built and run on both Windows and Linux, using Mono 3.3. The client is a HTML / JavaScript application and is based on XML3D.

Hardware Requirements

Server requirements:

For small to medium scenes, FiVES runs on a standard work station, that is

- 4GB of memory
- 10 GB disk space
- Dual core CPU with 2.33 GHz (Gigahertz)

Operating System Support

Server requirements

- Windows 7 or newer
- Ubuntu Linux 12.04 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 supports WebSocket and WebGL. As the client is based on the 3D-UI-XML3D Generic Enabler, the browser needs to be able to run XML3D, that is, you may use:

- Google Chrome on Windows, Mac OS X and Android
- Mozilla Firefox on Windows, Mac OS X and Android
- Microsoft Internet Explorer 11
- Safari on iOS

In addition, you need an Apache web server to serve the HTML5 / JavaScript client.

Software Installation and Configuration

Server Installation

Prebuilt binaries (Windows)

A prebuilt binary version of FiVES can be downloaded from the FIWARE Forge Repository:

- https://forge.fiware.org/frs/download.php/1632/fives-fiware-4.3.zip/

The files in the archive need to be unzipped in a folder on the hard drive. Building from source

The FiVES source code is available on GitHub under LGPL v3 License:

- http://github.com/fives-team/fives/

FiVES is provided as single solution that contains the FiVES core project, a set of most commonly used plug-ins, and all required third party libraries. Some of these library dependencies are resolved using the NuGet package manager. To build FiVES from source in either Windows or Linux, please perform the following steps:

- Install NuGet package manager for your IDE: http://www.nuget.org/
- Open the solution in your IDE, right click on the solution item in the project browser, and select "Restore NuGet packages". Alternatively, configure NuGet to restore the packages automatically during the build process
- Build the entire solution

After the build process, all binary files for plug-ins, libraries and their configuration files, as well as the FiVES executable, are copied into the folder /Binaries, which is a direct subfolder of the FiVES project root folder.

Configuring the server

FiVES uses a slightly adapted version of the KIARA Advanced Middleware GE for server-to-client and server-to-server communication. The place to look for network configuration is the file SINFONIPlugin.dll.config in the Binaries folder, or the app.cfg file of the SINFONIPlugin project within the Plugins folder of the FiVES solution in the IDE. The configuration file looks like this:

```
<configuration>
 <ServerConfiguration>

  <!-- Specifies the listener to which clients connect in order retrieve…   -->
  <!-- This server configuration document will contain the URI to the IDL as well…  -->
  <!-- the implemented services according to the KIARA specification -->
  <ConnectionListener host ="+" port="8181" path="/fives/" />

  <!-- FiVES will host all Plugin-provided service under a combined service in…  -->
  <!-- specification. This service will run on the same IP as the server as…   -->
  <ServiceConfiguration host="Any" transport="ws" protocol="fives-json" port="34837" />

 </ServerConfiguration>

  <Paths>
    <ProtocolPath value = "./SINFONI" />
    <TransportPath value = "./SINFONI" />
  </Paths>
</configuration>
```

This configuration file defines two connection endpoints: The first one, ConnectionListener, is used by clients to request information about the server. The second one specifies the actual address for the realtime synchronization service. Please note that the host address may have to be changed from "Any" to the actual public IP of the machine on which the server is running.

Testing the installation (all methods)

To test the server installation, just execute FiVES.exe by either

- Windows: double clicking FiVES.exe
- Linux: navigate to the folder where FiVES.exe is located and execute it via

```
mono FiVES.exe
```

In the default configuration, this should give you the following output in the console:

```
The server is up and running. Use‘ quit’ command to stop it
>>
```

Typing plugins should print you the names of all plugins that were loaded successfully. No plugin should be marked as deferred or failed.

Tundra Protocol

FiVES currently uses a JSON-based serialization protocol that is part of the SINFONI middleware. Please refer to the FiVES GitHub Repository (http://github.com/fives-team-FiVES/) for more detailed information about SINFONI and other components FiVES is using.

However, there is a prototypical implementation of a FiVES Plugin set that enables a Tundra compatible communication. Please note that also with this Plugin set, FiVES does not – and does not intend to – replace the RealXtend Synchronization GE reference implementation for communication with WebTundra. Also, for full compatibility, all Tundra components need to be implemented also in FiVES. For more information about Tundra and the Tundra protocol, please refer to https://github.com/realXtend/tundra/wiki/Tundra-protocol/ . The WebTundra Communication package can be installed by the following steps:

1) Check out the repository at http://github.com/tospie/fives-wt-communication/
2) Set up and build the project according to the documentation of the GitHub repository.
3) Your built WebTundra communication Plugin set together with all dependencies will be contained in the Build folder of the project

While overwriting all existing files in the respective folders, perform the following steps:

1) Copy all .dll files into the folder where the your FiVES plugins are loaded from. This path is given in the FIVES.exe.config file as PluginDir .
2) Copy WTProtocol.dll into the folder where your SINFONI modules are loaded from. This path is given in the SINFONIPlugin.dll.config file as ProtocolPath
3) Copy the pre-configured FiVES.exe.config from the Build folder to the folder where your FiVES executable is located
4) Copy the pre-configured SINFONI.dll.config from the Build folder to the folder where SINFONIPlugin.dll is located.

Run now FiVES as usual. Your FiVES deployment is now running in WebTundra compatibility mode!

Client Installation

The example Web client is entirely contained in the WebClient subfolder of the FiVES project. It is entirely based on HTML5, JavaScript, and libraries based on these, one of the most important being XML3D that is used for the description of interactive 3D scenes that are provided by FiVES.

To run the WebClient, you basically need nothing but an ordinary Apache web server. Copy the entire WebClient folder somewhere within the folder from which your Apache server serves the websites, usually htdocs or www.

Client Configuration

The Web client is designed and written such that it basically runs immediately out of the box. However, if the default network configuration of the server is changed, you will need to make some slight changes to the client's configuration file. This file can be found in the subfolder kiara, named fives.json:

```
{
"info": "TestServer",
"idlURL": "kiara/fives.kiara",
"servers": [{
  "services": "*",
  "protocol": {
    "name": "fives-json",
    "port": 34837,
    "host": "localhost"
    }
  }]
}
```

For the Web client to successfully connect to FiVES, make sure that the entry for "protocol" matches the entries you have made for the ServiceConfiguration on the server side.

Sanity Check Procedures

End to End testing

Real-time Synchronization

When the server is running, access the web client via Web browser. You should be prompted with a login screen. Enter arbitrary credentials. The login window should disappear, and a 3D avatar should be crated for you.

Typing nc in the console should prompt you with the following output:

```
The server is up and running. Use 'quit' command to stop it...
>> nc
Number of connected clients: 1
```

You can control your avatar using the WASD keys. Pressing one of the keys should lead to the avatar moving or rotating.

List of Running Processes

Server: FiVES.exe (mono FiVES.exe in Linux)

Network interfaces Up & Open

By default, FiVES opens a TCP WebSocket on port 34837. This port may be changed within the server configuration file. In addition, FiVES opens an HTTP listener on port 8181, path /fives'// . Accessing

```
http://localhost:8181/fives/ (Default configuration of connection listener)
```

via web browser should prompt a server configuration document.

Databases

N/A

Diagnosis Procedures

When the connection between client and server fails, check the following points:

- Check whether correct host and port are entered in the client's configuration file
- Check whether the console of the server prints any error message.
- Check whether all needed plugins are loaded on server side:
- Type plugins into the server console
- Check if a plugin that is requested by the client was not loaded.
- Check if a plugin that is requested by the client is mark as deferred.
- Check if your firewall is configured correctly
- We have experienced issues with some anti virus programs that cause trouble for WebSocket connections. Try disabling virus scanners and connect again

Resource availability

The server memory consumption should be around 50 MB when running a simple scene. 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 close to 0 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.

I/O flows

The Synchronization GE implementation FiVES will by default use TCP (WebSocket) traffic on the port 34837. 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.