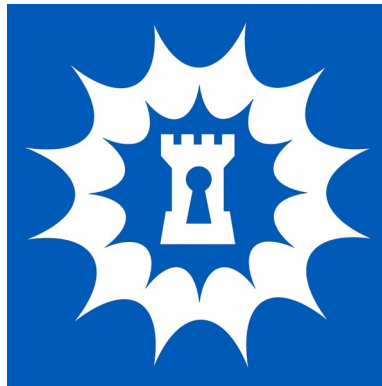


# RSocketpol

*Reliable Socket Policy Server for Unity*



Documentation

**crosstales** LLC

Date: 14. January 2015

Version: 1.2.0

## Table of Contents

1. Overview.....	3
2. Features.....	4
3. Installation.....	4
3.1. Prerequisites.....	4
3.1.1. Java on Windows.....	4
3.1.2. Java on Mac.....	4
3.1.3. Java on Unix/Linux.....	5
3.2. Install the server.....	5
4. Configuration.....	6
4.1. standard.properties.....	6
4.1.1. Server settings.....	6
4.1.2. Test application settings.....	7
4.2. Logback.xml.....	8
5. Run.....	9
5.1. Server.....	9
5.2. Test application.....	10
5.3. Unity test scene.....	11
6. Problems.....	12
7. Release notes.....	13
8. Contact.....	13

## 1. Overview

Whenever you need your web-based game/application to access a web address (e.g. download asset bundles, access a RESTful server), this web address needs a „Socket policy server“ to be accessible:

<http://docs.unity3d.com/Manual/SecuritySandbox.html>

We were using the standard „sockpol.exe“ from Unity quite a while, until we discovered how unreliable and dangerous this software is.

Here are some of our concerns:

- because of the **fixed port 843**, which is an official system port (all ports below 1024 are system reserved ports), it needs to run as **root** under a Mac or Linux machine. This means that, if a bad guy somehow gains control over the „sockpol“-process, he could do anything on your machine – like wipe all data from the system... **This is a severe and unnecessary security risk!**
- Any established connection to the „sockpol“ stays open until the server receives the command „<policy-file-request/>“. When you like to monitor your „sockpol“ process with a tool like [CheckHost](#), the server will end up with **unclosed connections** and every connection **consumes a lot of performance** – We realized that one unclosed connection leads up to 50% CPU consumption. Again, if a bad guy wanted to do ugly stuff like a „denial-of-service“ of your socket policy server by sending simple socket connections, he could easily do that. **That would bring your server down and hinder your real customer from using your product/services!**
- „sockpol.exe“ **doesn't write any log files**. For us, it's very interesting to know what's going on on our servers. We would like to know, how many connections had our servers handled etc.
- **Lack of configuration options** – port is fixed in code, time-out not implemented
- **No simple tests** for the socket policy server available
- To run „sockpol.exe“ under **Mac or Linux**, you have to install **Mono**. In our case, we had to install the whole thing on server for this little „EXE“ - we don't use any „Mono“ specific apps, so for us, it's just an **unnecessary dependency**. But this is very individual – probably you use it and love it :-)

We could, of course, implement the necessary changes in the source file „socketpol.cs“ and build an „EXE“. Nevertheless, the Mono-issue would still be there. Since we have over 15 years of experience in Java we decided to write our own implementation. Some of you may say that we „just replace one evil with an other“ - but that's not true. Java has proved to be a very stable, fast and widely used language.

## 2. Features

- Non-blocking, **reliable** socket policy server alternative for Unity
- **Port**, time-out and queue size for incoming connections are freely **configurable**
- Full **customizable logging** (incl. rolling file appenders with size settings)
- **Multi-threaded**
- Much **faster** response time (20%-50%)
- **Test-scene** for Unity
- Configurable **test-application** (load-test with threads and iterations)
- Runs on **Windows, Mac and Unix/Linux**
- Extensive **tests, documentation** and **support!**
- Full C# and Java **source code** provided
- We are **committed** to all our assets! This means, we will add new features over time!

## 3. Installation

### 3.1. Prerequisites

**Java 7** must be installed on your **target** system. To verify, enter the following command in the system prompt:

```
java -version
```

The result should look like this:

```
java version "1.7.0_51"  
Java(TM) SE Runtime Environment (build 1.7.0_51-b13)  
Java HotSpot(TM) 64-Bit Server VM (build 24.51-b03, mixed mode)
```

The „java version“ should be „1.7“ or higher – if that's the case, move on to 3.2.

If you get an error instead saying something like „command not recognized“, you have to install Java (see below).

#### 3.1.1. Java on Windows

Download and install:

<https://www.java.com/en/download/>

#### 3.1.2. Java on Mac

Download and install:

<https://www.java.com/en/download/>

### 3.1.3. Java on Unix/Linux

Download an install:

[https://www.java.com/en/download/help/linux\\_x64rpm\\_install.xml](https://www.java.com/en/download/help/linux_x64rpm_install.xml)

Or you can use the free alternative OpenJDK:

<http://openjdk.java.net/install/>

## 3.2. Install the server

Do the following steps:

1. **Unzip** the „RSockpol.zip“ to a desired folder
2. optional: copy the folder to your server

That's it, unless your target is a Unix/Linux machine. Then you have to add the execution-flag to „rsockpol.sh“ like this:

```
chmod +x rsockpol.sh
```

“RSockpol“ will work out of the box and be available at the port 65432. If that's fine with you, feel free to continue to 5.

## 4. Configuration

„RSockpol“ has two different configuration files:

- standard.properties – Server and test application related settings, like port, timeout, queue size etc.
- logback.xml – Settings for the logging system, like log file destination, size settings etc.

### 4.1. standard.properties

The file content can be edited with any text editor and looks like this:

```
#####
# Reliable Socket Policy Server 1.2.0 #
# Properties #
#####

port = 65432
timeout = 2000
queue = 200
save.interval = 5000

# Clients
server.url = localhost
server.port = 65432
client.iterations = 5
client.threads = 2
client.mode = reality
```

#### 4.1.1. Server settings

Key	Value	Description
port	0 – 65535 standard: <b>65432</b>	Listener port for the socket policy server. This port must be reachable by your web-based game/application. You have to be sure to forward it correct and set the appropriate firewall settings. <i>We recommend to choose a value greater than 1023.</i>
timeout	20 – 60000 standard: <b>2000</b>	Socket time-out in milliseconds. Defines the time before the server automatically closes a connection. <i>We don't recommend setting this value lower than 200 and higher than 5000.</i>
queue	20 – 50000 standard: <b>200</b>	Defines how many parallel request are queued. We don't recommend to set this value too high except you have literally tons of parallel request.
save.interval	500 – 600000 standard: <b>5000</b>	Saves the statistics in a defined interval (in milliseconds).

#### 4.1.2. Test application settings

Key	Value	Description
server.url	ip or hostname standard: <b>localhost</b>	Socket policy server url. This can be an ip-address or an hostname (without protocol) like „myserver.com“.
server.port	0 – 65535 standard: <b>65432</b>	Socket policy server port. This is the listener port of the server. If you forward this port to another port, it must be the „outside“ reachable port.
client.iterations	1 – 99999 standard: <b>5</b>	Defines how many times the test application will run all the threads (tests). Don't set this value too high or you will have to wait a long time.
client.threads	1 – 50000 standard: <b>2</b>	Defines how many threads (parallel request tests) are started per iteration. Too many threads will dramatically slow your test machine – so be careful!
client.mode	<i>reality</i> or <i>unity</i> standard: <b>reality</b>	Connection test with (unity) and without (reality) sending a command to the server. Important: the reality-mode will block the standard „sockpol.exe“ completely. Try it out :-)

## 4.2. Logback.xml

The file content can be edited with any text editor and looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <appender name="mainFileAppender"
class="ch.qos.logback.core.rolling.RollingFileAppender">
    <file>./logs/rsockpol.log</file>
    <append>true</append>

    <encoder>
      <pattern>%date{ISO8601} [%thread] %-5level %logger{35} - %msg
%n</pattern>
    </encoder>

    <rollingPolicy
class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
      <fileNamePattern>./logs/rsockpol.log.%i</fileNamePattern>
      <minIndex>1</minIndex>
      <maxIndex>3</maxIndex>
    </rollingPolicy>

    <triggeringPolicy
class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
      <maxFileSize>5MB</maxFileSize>
    </triggeringPolicy>
  </appender>

  <logger name="com.crosstales" level="INFO" />

  <root level="ERROR">
    <appender-ref ref="mainFileAppender" />
  </root>
</configuration>
```

We marked the relevant parts **bold**.

In there you can change the file location, rolling policy and logfile size.

Furthermore, it's possible to set the debug-level, e.g. ERROR, INFO, DEBUG.

For more informations and advanced configurations, take a look at the „Logback“ site:

<http://logback.qos.ch/manual/index.html>



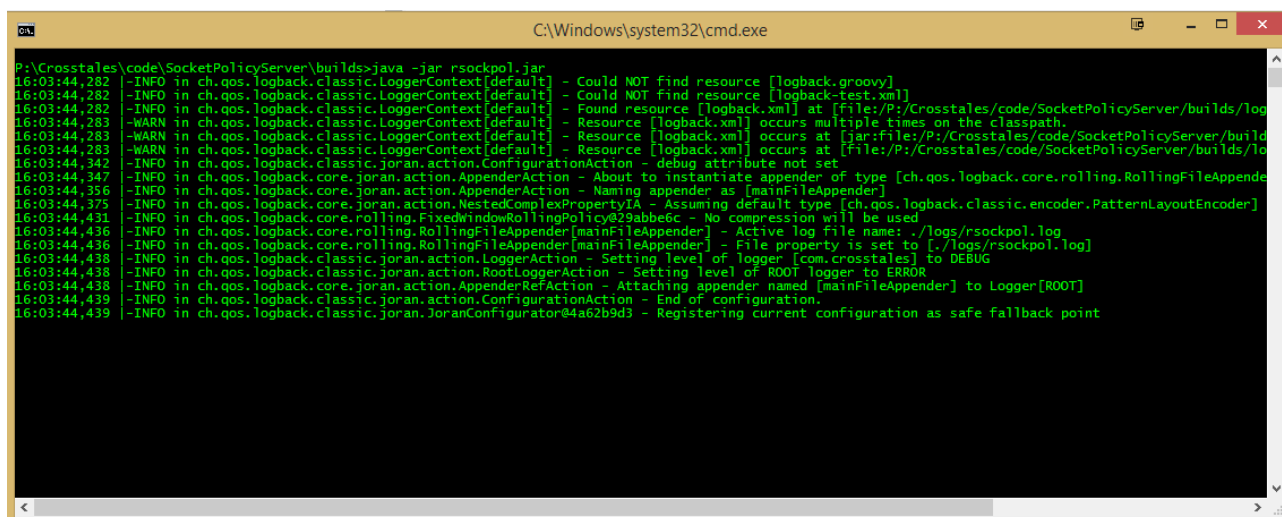
## 5. Run

### 5.1. Server

You have two possibilities to start the server:

1. on the command prompt, go to your „RSockpol“ directory and enter:  
**java -jar rsockpol.jar**
2. Double-click/run the corresponding script file for your system:
  - o Windows: rsockpol.bat
  - o Mac: rsockpol.command
  - o Unix/Linux: rsockpol.sh

After the start, you should see something like this:



```
C:\Windows\system32\cmd.exe
P:\Crosstales\code\SocketPolicyServer\builds>java -jar rsockpol.jar
16:03:44,282 -INFO in ch.qos.logback.classic.LoggerContext[default] - Could NOT find resource [logback.groovy]
16:03:44,282 -INFO in ch.qos.logback.classic.LoggerContext[default] - Could NOT find resource [logback-test.xml]
16:03:44,282 -INFO in ch.qos.logback.classic.LoggerContext[default] - Found resource [logback.xml] at [file:/P:/Crosstales/code/SocketPolicyServer/builds/logback.xml]
16:03:44,283 -WARN in ch.qos.logback.classic.LoggerContext[default] - Resource [logback.xml] occurs multiple times on the classpath.
16:03:44,283 -WARN in ch.qos.logback.classic.LoggerContext[default] - Resource [logback.xml] occurs at [jar:file:/P:/Crosstales/code/SocketPolicyServer/builds/logback.xml]
16:03:44,283 -WARN in ch.qos.logback.classic.LoggerContext[default] - Resource [logback.xml] occurs at [file:/P:/Crosstales/code/SocketPolicyServer/builds/logback.xml]
16:03:44,342 -INFO in ch.qos.logback.classic.joran.action.ConfigurationAction - debug attribute not set
16:03:44,347 -INFO in ch.qos.logback.core.joran.action.AppenderAction - About to instantiate appender of type [ch.qos.logback.core.rolling.RollingFileAppender]
16:03:44,356 -INFO in ch.qos.logback.core.joran.action.AppenderAction - Naming appender as [mainFileAppender]
16:03:44,375 -INFO in ch.qos.logback.core.joran.action.NestedComplexPropertyIA - Assuming default type [ch.qos.logback.classic.encoder.PatternLayoutEncoder]
16:03:44,431 -INFO in ch.qos.logback.core.rolling.FixedWindowRollingPolicy@29abb6c - No compression will be used
16:03:44,436 -INFO in ch.qos.logback.core.rolling.RollingFileAppender[mainFileAppender] - Active log file name: ./logs/rsockpol.log
16:03:44,436 -INFO in ch.qos.logback.core.rolling.RollingFileAppender[mainFileAppender] - File property is set to [./logs/rsockpol.log]
16:03:44,438 -INFO in ch.qos.logback.classic.joran.action.LoggerAction - Setting level of logger [com.crosstales] to DEBUG
16:03:44,438 -INFO in ch.qos.logback.classic.joran.action.RootLoggerAction - Setting level of ROOT logger to ERROR
16:03:44,438 -INFO in ch.qos.logback.core.joran.action.AppenderRefAction - Attaching appender named [mainFileAppender] to Logger [ROOT]
16:03:44,439 -INFO in ch.qos.logback.classic.joran.action.ConfigurationAction - End of configuration.
16:03:44,439 -INFO in ch.qos.logback.classic.joran.JoranConfigurator@4a62b9d3 - Registering current configuration as safe fallback point
```

While the process is executed, you can take a look at the log file to see what's happening.

To **stop** the server press **Ctrl+C**.

## 5.2. Test application

To run the test application, open the command prompt and go to your „RSockpol“ directory. Then enter:

```
java -cp rsockpol.jar com.crosstaes.rsockpol.client.SocketPolicyClient
```

*Good:*

```

C:\Windows\System32\cmd.exe
P:\Crosstaes\code\SocketPolicyServer\builds>java -cp rsockpol.jar com.crosstaes.rsockpol.client.SocketPolicyClient
16:39:41,591 -INFO in ch.qos.logback.classic.LoggerContext[default] - Could NOT find resource [logback.groovy]
16:39:41,591 -INFO in ch.qos.logback.classic.LoggerContext[default] - Could NOT find resource [logback-test.xml]
16:39:41,592 -INFO in ch.qos.logback.classic.LoggerContext[default] - Found resource [logback.xml] at [file:/P:/Crosstaes/code/SocketPolicyServer/builds/1
16:39:41,592 -WARN in ch.qos.logback.classic.LoggerContext[default] - Resource [logback.xml] occurs multiple times on the classpath.
16:39:41,592 -WARN in ch.qos.logback.classic.LoggerContext[default] - Resource [logback.xml] occurs at [jar:file:/P:/Crosstaes/code/SocketPolicyServer/bui
16:39:41,643 -INFO in ch.qos.logback.classic.joran.action.ConfigurationAction - debug attribute not set
16:39:41,650 -INFO in ch.qos.logback.core.joran.action.AppenderAction - About to instantiate appender of type [ch.qos.logback.core.rolling.RollingFileAppen
16:39:41,658 -INFO in ch.qos.logback.core.joran.action.AppenderRefAction - Naming appender as [mainFileAppender]
16:39:41,677 -INFO in ch.qos.logback.core.joran.action.NestedComplexPropertyIA - Assuming default type [ch.qos.logback.classic.encoder.PatternLayoutEncoder
16:39:41,726 -INFO in ch.qos.logback.core.rolling.FixedWindowRollingPolicy@2ca33d2c - No compression will be used
16:39:41,732 -INFO in ch.qos.logback.core.rolling.RollingFileAppender[mainFileAppender] - Active log file name: ./logs/rsockpol.log
16:39:41,732 -INFO in ch.qos.logback.classic.joran.action.LoggerAction - Setting level of logger [com.crosstaes] to DEBUG
16:39:41,734 -INFO in ch.qos.logback.classic.joran.action.RootLoggerAction - Setting level of ROOT logger to ERROR
16:39:41,735 -INFO in ch.qos.logback.core.joran.action.AppenderRefAction - Attaching appender named [mainFileAppender] to Logger[ROOT]
16:39:41,735 -INFO in ch.qos.logback.classic.joran.action.ConfigurationAction - End of configuration.
16:39:41,736 -INFO in ch.qos.logback.classic.joran.JoranConfigurator@11052a99 - Registering current configuration as safe fallback point

Total time (ms): 8.547926000
Time per request (ms): 0.884557000
Total Requests: 1
Fails %: 0.0
Iterations: 5
P:\Crosstaes\code\SocketPolicyServer\builds>
    
```

You can see the infos of the test, like the total execution time for all request etc.

*Bad* - if something went wrong (server not running or reachable), it looks like this:

```

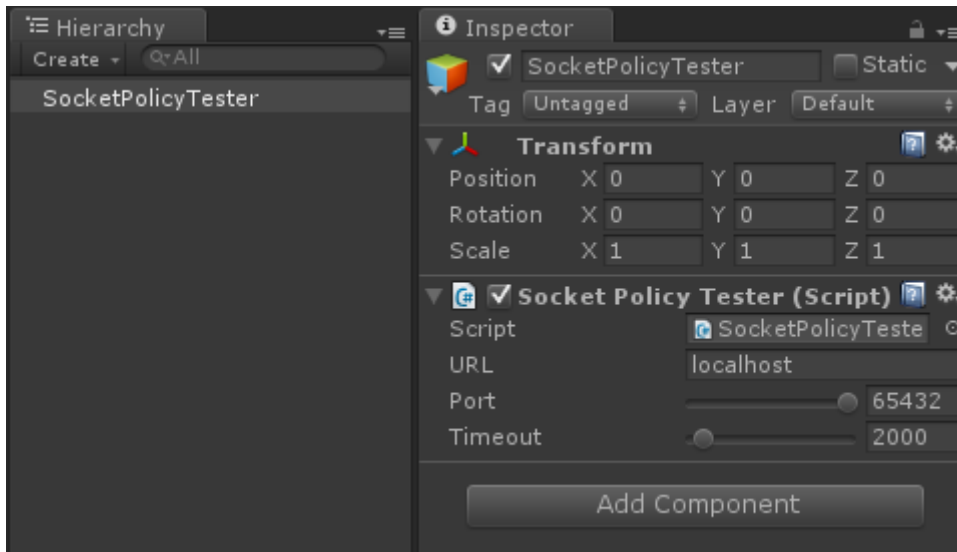
C:\Windows\System32\cmd.exe
P:\Crosstaes\code\SocketPolicyServer\builds>java -cp rsockpol.jar com.crosstaes.rsockpol.client.SocketPolicyClient
16:48:27,137 -INFO in ch.qos.logback.classic.LoggerContext[default] - Could NOT find resource [logback.groovy]
16:48:27,137 -INFO in ch.qos.logback.classic.LoggerContext[default] - Could NOT find resource [logback-test.xml]
16:48:27,137 -INFO in ch.qos.logback.classic.LoggerContext[default] - Found resource [logback.xml] at [file:/P:/Crosstaes/code/SocketPolicyServer/builds/1
16:48:27,138 -WARN in ch.qos.logback.classic.LoggerContext[default] - Resource [logback.xml] occurs multiple times on the classpath.
16:48:27,138 -WARN in ch.qos.logback.classic.LoggerContext[default] - Resource [logback.xml] occurs at [jar:file:/P:/Crosstaes/code/SocketPolicyServer/bui
16:48:27,138 -INFO in ch.qos.logback.classic.joran.action.ConfigurationAction - debug attribute not set
16:48:27,190 -INFO in ch.qos.logback.core.joran.action.AppenderAction - About to instantiate appender of type [ch.qos.logback.core.rolling.RollingFileAppen
16:48:27,196 -INFO in ch.qos.logback.core.joran.action.AppenderRefAction - Naming appender as [mainFileAppender]
16:48:27,203 -INFO in ch.qos.logback.core.joran.action.NestedComplexPropertyIA - Assuming default type [ch.qos.logback.classic.encoder.PatternLayoutEncoder
16:48:27,222 -INFO in ch.qos.logback.core.rolling.FixedWindowRollingPolicy@11052a99 - No compression will be used
16:48:27,273 -INFO in ch.qos.logback.core.rolling.RollingFileAppender[mainFileAppender] - Active log file name: ./logs/rsockpol.log
16:48:27,280 -INFO in ch.qos.logback.classic.joran.action.LoggerAction - Setting level of logger [com.crosstaes] to DEBUG
16:48:27,280 -INFO in ch.qos.logback.classic.joran.action.RootLoggerAction - Setting level of ROOT logger to ERROR
16:48:27,282 -INFO in ch.qos.logback.core.joran.action.AppenderRefAction - Attaching appender named [mainFileAppender] to Logger[ROOT]
16:48:27,282 -INFO in ch.qos.logback.classic.joran.action.ConfigurationAction - End of configuration.
16:48:27,283 -INFO in ch.qos.logback.classic.joran.JoranConfigurator@0a2fab89 - Registering current configuration as safe fallback point

Total time (ms): 5037.475222000
Time per request (ms): 503.772924000
Total Requests: 1
Fails %: 100.0
Iterations: 5
P:\Crosstaes\code\SocketPolicyServer\builds>
    
```

During the test run you can take a look at the log file to see what's happening. If the test application runs **too long**, press **Ctrl+C** to terminate the process.

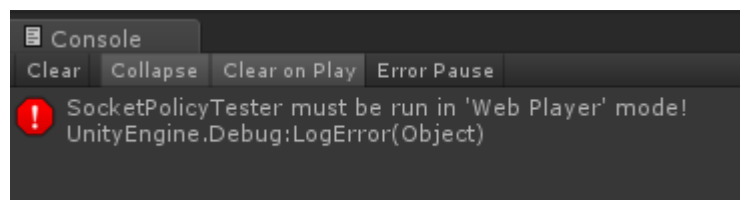
### 5.3. Unity test scene

There is also a test scene called „Test“ inside the Unity package. Load and configure it:



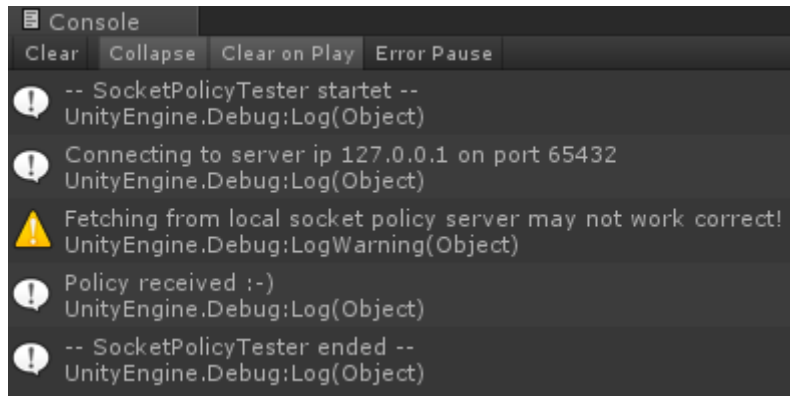
Key	Value	Description
URL	ip or hostname standard: <b>localhost</b>	Socket policy server url. This can be an ip-address or an hostname (without protocol) like „myserver.com“.
Port	0 – 65535 standard: <b>65432</b>	Socket policy server port. This is the listener port of the server. If you forward this port to another port, it must be the „outside“ reachable port.
Timeout	10 – 60000 standard: <b>2000</b>	Socket time-out in milliseconds. Defines the time before the test client automatically closes a connection. We don't recommend setting this value lower than 200 and higher than 4000.

Make sure you switched to „Web player“ mode, or you get an error like this:

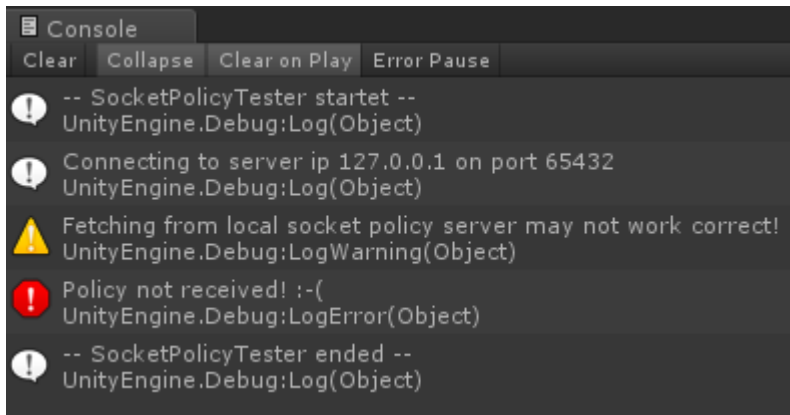


Hit „**Play**“ in Unity!

**Good:**



**Bad** - if the server is not running or reachable, you get the following message:



**Important:**

When you run the „Socket Policy Server“ on the same machine as the „Test scene“, the fetch will only work once (see warnings above) until you restart Unity or change the port of the server!

Try to use different machines for the server- and test-applications.

## 6. Problems

If you encounter any problems with this asset, just [send us an email](#) with a problem description, the Unity version and the invoice number and we will try to solve it.

## 7. Release notes

- 2015-02-14
  - Minor improvements
  - Java source code added
- 2015-01-08
  - Documentation updated
- 2015-01-01
  - Added an interval-based save-function for the statistics
  - Minor improvements
- 2014-12-01
  - First release 1.0.0 submitted to the Unity AssetStore

## 8. Contact

**crosstales** LLC  
Bullingerstrasse 53  
CH-8004 Zürich

Homepage: <http://www.crosstales.com/en/assets/rsockpol/>

Email: [assets@crosstales.com](mailto:assets@crosstales.com)

AssetStore: <https://www.assetstore.unity3d.com/en/#!/content/24860>

Forum: <http://forum.unity3d.com/threads/rsockpol-reliable-socket-policy-server-for-unity.289865/>

Documentation: <http://www.crosstales.com/en/assets/rsockpol/RSocketpol-doc.pdf>