|  |
| --- |
|  |
| SIP Sorcery |
| Experimental SIP Software |

|  |
| --- |
| Aaron Clauson  26 May 2009 |

Contents

[1.0 Overview 3](#_Toc231116718)

[2.0 General Configuration 3](#_Toc231116719)

[3.0 SIP Server Agent Configurations 4](#_Toc231116720)

[3.1 Application Server 4](#_Toc231116721)

[3.2 Monitor Server 4](#_Toc231116722)

[3.3 Stateless Proxy 5](#_Toc231116723)

[3.4 Registrar 5](#_Toc231116724)

[3.5 Registration Agent 5](#_Toc231116725)

# 1.0 Overview

This document details the configuration and settings for the SIPSorcery Server Agents.

# 2.0 General Configuration

Each of the SIPSorcery applications stores their settings in an XML configuration file that has the same name as the executable but with a “.config” extension. For example the configuration file for the sipsorcery-appsvr.exe process is sipsorcery-appsvr.exe.config.

The format of the configuration files follows that of the standard .Net [Application Configuration](http://msdn.microsoft.com/en-us/library/ms229689(vs.71).aspx) files. The configSections node lists the other nodes in the file and which class and process is responsible for processing them. This section is very relevant for the SIPSorcery applications and it is critical that the process name correctly matches the executing process. Each SIPSorcery server agent has its own configuration node from which it will load its specific settings from.

|  |  |
| --- | --- |
| Server Agent | Configuration Node |
| Monitor | sipmonitor |
| Stateless Proxy | sipproxy |
| Registrar | sipregistrar |
| Registration Agent | sipregistrationagent |
| Application Server | sipappserver |

The configuration file also contains an appSettings node which contains settings that are common to multiple server agents. Examples of common settings are the ones for persistence settings which determine where the server agents that rely on persistent storage will look up and store their data.

The list of common settings and their values is shown in the table below.

|  |  |
| --- | --- |
| Setting | Description |
| PersistenceStorageType | Determines the type of persistent storage that the Server Agents will use. Can be one of XML|DBLinqPostgresql|DBLinqMySQL which inidicate XML file based storage, a Postgresql database or MySQL database respectively. |
| PersistenceConnStr | The directory for the XML persistence option or the database connection string for the DBLinq options. Example: C:\Temp\sipsorcery\xmlconfig\. |

The other common node in the configuration file is log4net the node. This node controls the logging options for the application and provides a large variety of different logging options and mechanisms. Full information on configuring logging can be obtained from the [log4net](http://logging.apache.org/log4net/index.html) web site.

There are other common elements within each of the agent specific nodes. The most significant one is the sipsockets node. This node controls the configuration of the SIP transport layer for the particular server agent and an example is show below.

<sipsockets>

<socket>\*:5060</socket>

<socket>10.0.0.1:5090</socket>

<socket protocol="tcp">\*:5060</socket>

<socket protocol="tcp">10.0.0.2:5091</socket>

<socket protocol="tls" certificatepath="server.pfx">\*:5061</socket>

</sipsockets>

The use of the \* character indicates that all local IPv4 addresses should be utilised by the SIP transport layer and will be listened on for SIP traffic.

# 3.0 SIP Server Agent Configurations

## 3.1 Application Server

The Application Server’s settings are read from the sipappserver node.

|  |  |
| --- | --- |
| Setting | Description |
| MonitorLoopbackPort | The loopback port the server will send application log and notification messages to. Example: 10001. |
| TraceDirectory | The directory to store the Application Server dial plan traces. Example: c:\temp\sipsorcery\traces\. |
| RubyScriptCommonPath | The path to the file that contains the common Ruby script that will be added to the start of every user’s dial plan. Example: c:\temp\sipsorcery\dialplan-common.rby. |
| OutboundProxy | The SIP end point that the server will use when sending SIP requests. Can be empty if an outbound proxy is not being used. Example: udp:10.1.1.2:5060. |
| WebServiceEnabled | Whether or not the web service end points should be enabled. Example: True. Omitting the setting or leaving it blank will disable the web services. |
| sipsockets | The XML configuration node that controls the initialisation of the server agent’s SIP transport layer. See the [General Configuration](#_2.0_General_Configuration) chapter for details. |

If the Application Server has web services enabled then it will rely on a system.serviceModel node to configure the service endpoints and behaviours. The web service software infrastructure employed is [Microsoft’s Windows Communication Foundation (WCF)](http://msdn.microsoft.com/en-us/library/ms731082.aspx). For correct operation of the web services with the Application Server no changes should be made to the node except for the baseAddress attributes which dictate the IP socket that will be listened on for web service clients. By default the web service socket is configured to listen on all local IPv4 addresses and port 8080. The XML configuration node that does that is show below.

<host>

<baseAddresses>

<add baseAddress="http://\*:8080/provisioning"/>

</baseAddresses>

</host>

If it’s desired that the web service operates on a different IP address or port then the baseAddress attributes can be changed an example of which is shown below.

<host>

<baseAddresses>

<add baseAddress="http://10.0.0.1:1090/provisioning"/>

</baseAddresses>

</host>

Changing the socket will have implications for any clients such as the SIPSorcery GUI which are hard coded to look for the web service end point on a particular socket.

## 3.2 Monitor Server

The Monitor Server’s settings are read from the sipmonitor node.

|  |  |
| --- | --- |
| Setting | Description |
| MonitorLoopbackPort | The loopback port the server will send application log and notification messages to. Example: 10001. |
| SilverlightPolicyFilePath | Example: silverlight-policy.xml. |
| sipmonitorclientsockets | The XML configuration node that controls which sockets the Monitor Server will listen on for connections from human clients. An example node is shown below. |
| sipmonitormachinesockets | The XML configuration node that controls which sockets the Monitor Server will listen on for connections from software processes that want to be notified of certain events from Server agents. An example node is shown below. |

The XML configuration nodes below show an example of how to configure the sockets the Monitor Server will listen on for human and machine connections. An IP address value of \* indicates that all local IPv4 addresses should be listened on.

<sipmonitorclientsockets>

<socket>10.0.0.1:11001</socket>

<socket>\*:4502</socket>

</sipmonitorclientsockets>

<sipmonitormachinesockets>

<socket>\*:4503</socket>

</sipmonitormachinesockets>

## 3.3 Stateless Proxy

The Stateless Proxy’s settings are read from the sipproxy node.

|  |  |
| --- | --- |
| Setting | Description |
| MonitorLoopbackPort | The loopback port the server will send application log and notification messages to. Example: 10001. |
| ProxyScriptPath | The path to the script file that controls the routing of SIP messages through the Proxy. The script is critical for the operation of the Proxy and without it the Proxy does not know how to route SIP traffic and will drop any SIP packets it receives. The extension of the file can be either .py to indicate the script is Python or .rb to indicate the script is Ruby. Example: C:\Temp\sipsorcery\proxyscript.py. |
| NATKeepAliveSocket | The socket to listen for NAT keep-alive requests from a Registrar on. Example: 127.0.0.1:9001. |
| sipsockets | The XML configuration node that controls the initialisation of the server agent’s SIP transport layer. See the [General Configuration](#_2.0_General_Configuration) chapter for details. |

## 3.4 Registrar

The SIP Registrar’s settings are read from the sipregistrar node.

|  |  |
| --- | --- |
| Setting | Description |
| MonitorLoopbackPort | The loopback port the server will send application log and notification messages to. Example: 10001. |
| MaximumAccountBindings | This value dictates the maximum number of bindings the Registrar will permit per SIP account. Example: 10. |
| NATKeepAliveRelaySocket | The socket to send NAT keep-alives to. Typically the Proxy will listen for NAT keep-alive requests from the Registrar and this parameter must match the on NATKeepAliveSocket setting on the Proxy. It is the connection between the Proxy and User Agent that needs to be held open on end user NATs. Example: 127.0.0.1:9001. |
| useragentconfigs | The XML configuration node that controls two aspects of the Registrar’s operation depending on the UserAgent string reported by the registering user agent. See below for details. |
| sipsockets | The XML configuration node that controls the initialisation of the server agent’s SIP transport layer. See the [General Configuration](#_2.0_General_Configuration) chapter for details. |

The XML configuration node below shows an example of the user agent configuration that can be applied to the Registrar. Each XML element in the useragentconfigs node attempts to match the UserAgent header on REGISTER requests. The value in the useragent element is a regular expression and the first node that matches is the one that will be used. The last node should always be the “.\*” catch all regular expression to ensure that every user agent is matched.

The expiry attribute controls the maximum allowed expiry that will be accepted for a user agent. The reason for employing this setting is that different user agents have been found to not work properly if they cannot set their requested expiry value. For example the Fring user agent attempts to set an expiry of 3600 seconds and assumes that it will accepted irrespective of the value set on the binding by the Registrar. The consequence of that is the Registrar will expire the binding when the Fring user agent does not renew its binding.

The contactlists attribute controls whether the Registrar will return a list of all current bindings in Ok responses as mandated by the SIP standard or whether it will only return the single Contact header that was received in the REGISTER request. The reason for this is that some user agents do not operate properly if the list of current contacts is returned and will assume their registration request has failed unless the single Contact they specified is returned in the response.

<useragentconfigs>

<useragent expiry="3600" contactlists="true">fring</useragent>

<useragent expiry="3600" contactlists="false">Nokia</useragent>

<useragent expiry="60" contactlists="false">Cisco</useragent>

<useragent expiry="113">.\*</useragent>

</useragentconfigs>

## 3.5 Registration Agent

The Registrar Agent’s settings are read from the sipregistrationagent node.

|  |  |
| --- | --- |
| Setting | Description |
| MonitorLoopbackPort | The loopback port the server will send application log and notification messages to. Example: 10001. |
| OutboundProxy | The SIP end point that the server will use when sending SIP requests. Can be empty if an outbound proxy is not being used. Example: udp:10.1.1.2:5060. |
| sipsockets | The XML configuration node that controls the initialisation of the server agent’s SIP transport layer. See the [General Configuration](#_2.0_General_Configuration) chapter for details. |