SIP Sorcery

Configuration Guide

Aaron Clauson

11 Feb 2010

Document History

Date	Author	Description	Version
23 Aug 2009	Aaron Clauson	Created for v1.1 local release.	1.1
11 Feb 2010	Aaron Clauson	Updated for v1.2 local release.	1.2
16 Feb 2010	Aaron Clauson	Incorporate feedback from mgbfan78	

Contents

1.0 Overview

This document details the configuration and settings for installing the SIPSorcery Server Agents as a single executable that is suitable for small numbers of users. The single executable deployment is referred to as a local install as recognition that it's suitable for installing locally on a PC with a typical home internet connection.

The SIPSorcery source along with this and other documents is available at the SIPSorcery codeplex site. A community forum dedicated to the SIPSorcery project is hosted at SIPSorcery forums. A blog that discusses SIPSorcery and related technologies is hosted at SIPSorcery blog.

2.0 Quickstart

This chapter lists the minimum steps required to get SIPSorcery servers up and running.

1. Download the latest version from SIP Sorcery releases,
2. Edit the **sipdomains.xml** file and set the domain value to the hostname or IP address that your SIP user agents will be accessing the host running the SIPSorcery application. For example if the local SIPSorcery instance is running on a machine with an IP address of 192.168.0.3 the contents of the sipdomains.xml would be along the lines of the XML below. Other aliases may need to be added if the instance is using additional ports or if a hostname is being used. If the computer being used has a **dynamic IP address** then the **domain will need to be a hostname** otherwise the **sipdomains.xml** file will need to be updated each time the dynamic IP address changes.

```
<sipdomains>
  <sipdomain>
   <domain>192.168.0.3</domain>
   <owner></owner>
   <sipdomainaliases>
     <domainalias>192.168.0.3:5060</domainalias>
     <domainalias>local</domainalias>
     <domainalias>*</domainalias>
   </sipdomainaliases>
  </sipdomain>
</sipdomains>
```

3. Start the SIPSorcery application from a command line with:
 sipsorcery-appsvr.exe –c
4. Connect to the SIPSorcery Silverlight GUI using the URL below, the usernames and passwords are taken from **Customers.xml** and the default values are username/password. There is a problem using FireFox to open the URL as the HTTP Content-Type header is incorrect and FireFox tries to save the page rather than open it, Internet Explorer works correctly:
http://localhost:8080/sipsorcery.html
5. Once the Silverlight GUI is open it is possible to create new user records. Part of the user creation process involves confirming the entered email address. If no mail server is available this can be done manually by updating the **emailaddressconfirmed** field to **True** of if using a database to **'1'**,
6. When creating new SIP Provider entries the register contact will assume a host of sipsorcery.com by default. The sipsorcery.com should be replaced by the IP address or public hostname of the computer running the sipsorcery instance.

If the above steps are successful and you wish to run the SIPSorcery server as a Windows Service instead of a console application perform these additional steps:

1. Access permissions for the SIPSorcery application server to allow it to utilise HTTP service endpoints need to be set. Open a DOS box as an administrator using runas. At the command prompt enter the command below:
   Vista or Windows 7: **netsh http add urlacl url=http://+:8080/ user="NT Authority\Local Service"**
   XP or 2k3: **httpcfg set urlacl /u http://+:8080/ /a D:(A;;GX;;;LS)**
   (the httpcfg utility for XP can be downloaded from XP SP2 Support Tools),
2. On the command line install the service using:
   installutil sipsorcery-appsvr.exe

To configure the SIPSorcery servers to use a Postgresql , MySQL, MSSQL (including SQL Azure) database instead of the XML data store perform these additional steps:

1. Create a new database call sipsorcery on your database server,
2. Open the query tool and paste in the DDL from **\sipsorcery-servers\SQL\sipsorcery-mssql.sql** for a Microsoft SQL database or from **\sipsorcery-servers\SQL\sipsorcery-others.sql** for a MySQL or Postgresql database at latest SIPSorcery Source,
3. Open the **sipsorcery-appsvr.exe.config** file and replace and search for the `appSettings` XML node and set:
   MySQL:
   ```
    <add key="PersistenceStorageType" value="SQLLinqMySQL" />
   <add key="PersistenceConnStr" value="Database=sipsorcery;Data
   Source=localhost;User Id=root;Password=password" />
   ```
   Microsoft SQL (including SQL Azure):
   ```
   <add key="PersistenceStorageType" value="SQLLinqMSSQL" />
   <add key="PersistenceConnStr" value="tcp:server;Database=sipsorcery;User
   ID=sa;Password=password;Trusted_Connection=False;Encrypt=True;" />
   ```
   Postgresql:
   ```
   <add key="PersistenceStorageType" value="SQLLinqPostgresql" />
   <add key="PersistenceConnStr"
   value="Database=sipsorcery;Host=localhost;User
   Id=postgres;Password=password" />
   ```
4. Insert a record into the sipdomains table as shown below and where "myhostname" matches the hostname or IP address your SIP users will use to connect:
   insert into sipdomains values ('7dcc3cf9-7687-4e29-add3-1b97ba545088', 'mydomain', 'local;*', null, '2010-02-09T13:01:21.3540000+00:00');

3.0 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 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`
SSH Server	`sshserver`

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|SQLLinqPostgresql|SQLLinqMySQL|SQLLinqMSSQL which indicate XML file based storage, a Postgresql database, a MySQL database or a Microsoft SQL database respectively.
PersistenceConnStr	The directory for the XML persistence option or the database connection string for the DBLinq options. Example: C:\Temp\sipsorcery\xmlconfig\.
UserDataDBType	The type of database connection for the DBWrite and DBRead dialplan applications. Can be one of MSSQL|MySQL|Postgresql.
UserDataDBConnStr	The database connection string for the DBWrite and DBRead dialplan applications.
HTTPServerBaseDirectory	An optional setting for the directory the WCF hosted services will use as a base directory for loading the sipsorcery.html and clientbin/sipsorcery.xap files. If empty the directory the application is running from will be used.

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 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.

4.0 SIP Server Agent Configurations

4.1 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`.
sipsockets	The XML configuration node that controls the initialisation of the server agent's SIP transport layer. See the 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). 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 would also need to have the URL it uses for the services updated. That can be done on the initial screen using the **Set Service** link.

4.2 Monitor Server

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

Setting	Description
`MonitorLoopbackPort`	The loopback port the Monitor Server will listen on for application log and notification messages to. Example: `10001`.

The monitor server exposes a WCF HTTP endpoint that is consumed by the SSH Server and by the Silverlight client. This endpoint is what allows the log messages from all the SIPSorcery server agents to be delivered to client applications, if it is disabled then no messages will show up in the Silverlight console tab or in SSH Server sessions.

4.3 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`.
`STUNServerHostname`	Optional setting that if set will result in the Proxy daemon creating a thread to send STUN client requests every 60s to attempt to determine the public IP address it is accessible on. The public IP address will be conveyed to the Application Server and used in the SDP when sending external call requests. Example: `stun.ekiga.com`.
`sipsockets`	The XML configuration node that controls the initialisation of the server agent's SIP transport layer. See the General Configuration chapter for details.

4.4 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 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 chapter for details.

5.0 Runtime Data

The SIPSorcery server agents require a persistent data store in order to be able to operate properly. The data that needs to be stored are:

- Customers: Username, passwords and other data for each user,
- SIP Accounts: Username, passwords and configuration options for each SIP Account,
- SIP Dial Plans: Dial plan scripts for users,
- SIP Domains: The domain names that the SIPSorcery application server will process,
- SIP Providers: The list of 3rd party SIP Providers under management for each user.

The options available for data store persistence are:

- XML files,
- Postgresql relational database,
- MySQL relational database,
- Microsoft SQL relational database (including SQL Azure).

XML files are the easiest to set up and manage and are the recommended option for small installations. The use of an XML file (or any flat file) approach for a high number of concurrent writes is not ideal and in the case of the SIPSorcery servers there can be issues writing Call Detail Records (CDRs) in high call volume installs. In that case of if CDRs are critical an XML file approach is not recommended and instead a relational database should be used.

For the relational databases supported there are two different schemas supported one for Microsoft SQL and the other for Postgresql and MySQL. The latest versions can be downloaded from **\sipsorcery-servers\SQL\** using SVN or through a browser at latest SIPSorcery Source.

5.1 Minimal Runtime Data Configuration

The SIPSorcery servers will run without any runtime data configured but no call, registrations or other SIP operations will be possible since the servers will not know anything about the users or domains attempting them.

The first thing that needs to be configured is the SIP domains so that the SIPSorcery servers can differentiate between incoming and outgoing calls. Without the differentiation the Application Server does not know whether a call it receives should be challenged for authentication or whether it should be treated as an incoming public call.

5.1.1 SIP Domains Configuration

An example of a minimal SIP Domain XML file is:

```
<sipdomains>
 <sipdomain>
  <domain>mydomain</domain>
  <sipdomainaliases>
```

```
    <domainalias>local</domainalias>
    <domainalias>10.1.1.2</domainalias>
    <domainalias>*</domainalias>
   </sipdomainaliases>
  </sipdomain>
</sipdomains>
```

In this example **mydomain** is the host name or IP address that user agents will have configured as their SIP Server or Proxy setting. The SIP From header is used to identify whether the caller should be challenged or not and if the From header URI host portion contains the domain or an alias of a domain the request will be challenged.

For requests that don't contain a From header with a host portion matching a configured domain the request URI will be inspected. If the request URI contains a host portion with a matching domain it will be processed as an incoming call. If it doesn't a domain not serviced response will be returned.

For relational data stores each domain and its aliases are stored in a single record an example of the insert statement to achieve the same configuration as the XML file above is:

insert into sipdomains values ('7dcc3cf9-7687-4e29-add3-1b97ba545088', 'mydomain', 'local;10.1.1.2;*', null, '2010-02-09T13:01:21.3540000+00:00');

The **aliaslist** column in the **sipdomains** table can be empty or contain a list of semi-colon separated aliases for the domain. When a request is received by a SIPSorcery server and there is no direct match on a sipdomain then a check will be done against each alias and if a match is found the request will be treated as if it was for the domain the alias belongs to. The asterisk '*****' character represents a special wild card alias. When it is present as an alias on a domain and there is no other matching domain or alias it will be matched and the domain it belongs to will be used.

Once the SIP Domains have been configured then it will be possible to create SIP Accounts for those domains and make and receive calls and accept registrations.

5.1.2 Customer, SIP Account, SIP Provider and SIP Dial Plans Configuration

It is possible to create and modify all the assets the SIPSorcery servers use by directly editing either the XML files or relational database records. However apart from the SIPDomains it is recommended that the SIPSorcery Silverlight client be used to manage the assets. The client is able to create, update and delete Customer, SIP Account, SIP Provider and SIP Dial Plans.

If manual editing is desired there are examples of the XML files at **\sipsorcery-servers\SIPSorcery.SIPAppServer\ExampleXMLConfig** using SVN or through a browser at latest SIPSorcery Source.

6.0 Miscellaneous

On IIS to get the WSDL file to hold the public hostname rather than the private one use:
cscript //nologo %systemdrive%\inetpub\adminscripts\adsutil.vbs set W3SVC/1/SecureBindings ":443:www.fancydomain.com"

For more info http://blogs.msdn.com/wenlong/archive/2007/08/02/how-to-change-hostname-in-wsdl-of-an-iis-hosted-service.aspx.