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

|  |
| --- |
| Aaron Clauson  4/21/2009 |

Contents

[Overview 3](#_Toc228079968)

[SIP Server Agents 3](#_Toc228079969)

[Application Server 3](#_Toc228079970)

[Monitor Server 3](#_Toc228079971)

[SIP Proxy 3](#_Toc228079972)

[SIP Registrar 4](#_Toc228079973)

[SIP Registration Agent 4](#_Toc228079974)

[How to Use 4](#_Toc228079975)

# Overview

The SIP Sorcery project revolves around a SIP protocol stack and a set of experimental SIP server applications built on top of it. The SIP Sorcery project has evolved out of an earlier project called MySIPSwitch which has been running as a publicly accessible SIP service from Nov 2006. SIP Sorcery is predominantly a name change for the MySIPSwitch project but has been combined with a major software upgrade which has included enhancements to the SIP Stack, the SIP Server Agents and has also added a new rich client interface based on Silverlight to replace the previous HTML and Javascript based interface.

The SIP Sorcery software is all based on C# and version 3.5 SP1 of the .Net framework. The various assemblies that make up the project fall into one of four main categories:

* Core -This category indicates an assembly is part of the SIP Sorcery “plumbing”. It includes the assembly which implements the SIP protocol stack, **SIPSorcery.SIP.Core**, as well as five other assemblies that implement services like persistence, network services such as DNS and also more application type services such as managing resources such as SIP Accounts, SIP Providers and Customer records,
* Servers – This category indicates an assembly is a SIP Server Agent and represents an application that can be run to provide a specific SIP service such as a Stateless SIP Proxy or SIP Registrar,
* Silverlight – This category indicates the assembly is part of the new Silverlight user interface and will be run within the Silverlight plugin inside a web browser,
* SoftPhone – This category indicates the assembly is part of a new SoftPhone which has been implemented predominantly as a test tool for the Core and Server parts of the project. The SoftPhone is extremely rudimentary and does not implement any audio processing.

# SIP Server Agents

## Application Server

The Application Server is a SIP B2BUA which can process SIP call traffic based on user customisable dial plans.

## Monitor Server

The Monitor Server is a monitoring agent that captures log messages from other server agents and that also provides a telnet interface for end users to connect and view the log messages.

## SIP Proxy

The SIP Proxy Server is a stateless SIP Proxy that utilises either a Ruby or Python script to control SIP message flows.

## SIP Registrar

The SIP Registrar is a standard SIP Registrar that processes user agent binding requests.

## SIP Registration Agent

The SIP Registration Agent can be thought of as the opposite of a SIP Registrar, it sends binding requests to external SIP Registrars.

# How to Use

The SIP Server Agents provide a number of pre-canned applications that can be used as is or tweaked as needed. For different SIP applications where there is no pre-existing agent the SIP protocol stack can be quickly and easily incorporated.

In order to use the SIP stack all that’s needed is a reference to **SIPSorcery.SIP.Core**. Before making use of the stack there are a small number of SIP concepts that need to be understood so the stack can be configured to operate in the best way for an application.

* The SIP standard specifies a number of different network and transport layer protocols that SIP can operate on. The SIP Sorcery stack supports UDP, TCP and TLS. When configuring the SIPSorcery SIP stack the first step is to inform it on which and what type of endpoints it needs to operate with,
* SIP uses transactions to link together requests and responses. Most SIP agents do need to operate with transactions but some such as Stateless SIP Proxy’s do not. The SIPSorcery SIP stack can operate in either mode as shown in the code example below.

As an example application a code sample is provided that sends a periodic OPTIONS request to a SIP agent to check whether it is alive.