The inGate SIParator is an enterprise Session Border Controller (eSBC) that can be deployed in a number of different configurations (see http://www.ingate.com/files/422/sipmanual-en/xc974.html). This guide details the steps I went through to configure DMZ-LAN SIParator hosted on a virtual machine with a second virtual machine acting as a PBX on the same LAN. This is a valid way of deploying the SIParator but, as stated, not the only one. This guide, therefore, should be used as an introduction and a pointer to setting up a SIParator and is not intended to replace the wealth of documentation and help available on inGate’s website. For example, if you have a physical SIParator and are not running it on a VM, the first section below, “Setting up the Virtual Machines” isn’t going to be any help to you and you should find installation instructions on inGate’s website, instead.

The other thing worth noting is that inGate have a setup tool that one can download from their website that helps with most of the basic configuration. I was unable to run this tool because I could not deploy a Windows VM on the same LAN as my SIParator VM. Most of the SIParator setup that I explain in this guide would be handled by the setup tool.

Contents:
- Setting up the Virtual Machines
- Configuring your Twilio Elastic SIP trunk
- Configuring The inGate SIParator
  - Basic configuration
  - Network Configuration
  - SIP Configuration
  - SIP Trunking
  - SIP Routing
- Setting up TLS and SRTP
- Appendix 1
Setting up the Virtual Machines

The first thing you will need to do is to get an ISO image for the SIParator from inGate. They have a guide on how to do this, and on how to install the VM SIParator, [here](#).

If you are using the VM version of the SIParator, you need to be able to boot from an ISO image. If you have local hosts then this is not so hard. If you want to build the VMs in the cloud, this becomes a bit more difficult. I found Ravello ([www.ravellosystems.com](http://www.ravellosystems.com)) who allow you to build VMs in either AWS for the Google Cloud and boot them from ISO. They have instructions on how to do so at [https://www.ravellosystems.com/blog/create-configure-publish-vm-ravello-using-iso-image/](https://www.ravellosystems.com/blog/create-configure-publish-vm-ravello-using-iso-image/).

Using the above, I build a VM SIParator and I then built a standard CentOS VM to act as an Asterisk PBX. When I was done, I had the following topology.
The inGate VM had two ethernet ports:

- **eth0**
  - 10.0.2.3 / 255.255.255.0
  - LAN interface
- **eth1**
  - 10.0.1.3 / 255.255.255.0
  - DMZ interface

If you are using the SIParator in a LAN/DMZ configuration, it is very important that you use eth0 as the LAN interface and eth1 as the DMZ interface. If you have them the other way round, the SIParator will not correctly bridge media.

The CentOS6.3 VM, my Asterisk PBX, had one ethernet port:

- **NIC #0**
  - 10.220.2.4 / 255.255.255.0

I gave every interface a public IP address.

- eth0 on the inGate VM needed a public address so that I could access it via http, for configuration via the web UI
- eth1 on the inGate VM needed a public address so that I could send SIP and RTP from it to Twilio and vice versa
- the CentOS VM needed a public address so that I could register remote phones against the Asterisk PBX running on it

On ravello, if you want to be able to access a VM on a certain port, you need to configure it with services. I created the following

- **inGate**
  - **eth0**
    - HTTP, external allowed
    - SIP
    - RTP
  - **eth1**
    - HTTP, external allowed
    - SIP, external allowed
    - RTP, external allowed
    - TLS, external allowed (port 5061)
- **Asterisk on CentOS**
  - **NIC #0**
    - SIP
    - RTP

I configured Asterisk on the CentOS server using the help of Google.
Configuring your Twilio Elastic SIP trunk

You can find instructions on how to build your Twilio Elastic SIP trunk at [https://www.twilio.com/docs/sip-trunking/getting-started](https://www.twilio.com/docs/sip-trunking/getting-started). I have included screenshots of the trunk I set up, with comments, to make the configuration of the inGate clearer.

**InGate on Ravello**

**Properties**

- **Friendly Name**: InGate on Ravello
- **Trunk SID**: TK219001c473dd6b183d4d4b8df3c10

Configure a Friendly name to identify this Elastic SIP Trunk. The Trunk SID is the unique identifier of this Trunk, and is assigned automatically.

**Call Recording**

- **Recording Setting**: Do not Record

**Secure Trunking**

Encryption ensures that the call media and associated signalling remains private during transmission. Transport Layer Security (TLS) provides encryption for call signaling and Secure Real-time Transport Protocol (SRTP) provides encryption for call content/media packets.

- **Enabled ($0.001 per minute)**: When Secure Trunking is enabled, TLS must be used to encrypt SIP messages on port 5061, and SRTP must be used for the media packets. Any non-encrypted calls will be rejected.
- **Disabled**: When Secure Trunking is disabled, RTP must be used for media packets and SIP messages may be sent in the clear or using TLS. Any SRTP encrypted calls will be rejected.

**Save**  **Cancel**  **Delete this Trunk**
Ingate on Ravello

Termination: Outgoing traffic from your communications infrastructure to the PSTN. In order to use a Trunk for termination it must have a Termination SIP URI and at least one authentication scheme (IP Access Control Lists and/or Credentials Lists).

**Termination URI**

- **Termination SIP URI**: ingate pstn.twilio.com
- Configure a SIP Domain Name to uniquely identify your Termination SIP URI for this Trunk. This URI will be used by your communications infrastructure to direct SIP traffic towards Twilio. When you point your infrastructure toward this URI, Twilio uses a Geo DNS lookup to intelligently direct your traffic to our closest POPL.
- Learn more about Termination Settings »

**Authentication**

- **IP Access Control Lists**: Ingate on Ravello
- **Credential Lists**: Click to select an Credential List

**Save**  **Cancel**  **Delete this Trunk**
Ingate on Ravello

General  Termination  Origination  Numbers

Origination: Incoming traffic to your communications infrastructure from the PSTN.

Origination URI

Configure the IP address (or FQDN) of the network element entry point into your communications infrastructure (e.g. IP-PBX, SBC). Provisioning for high service availability: show more.

Secure Trunking is enabled on this Trunk. TLS will be used to encrypt SIP messages towards your communications infrastructure. If the transport parameter is present on any of your URIs specifying a different transport (e.g. transport-UDP), it will be ignored. By default port 5061 will be used, however you may specify the port you wish to you in your Origination URI.

<table>
<thead>
<tr>
<th>Origination URI</th>
<th>Priority</th>
<th>Weight</th>
<th>Enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:31.220.67.40</td>
<td>10</td>
<td>10</td>
<td>✔️</td>
</tr>
</tbody>
</table>

Add an Origination URI

Disaster Recovery

Disaster Recovery URL  HTTP GET

In the case of a disaster, preventing your calls from being delivered to your Origination SIP URI above, you can configure a Disaster Recovery URL pointing to your application built on Twilio's powerful scripting tool called TwiML. You can use TwiML to build an application that will manage calls as required by your disaster recovery plan including replicating the functionality of your PBX (e.g. IVR).

Save  Cancel  Delete this Trunk
Ingate on Ravello

General  Termination  Origination  Numbers

Twilio Numbers

Below is a list of numbers associated with this Trunk.

<table>
<thead>
<tr>
<th>Number</th>
<th>Friendly Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>+14155884963</td>
<td>(415) 688-4963</td>
</tr>
</tbody>
</table>

Save  Cancel  Add a Number to this Trunk  Buy a Number  Delete this Trunk
Configuring the inGate SIParator

A lot of the configuration I am going to show here would be handled by the inGate setup tool so you wouldn't have to worry about it.

Basic configuration
The only things I did on this page were to give my SIParator a name and configure two DNS servers.

Network Configuration
I found this to be one of the least intuitive pieces of config on the SIParator - having to configure the networks around it. Thankfully, I found https://www.ingate.com/appnotes/Ingate_Connected_Networks.pdf, which made everything easier. Note that the 10.0.1.0/24 network isn't mentioned on this page.
With the LAN/DMZ configuration, the default gateway should be on the DMZ side.
And this is a summary of the interfaces configured on the SIPvator.
You need to enable the SIP module and specify the NAT'ed public IP address. Note that this NATed address is the same one used in the Origination URI on your Twilio SIP trunk.
SIP Trunking

This is where we tell the SIParator how to send calls to Twilio.

This is the first part of the trunking config where you set:
- Service Provider Domain: the FQDN or IP address to which the SIParator should send its SIP messages. This is the Termination URI that you configured when setting up your Twilio trunk.
- Restrict to calls from: only if a call comes in from the DMZ should we consider it as having originated from Twilio
- Host name in Request-URI of incoming calls: this is the NATed IP address of the SIParator. Twilio will put this address in the Request-URI of SIP messages that it sends to the SIParator.
- Route incoming based on: when Twilio sends a call to the SIParator, we will route it based on the contents of the Request-URI

This is the second part of configuring the trunk where you tell the SIParator how to find the Asterisk PBX and how to refer to users on that PBX.

- PBX Lines: My Twilio trunk is configured so that, if a call comes in to +1 (415) 688-4963, it will send the SIParator an INVITE for sip:+14156884963@31.220.67.40. I decided that the extension on my Asterisk PBX should be called 963 so I set up some translation rules here to reflect that.
- Setup for the PBX: here we tell the SIParator what the Asterisk PBX’s IP address is.
# SIP Routing

## Matching From Header

<table>
<thead>
<tr>
<th>Name</th>
<th>Use This</th>
<th>Or This</th>
<th>Transport</th>
<th>Network</th>
<th>Delete Row</th>
</tr>
</thead>
<tbody>
<tr>
<td>From PBX</td>
<td>*</td>
<td>*</td>
<td>Any</td>
<td>LAN2</td>
<td></td>
</tr>
<tr>
<td>From Trunk</td>
<td>*</td>
<td></td>
<td>Any</td>
<td>Trunk</td>
<td></td>
</tr>
</tbody>
</table>

## Matching Request-URI

<table>
<thead>
<tr>
<th>Name</th>
<th>Use This</th>
<th>Or This</th>
<th>Prefix</th>
<th>Head</th>
<th>Tail</th>
<th>Min. Tail</th>
<th>Max. Tail</th>
<th>Domain</th>
<th>Reg Exp</th>
<th>Delete Row</th>
</tr>
</thead>
<tbody>
<tr>
<td>e164 numbers</td>
<td>*</td>
<td>*</td>
<td>0.9</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Forward To

<table>
<thead>
<tr>
<th>Name</th>
<th>Use This</th>
<th>Or This</th>
<th>Replacement Domain</th>
<th>Transport</th>
<th>Reg Exp</th>
<th>Delete Row</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trunk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Dial Plan

### Methods in Dial Plan

The ACK, PRACK, CANCEL, BYE, UPDATE and INFO methods cannot be handled by the Dial Plan.

<table>
<thead>
<tr>
<th>Method</th>
<th>Delete Row</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVITE</td>
<td></td>
</tr>
<tr>
<td>OPTIONS</td>
<td></td>
</tr>
<tr>
<td>SUBSCRIBE</td>
<td></td>
</tr>
<tr>
<td>MESSAGE</td>
<td></td>
</tr>
<tr>
<td>NOTIFY</td>
<td></td>
</tr>
</tbody>
</table>

## ENUM Root

<table>
<thead>
<tr>
<th>Name</th>
<th>ENUM Root</th>
<th>Delete Row</th>
</tr>
</thead>
<tbody>
<tr>
<td>e164.org</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e164.org</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This is where you define how SIP calls should be routed. We want calls from the Asterisk PBX (on the CentOS VM) to be routed across a trunk to Twilio. The SIP trunk config we defined above already handles routing in the opposite direction - from Twilio to the PBX.

Setting up TLS and SRTP

If you want to use SIP over TLS and SRTP on your trunk, you will need to do the following.

Configure your Twilio trunk for Secure Trunking. This is at the bottom of the front page for your trunk.

inGate have a nice guide for configuring SSL and SRTP on the SIParator at https://www.ingate.com/appnotes/Ingate_Setting_Up_TLS-en.pdf.

I created a self signed server certificate (called “mycert” in the image below) on the SIParator because Twilio does not currently verify certificates for TLS connections and creating a self signed one was quicker and easier than getting one from a real CA.

You do have to upload Twilio’s whole certificate chain to the inGate so that it can accept inbound TLS connections. You do this in the “CA Certificates” section. “Twilio2” is the certificate that I uploaded correctly. I tried using just our root CA (“Twilio”) but this did not work.
Twilio's full chain is at the end of this document in Appendix 1.

You then need to configure the SIParator to use these certificates for TLS and to also allow SRTP.
- Changes have been made to the preliminary configuration, but have not been applied.

### SIP Transport
- [ ] TCP or UDP
- [ ] Any
- [ ] TLS

### TLS CA Certificates
- **CA**: Twilio2
- **Delete Row**: [ ]

Add new rows: 1 rows.

### Check Server Domain Match
- **Check if the server domain matches the certificate:**
  - [ ] Yes
  - [ ] No

### TLS Connections On Different IP Addresses
- **IP Address**: eth1 (10.0.1.3)
- **Own Certificate**: myert
- **Use CN FQDN**: Yes
- **Require Client Cert**: No
- **Accept Methods**: Any TLSv1.x
- **Delete Row**: [ ]

Add new rows: 1 rows.

### Making TLS Connections
- **Default own certificate**: -
- **Use methods**: Any TLSv1.x

Save    | Unde
Changes have been made to the preliminary configuration, but have not been applied.

### Media Encryption
- Enable media encryption
- Disable media encryption

### SIP Media Encryption Policy

<table>
<thead>
<tr>
<th>Media Via Interface/VLAN</th>
<th>Suite Requirements</th>
<th>Allow Transcoding</th>
<th>Delete Row</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet (eth1 untagged)</td>
<td>SRTP</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

Add new rows: 1 row.

### Default Encryption Policy

Suite requirements:
- Allow transcoding:
  - Yes
  - No

### Require TLS
- Require TLS for all cryptos but cleartext
- Do not require TLS

### RTP Profile
- Prefer RTP/SAVP (sdescriptions)
- Prefer RTP/AVP (cleartext and legacy encryptions)

### Crypto Suite Groups

<table>
<thead>
<tr>
<th>Name</th>
<th>Suite</th>
<th>Delete Row</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any (transcod</td>
<td>Sau pdesd (AES-128, SHA1 32)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SRTP pdesd (AES-128, SHA1 32)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SRTP pdesd (AES-128, SHA1 80)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SRTP pdesd (AES-CM 128, SHA1 32)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SRTP pdesd (AES-CM 128, SHA1 80)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SRTP pdesd (AES-CM 128, SHA1 32)</td>
<td></td>
</tr>
</tbody>
</table>

Add new rows: 1 group with 1 rows per group.
And then, finally, we update our trunk to use TLS and SRTP.

Appendix 1
Twilio’s certificate chain for TLS

-----BEGIN CERTIFICATE-----
MIIE7DCCA9SgAwIBAgIQTvJNraaWfHy0ukPRq3ngdzANBgkqhkiG9w0BAQsFADBbMQswCQYDVQQGEwJVUzEVMBMGA1UEChMMdGhhd3RlLCBJbmMuMRswGQYDVQQDExJ0aGF3dGUgU1NMIENBIC0gRzIwHhcNMTUwODI3MDAwMDAwWhcNMTYwODI2MjM1OTU5WjBtMQswCQYDVQQGEwJVUzETMBEGA1UECAwKQ2FsaWZvcm5pYTEWMBQGA1UEBwwTU2FuIEZyYW5jaXNjbzEVMBMGA1UECgwMVHdpbGljLCBJbmMuMRowGAYDVQQDDBEqLnBzdG4udHdpbGljLmNvbTCCASiwcGl0eC5jbi5qYXNzLnR3aWxpby5jb22CFSoucHN0bi51czEudHdpbGlvLmNvbYIUKi5zaXAudXMxLnR3

U2FulEzYyYW5jaXNjbzEVMBMGA1UECgwMVHdpbGljLCBJbmMuMRowGAYDVQQDDBEqLnBzdG4udHdpbGljLmNvbTCCASiwcGl0eC5jbi5qYXNzLnR3aWxpby5jb22CFSoucHN0bi51czEudHdpbGlvLmNvbY

-----END CERTIFICATE-----