SBC FAQ

What is the SBC capacity?

Sangoma provides two different tiers for its SBC. The **Vega Session Controller** and the **NetBorder Session Controller**. Both are based in the same software base, but Vega SBC is tailored to small densities (ie enterprise), from 1-250 concurrent calls. The NetBorder SBC it is aimed to big enterprises or ITSP/carriers, it goes all the way to 4,000 concurrent calls with hardware-assisted RTP/transcoding. In the near future we will at least double this capacity with more powerful DSPs and memory size.

The CPS (calls per second) measurement depends on many factors, including the hardware where you run it. Sangoma's SBC can run in standard Sangoma hardware appliances, custom hardware or even virtual machines. The carrier-level SBC appliance from Sangoma has been tested with 75 CPS with hardware transcoding involved.

What can it transcode?

Sangoma's NetBorder/Vega SBC does virtually all major codecs used in the industry, from narrow band (PCMU, G.729) to wide band codecs (ie G.722 and Siren/G.722.1 from Polycom)
The following is a list of supported codecs: G.711 (PCMU/PCMA), G.729, iLBC, G.722, G.722.1, GSM, G.723.1, G.726, AMR

The SBC is also capable of translating a variety of protocols, such as encrypted SIP TLS/SRTP traffic into non-encrypted UDP/TCP SIP traffic.

Where does it transcode?

Sangoma's NetBorder/Vega SBC is extremely flexible regarding transcoding. You can decide to do transcoding in hardware or software. You can also opt for bypassing media processing and allow the RTP flow directly between endpoints (this increases SBC overall capacity to handle more sessions easily). When doing hardware transcoding there is the option to do it built-in in the appliance DSPs, or with external DSPs (connected thru an ethernet network).

How does it ensure QoS (Quality of Service)?

There is several built-in mechanisms to protect QoS. You can specify CPU threshold limits to protect the quality of existing calls. Whenever the specified threshold is exceeded (ie, 80% CPU usage) the SBC will start refusing to accept new calls using a configurable SIP response code (ie 503 Service Unavailable), your equipment upstream can defer traffic to another SBC or gateway whenever it receives such code.

It is also possible to specify the ToS/DiffServ octet of SIP and RTP traffic to enforce QoS policies in the routing devices.

What kind of call routing does it do?

All the call routing is based on an XML scripting language, you can basically match SIP requests based on any field in the SIP packet (including source IP, SDP properties, codecs, headers etc) and route it to a defined SIP trunk/gateway or using the SBC built-in ENUM or LCR modules. You can also decide to reject the call or challenge the request yourself (this can also be done automatically by the SBC based on Call Admission Control rules).

How does it handle attacks?

There is multiple security mechanisms. The SBC comes with an IDS/IPS system (Intrusion Detection/Intrusion Prevention) system to block suspicious traffic. The definition of suspicious comes from a set of security rules/signatures of well-known VoIP attacks (there is rule sets for other protocols available as well, such as icmp, http).

You can also specify rules for failed SIP authentication requests (REGISTER or INVITE). If a given IP is sending you multiple failed authentication requests, it is either a misconfigured device or someone trying to perform a dictionary attack or scanning your network for valid users. NetBorder SBC can detect this patterns and block the offender immediately.

You can also detect malformed packets/traffic (someone trying sending garbage to see if it can crash your PBX or softw-switch), and the SBC can automatically block the offending IP address at the operating system level, where is extremely efficient to discard further packets from the offender.

How does it ensure reliability?

The SBC is capable of detecting SIP devices (ie gateways, proxies, soft-switches) that are down and re-route the traffic to alternate routes. This can be configured to be done automatically.

Does it provide ENUM support?

Yes, we support ENUM-based routing

Does it provide DTMF translation?

Yes we can translate from RFC2833 to inband

How well does it play with others?

We've done interoperability test with a number of PBX, phones and gateways. Some of them include:

PBX / SoftSwitches:
- Microsoft Lync
- OpenUC / sipXecs
- Metaswitch
- Asterisk
- FreeSWITCH

Phones:
- Bria
- AAstra
- Polycom
- Grandstream

ITSPs (SIP carriers):
- Appia Communications
- BroadVox
- CallCentric
- SoTel
- Vitality
- VoIP MS

The list will keep growing in the coming weeks. If you do not see your vendor included in the list, we'll make it work for you. Our SBC is extremely flexible and we're confident that with the right configuration we can interop with any vendor.

Is there any limits for SIP trunks?

The number of SIP trunks you can create is only limited by the amount of memory (RAM) and hard-drive space available. In realistic situations you won't hit the limit ever, we've tested with 200 SIP trunks without problems (or even start to scratch any limit). Licensing is done only based on active calls, not on any other SIP dialog or request.

Is there any limits for Virtual IPs?

The number of Virtual IPs is unlimited.

How is SIP header manipulation done?

All header manipulation is done at the same time the routing is done in the XML script. Special variables define the meaning of different headers and parameters within those headers. This is an example of the INVITE URI modification in a SIP Refer request:

```
<conditionfield="${sip_refer_to}" expression="(^.+)@some-company.com">
    <action application="export" data="sip_invite_from_uri=$1@company.com"/>
</condition>
```

Basically you match headers (ie ${sip_refer_to} is the variable where Refer-To header is populated by our SBC)when match against a desired regular expression,andthen replace either that same header or other headers by using the “export” or “set” application.

How can the RTP DTMF payload type be changed (ie from 96 to 101)?

You just set a variable during call routing before sending out the outgoing INVITE.

<action application="set" data="rfc2833_pt=96" />

If a SIP trunk is configured to use RFC2833 for DTMF but the remote end sends inband, can the SBC detect the tones?

Yes, the SBC can convert inband tones to RFC2833

Can your SBC can handle SIP and media/RTP on separate physical Ethernet lines/port.

Simple answerisYes.

Sangoma Carrier and Enterprise SBC perform RTP in hardware.
We can have two RTP operation modes in our SBC: Exposed or Hidden
Exposed mode
- Exposes RTP hardware IP addresses.
- RTP hardware communicates directly to remote agents via separate Ethernet port.

Hidden mode
- Hides RTP hardware IP address
- Single IP and Ethernetportis used for both Media and Signaling

Sangoma SBC also support VLAN's for both Signaling and Media/RTP.

What is the difference between calls and sessions?

This terminology may vary across vendors and even sometimes even within the same vendor some people may mistaken one for another. Be sure to clarify the meaning when comparing telecom equipment.

At Sangoma the terms "call" and "session" depend on the context. In our sales organization and to facilitate comparing our pricing to other SBCs, the term session and call are equivalent. This means that when talking to sales about your licensing needs, a session is a call with bi-directional audio and/or video.

In technical terms (when going through the WebUI or talking to tech support or Sangoma engineers) Sangoma SBC's use the term "session" to refer to a "call leg". Your typical call in a Sangoma SBC will require 2 sessions (in technical terms). An inbound session and an outbound session. As a rule of thumb you can say a given call is composed of 2 sessions, however, in some circumstances, for example call forking, sometimes Sangoma SBC may actually have 3 or more sessions at the same time for the same call (one inbound session created multiple outbound sessions), until one of them receives early media or is answered and then the call session count is reduced to 2 (the inbound and only one outbound, the other outbound sessions are cancelled once one of the outbound sessions is confirmed).

If you acquire from our Sales organization an SBC with support for 250 sessions, you're getting an SBC with support for 250 sessions or calls (session and call is the same in this context). However, when you navigate through the WebUI (for example in the "Sessions" page) you will see 2 sessions per call (inbound/outbound legs) but you will be able to see up to 500 of these (twice as much as you have licensed).

How Does Call Forking work with the SBC?

With SIP Forking you receive one call, and the SBC as a result, forks into multiple calls (2 or more). Once one of theforkedcallsanswersthefirst one in answering gets bridged, the other ones get cancelled.

How to return multiple values from curl?

The response from curl is always stored in the variable $ {curl_response_data}, but you can return multiple values simply separate the values by commas, or any other character you want.

If you separated the values by commas the HTTP response would be: "value-1, value-2, value-3"

After you execute the curl application you have to transfer to a new extension. The transfer is necessary to make the variable $ {curl_response_data} available to be evaluated in a new condition.

In theexamplebelow the"exec_curl" extension is simply just running the curl app. The second extension "parse_curl_response" is cutting individual values separated by commas from the string.

```
<extension name="exec_curl">
<condition>
  <action application="curl" data="http://X.X.X.X:8282/roting.php?number=${destination_number}&amp;callerid=${caller_id_number}"/>
    <action application="set" inline="true" data="auto_hunt=true"/>
    <action application="transfer" data="parse_curl_response"/>
  </condition>
</extension>



<extension name="parse_curl_response">
  <condition field="${curl_response_data}" expression="^(\w+),?(\w+)?,?(\w+)?$">
    <action application="log" data="crit value #1: $1"/>
    <action application="log" data="crit value #2: $2"/>
    <action application="log" data="crit value #3: $3"/>
  </condition>
</extension>
```

Now can see the values are stored in the variables $ 1, $ 2, $ 3 and you can use them as you like in your dial plan.

Note as well how you cannot use the ampersand character in a curl request unless you escape it (because it's XML) using HTML entities (&)

How can I replace my SSD drives in the even to a failure?

1) Power off the Box
2) Replace the FIRST SSD
3) Power on the Box
4) Navigate to WEBUI and run RAID1 synchronization
5) Once synchronized, Power off the Box again
6) Replace the SECOND SSD
7) Power on the Box again
8) Navigate to WEBUI and run RAID1 synchronization again
9) Obtain the serial from the newly installed SSDs
10) Send both serial and current license file to support (http://support.sangoma.com/) so support can generate a new license with the new serial.

How can I manipulate a SIP header value?

The 2 requests that have better support for header manipulation are INVITE and REFER. Other SIP messages (such as responses) have also some header manipulation support but is more limited. In this FAQ we'll cover a regular INVITE request, but the logic applies to REFER messages as well.

Manipulation is a 2-step process.

First youmatchthe header you want to manipulate against a regular expression (more info here). The regular expression might simply test for the header to be present in the SIP request (INVITE or REFER for example), or it may check that the header has a particular format. The matching procedure can be also used to capture certain parts of the header value to be re-used later in the final result you want.

The second step is rewriting the header with the new value, most of the time composing it using some of the original header information.

In this example, we'll manipulate the "Diversion" header to change the host part and replace it for a different host value. Imagine you have a Diversion header that comes like this:

Diversion: "User Name" <sip:123456789@server1.domain.com:5060>;reason=deflection

And you want to change the host part of the header to public-server.domain.com, so the final result should look like:

Diversion: "User Name" <sip:123456789@public-server.domain.com:5060>;reason=deflection

First, you want to create a condition to determine if the SIP request contains a Diversion header at all, and capture all the characters before the @ symbol, then capture the original host name, then the rest of the header value.

```
<condition field="${sip_h_Diversion}" expression="^(.+)@(.+):(.+)$" break="never">
```

Here we match the inbound SIP Diversion header (which is presented by the SBC variable sip_h_Diversion) and capture using parenthesis () 3 different values. Everything from the start of the header value until before the @ symbol, then everything after the @symbol but before the colon (before the port number) and then everything after the colon until the end of the header value.

Inside the condition now we can use those 3 captured values as the special variables $1, $2, and $3 respectively for every string inside the set of parenthesis. Inside the condition, we can export a new sip_h_Diversion variable with the modified values.

```
<action application="export" data="sip_h_Diversion=$1@public-server.domain.com:$3" />
```

Here we have replaced the contents of the host (which was captured in variable $2) and only used $1 and $3 and inserted the desired hostname. The SBC will generate the new outbound SIP request with the new Diversion header value.

Can Virtual Machines Introduce Jitter?

Yes a virtual machine can introduce jitter into the audio stream. At the same rate other factors such as the network are the typical causes of jitter.

The reason for this is virtualization software is typically not optimized for the real time needs of voice over IP. As well the virtualization software introduces an abstraction layer on top of the hardware. This means the hardware is being emulated to your guest OS. Your guest OS does not have direct access to the real hardware it is running on. It always has to go through the abstraction layer to use the hard disk/CPU/memory and other resources.

The best defense against this is to ensure there is enough resources allocated to each VM. In addition to this ensure you never over allocate any resources on the host OS. As well it is always a good idea to keep your virtualization software up to date.