Configuration Note

AudioCodes Mediant™ Software Session Border Controller (SBC) Series

Automatic Provisioning via Cloud-Init

Mediant Virtual Edition (VE) SBC

Mediant Cloud Edition (CE) SBC

Version 7.2

audiocodes

Table of Contents

This page is intentionally left blank.

Notice

Information contained in this document is believed to be accurate and reliable at the time of printing. However, due to ongoing product improvements and revisions, AudioCodes cannot guarantee accuracy of printed material after the Date Published nor can it accept responsibility for errors or omissions. Updates to this document can be downloaded from https://www.audiocodes.com/library/technical-documents.

This document is subject to change without notice.

Date Published: September-06-2018

WEEE EU Directive

Pursuant to the WEEE EU Directive, electronic and electrical waste must not be disposed of with unsorted waste. Please contact your local recycling authority for disposal of this product.

Customer Support

Customer technical support and services are provided by AudioCodes or by an authorized AudioCodes Service Partner. For more information on how to buy technical support for AudioCodes products and for contact information, please visit our website at https://www.audiocodes.com/services-support/maintenance-and-support.

Abbreviations and Terminology

Each abbreviation, unless widely used, is spelled out in full when first used.

Document Revision Record

LTRT	Description
28660	Initial document release for Version 7.2.
28661	Minor formatting changes.
28662	#write-factory hashtag added; formatting (corporate logos and URLs)

Documentation Feedback

AudioCodes continually strives to produce high quality documentation. If you have any comments (suggestions or errors) regarding this document, please fill out the Documentation Feedback form on our website at https://online.audiocodes.com/documentation-feedback.

This page is intentionally left blank.

1 Introduction

This document describes how to automatically provision AudioCodes Mediant Virtual Edition (VE) and Mediant Cloud Edition (CE) Session Border Controllers (SBCs) that are deployed in a private / public cloud environment.

1.1 Supported Environments

The SBC provides native support for automatic provisioning in the following cloud environments:

- OpenStack
- Amazon EC2
- Azure

Other environments (e.g. VMware or KVM) are supported via "config-drive emulation" – as described in Chapter 6.

1.2 Boot Sequence

The SBC performs the following actions upon the first boot:

- Attempts to acquire an IP address on each available network interface via the DHCP server.
- Identifies the environment that it runs on (e.g. OpenStack, Amazon, VMware etc)
- Checks for presence of local config-drive and attempts to read "automatic configuration" from it.
- Checks for presence of cloud meta-data service (http://169.254.169.254) and attempts to read from it "automatic configuration" data and SSH public key.
- Processes acquired "automatic configuration" data.
- If one of the supported cloud environments is detected (as described in Section 1.1) memorizes network configuration acquired via the DHCP server.
- Stores configuration (ini file) and all other elements from "automatic configuration" data in persistent storage.
- Continues normal boot sequence.

Consequent reboots do not execute the above sequence except in the following cases:

- The SBC is restored to the "factory defaults" configuration (via "write factory" CLI command).
- New SBC instances are created from the snapshot of another SBC instance.

This page is intentionally left blank

2 “Automatic Configuration” Data

The SBC uses customized cloud-init implementation to acquire “automatic configuration” data. The data may be provided in one of the following ways:

- User-data on cloud meta-data service (http://169.254.169.254)
- User-data on local config-drive

When cloud meta-data service is used, the following conditions should be fulfilled:

- The SBC is able to acquire an IP address via DHCP on at least one network interface.
- Meta-data service is accessible via the interfaces on which IP addresses were acquired (via either implicit or explicit routing rules).

“Automatic configuration” data should be formatted as follows:

```
#hashtag1
<data1>
#hashtag2
<data2>
```

The following hashtags are supported:

- #ini-file – configuration file
- #ini-incremental – incremental configuration file
- #ini-default – default configuration file (AKA “client defaults”)
- #customer-id, #license-key – customer ID and customer-specific license key
- #dhcp-address. #no-dhcp-address – to override default behavior concerning IP addresses acquired via DHCP
- #network-if – to customize configuration of network interfaces
- #static-route – to customize configuration of static routes
- #cloud-end – indicates the end of “automatic configuration” data

2.1 #ini-file

#ini-file hashtag is used to specify the SBC configuration (in ini file format):

```
#ini-file
 [SYSTEM Params]
SyslogServerIP = 10.8.12.50
EnableSyslog = 1
TelnetServerEnable = 0
SSHServerEnable = 1
```

If the provided configuration includes InterfaceTable, it is applied “as is”, essentially overriding any network configuration that was acquired via the DHCP. In such case, it is mandatory to have the exact match between the actual SBC instance configuration and the network configuration in ini file. For example, if there are three network interfaces, the ini file should have three physical ports and typically three Ethernet groups, devices/VLANs and interfaces.

If the provided configuration does not include an InterfaceTable, the SBC uses IP addresses acquired via the DHCP to automatically populate an Interface table with the relevant information as follows:

- First interface is assumed to be of type “OAM + Media + Control” and is named “O+M+C”
- Additional interfaces are assumed to be of type “Media + Control” and are named “if 2”, “if 3” etc.
- If the ini file contains HARemoteAddress parameter, the last interface is assumed to be of type “Maintenance” and is named “HA”
- Any interface that fails to acquire an IP address via DHCP is assigned with a temporary IP address – “192.168.<i>.100”

Note: If the provided configuration does not include InterfaceTable, it is recommended to also remove from it the following tables: PhysicalPortsTable, EtherGroupTable, DeviceTable; as they will anyway be implicitly removed and generated based on the actual interfaces connected to the SBC instance.

2.2 #ini-url

#ini-url hashtag is used to specify the location on the external file server where the SBC configuration (ini file format) is stored. The following protocols are supported: HTTP, FTP and SFTP.

The syntax is as follows:

```
#ini-url
http://10.4.220.50/sbc/config.ini
```

2.3 #ini-s3

#s3-url hashtag is used to specify location on Amazon S3 cloud storage where the SBC configuration (ini file format) is stored. This is very useful for the Amazon EC2 environment that limits the size of user-data block to 16 KB and therefore makes it impossible to include large ini files via the #ini-file hashtag.

The syntax is as follows:

```
#ini-s3
region us-west-2
bucket ac-a1
file sbc.ini
```

Where:

- <region> – specifies name of the region, e.g. us-west-2
- <bucket> – specifies name of the S3 bucket
- <file> – specifies the file name; if file is located inside directory use the full path instead – e.g. “`file dir/sbc.ini`”

You must create a proper IAM role and assign it to the SBC instance to allow access to the S3 bucket. For example, use the following IAM role and policy to enable access to ini files stored in “ac-a1” bucket.

IAM > Roles > SbcS3Access

Permissions:

Policy Name: AccessINIBucket

IAM > Policies > AccessINIBucket

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": "arn:aws:s3:::ac-a1"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:DeleteObject"
            ],
            "Resource": "arn:aws:s3:::ac-a1/*"
        }
    ]
}
```

2.4 #network-if

It is possible to complement/override automatic IP address assignment (via DHCP) as described above by using #network-if hashtag. The syntax is as follows:

```
#network-if
<idx> <name> <app> <ip> <prefix> <default gateway> <dns> <mtu>
```

Where:

- <idx> - network interface index, starting from 0
- <name> - interface name (without spaces)
- <app> - interface application type, as per InterfaceTable_ApplicationTypes;

  The most common values are as follows:
  - 6 – O+M+C
  - 5 – M+C
  - 99 – Maintenance (HA)
- <ip> - IPv4 address
- <prefix> - prefix length
- <default gateway> - default gateway or “0.0.0.0” if not defined
- <dns> - DNS server or “0.0.0.0” if not defined
- <mtu> - maximum transmission unit (MTU) size

Every line in #network-if section should have EXACTLY eight tokens.

Any value except for <idx> may be omitted by using "-" (dash) instead of it. Only values that differ from "-" (dash) will be applied on top of the configuration acquired via the DHCP server.

It is also perfectly valid to specify only SOME indexes in #network-if section – thus overriding / complementing configuration of specific interfaces only.

For example:

- To specify an alternative name of the 1st network interface (instead of default “O+M+C”) use the following syntax:

  ```
  #network-if
  0 LAN - - - - - -
  ```
- To change the network types of interfaces for the SBC instance that has three network interfaces; however the second interface (and not 3rd) should be used as maintenance, use the following syntax:

  ```
  #network-if
  0 - 6 - - - - -
  1 - 99 - - - - -
  2 - 5 - - - - -
  ```
- To specify the IP address for the second interface (that is connected to the network which lacks a DHCP service) use the following syntax:

  ```
  #network-if
  1 - - 10.4.2.50 16 10.4.0.1 - -
  ```

- To define names and types for three interfaces and fully specify the IP configuration for the second interface (that is connected to the network that lacks a DHCP service) use the following syntax:

```
#network-if
0 LAN 6 - - - - -
1 HA 99 10.4.2.50 16 10.4.0.1 0.0.0.0 -
2 WAN 5 - - - - -
```

When #network-if is used to specify types of specific network interfaces, the implicit network type configuration behaves as follows:

- If the OAM interface is not explicitly specified by the user, the first interface with an unspecified type or of type M+C / C / M is assumed to handle OAM traffic (and it’s type is changed accordingly to O+M+C, O+C or O+M).
- If HARemoteAddress parameter is present in the configuration data (ini file) and the maintenance interface is not explicitly specified by the user, the last interface that doesn’t handle OAM traffic is assumed to be of type ‘Maintenance/HA’. This implies that if, for example, you only have two interfaces and specify that the second interface handles OAM traffic, the first interface will become the Maintenance/HA. Of course you may explicitly specify types of all interfaces – and not rely on implicit logic; however, it’s not mandatory.

Note: This hashtag is not applicable for the Amazon EC2 environment.

2.5 #dhcp-address

The SBC determines whether it’s deployed in a cloud environment or not by checking the availability of the cloud meta-data service (http://169.254.169.254). If the service is not available, it assumes that it’s deployed in a “pure virtual” (non-cloud) environment and does not memorize the network configuration acquired via the DHCP server.

The above described behavior may be overridden by specifying #dhcp-address hashtag in the “automatic configuration” data. The hashtag has no “body” and forces the SBC to memorize the network configuration acquired via the DHCP server regardless of the cloud meta-data service availability:

```
#dhcp-address
```

For example, you may use this hashtag to force the SBC to memorize network configuration acquired via the DHCP server in environments that lack the cloud meta-data service and use config-drive instead.

2.6 #no-dhcp-address

#no-dhcp-address hashtag may be used to restore SBC snapshots in the cloud environment, while preserving network configuration as specified in the snapshot. Without this option, the SBC will acquire IP addresses via the DHCP server and will use these new addresses instead of the IP addresses specified in the snapshot.

2.7 #static-route

#static-route hashtag may be used to append custom static routes to the configuration acquired via the DHCP server. The syntax is as follows:

```
#static-route
<idx> <ip> <prefix> <via>
```

Where:

- <idx> - network interface index, starting from 0
- <ip> - destination IP address
- <prefix> - destination prefix length
- <via> - IP address of the gateway/router

For example:

- To add static route to 10.3.0.0 subnet on 1st interface use the following:

```
#static-route
0 10.3.0.0 16 10.4.0.1
```

2.8 #customer-id and #license-key

Customers who purchased customer-specific bulk licenses from AudioCodes should use #customer-id and #license-key hashtags to provision the correct SBC license. The syntax is as follows:

```
#customer-id
0123456789
#license-key
okRTr5topwYRa4Nu6xkiu6Z3nAxzcOlc80N...
```

Make sure that you specify both #customer-id and #license-key hashtags in the “automatic configuration” data; otherwise the license will not be properly applied.

2.9 #ini-default

#ini-default hashtag specifies configuration data (using ini file syntax) that is stored in separate persistent storage (AKA client-defaults) and is not removed when new configuration data (e.g. ini file) is loaded on the SBC. In essence, values provided in this block of configuration data become new “default values” for corresponding parameters.

2.10 #ini-incremental

#ini-incremental hashtag is very similar to the #ini-file hashtag; however, configuration data specified in it is applied “on top” of the existing SBC configuration instead of overriding it. SBC images (QCOW2, AMI) published by AudioCodes do not contain configuration data – therefore this tag is not really useful. However customers may create snapshots of the SBC – with some pre-defined configuration – and use them to create new SBC instances. In such cases, #ini-incremental hashtag may be used to adjust the image configuration instead of specifying a new configuration “from scratch”.

2.11 #cloud-end

Any hashtag that starts with #cloud- (for example, #cloud-end) indicates the end of “automatic configuration” data. It may be needed for cloud / orchestrator implementations that inject custom Linux shell code into user-data. In such cases, the #cloud-end hashtag effectively separates between a meaningful configuration provided by the user and an automatic configuration injected by the orchestrator and is irrelevant to the SBC.

2.12 #write-factory

The #write-factory hashtag returns the SBC to its factory defaults, erasing all existing configuration, including current network configuration and the local users table. It may be used to regain access to the SBC if the administrator forgets the login credentials or is unable to access it (for whatever reason).

This page is intentionally left blank.

3 SSH Public Key

The SBC deployed in the cloud environment extracts the SSH public key from the cloud meta-data service and configures it as a means for management for user authentication (e.g. for user Admin). In addition to this, it automatically enables the SSH protocol for connectivity to the CLI management interface.

This page is intentionally left blank.

4 Network Configuration in Amazon EC2 Environment

The SBC deployed in the Amazon EC2 environment supports multiple network interfaces (ENIs) and both primary and secondary IP addresses. Primary IP addresses are acquired via the DHCP server and are assigned in the same way as in the OpenStack environment. Secondary IP addresses are acquired from the EC2 meta-data service and assigned after primary addresses using “if X_Y” naming convention (e.g. “if 1_1”). Note however that #network-if hashtag may be used to modify the configuration of primary IP addresses only.

In addition to the above, the SBC deployed in the Amazon EC2 environment automatically discovers its public IP address(es) and configures it/them accordingly in the NATTranslation table – to enable the proper NAT address translation.

This page is intentionally left blank.

5 Automatic Instance Provisioning

Take into consideration that the cloud-init auto-configuration sequence occurs on the first SBC boot only. This is not a problem for a typical “automatic provisioning” scenario, where the instance configuration is auto-generated and configured via the #ini-file hashtag in user-data and all network configuration and IP address assignment is completed as part of the instance creation, before the instance is started.

However, if you create an instance manually – e.g. via Amazon EC2 GUI – you may decide to complete and/or modify the SBC networking configuration after the instance is already running. In such cases, consider using the“write factory” CLI command that deletes the current SBC configuration, reboots the instance and forces the cloud-init auto-configuration process to re-run on the following reboot.

This page is intentionally left blank.

6 Config-drive Emulation

Pure virtualization (non-cloud) environments – e.g. VMware or KVM – may use automatic provisioning as described in Chapter 5 by emulating the config-drive method of the automatic provisioning.

Config-drive is essentially a virtual CD-ROM attached to the running the SBC instance that has a single FAT or ISO9660 filesystem with the label `config-2` and the configuration data is located at `openstack/latest/user_data`.

Therefore it can be easily created by executing the following commands on a Linux machine:

```
mkdir -p /tmp/new-drive/openstack/latest
cp user_data /tmp/new-drive/openstack/latest/user_data
mkisofs -R -V config-2 -o configdrive.iso /tmp/new-driv
rm -r /tmp/new-drive
```

`configdrive.iso` is created as described above and should be attached to the SBC instance as a virtual CD-ROM device.

Consider using #dhcp-address hashtag, as described in Section 2.52.5, to make the SBC use the network configuration acquired via the DHCP server.

This page is intentionally left blank.

7 HEAT Orchestration Templates

Automatic configuration may be specified in HEAT orchestration templates to automate provisioning complex services that include SBC instances.

For example, the following HEAT templates may be used to create and automatically provision two SBC instances that operate in 1+1 HA mode. Note that to create such a configuration, maintenance IP addresses of each instance must be configured as HARemoteAddress in another instance and “allowed_address_pairs” configuration on the second network port must be modified to support IP failover.

```
resources:
  server_group:
    type: OS::Nova::ServerGroup
    properties:
      name: sbc_pair
      policies:
        - anti-affinity

  server1:
    type: OS::Nova::Server
    properties:
      name: sbc1
      image: { get_param: image }
      flavor: { get_param: flavor }
      key_name: { get_param: key_name }
      networks:
        - port: { get_resource: server1_port1 }
        - port: { get_resource: server1_port2 }
      scheduler_hints: { group: { get_resource: server_group } }
      user_data:
        str_replace:
          template: |
            #ini-file
            HARemoteAddress = $ip
            HAPriority = 10
            HAUnitIdName = sbc1

          params:
            $ip: { get_attr: [server2_port2, fixed_ips, 0,
ip_address] }

  server1_port1:
    type: OS::Neutron::Port
    properties:
      network: { get_param: public_net }
      fixed_ips:
        - subnet: { get_param: public_subnet }
      security_groups: [{ get_resource: server_security_group }]

  server1_port2:
    type: OS::Neutron::Port
    properties:
      network: { get_resource: private_net }
      fixed_ips:
        - subnet: { get_resource: private_subnet }

  server2:
    type: OS::Nova::Server
    properties:
      name: sbc2
      image: { get_param: image }
```

```
      flavor: { get_param: flavor }
      key_name: { get_param: key_name }
      networks:
        - port: { get_resource: server2_port1 }
        - port: { get_resource: server2_port2 }
      scheduler_hints: { group: { get_resource: server_group } }
      user_data:
        str_replace:
          template: |
            #ini-file
            HARemoteAddress = $ip
            HAPriority = 5
            HAUnitIdName = sbc2

          params:
            $ip: { get_attr: [server1_port2, fixed_ips, 0,
ip_address] }
    depends_on: server1

  server2_port1:
    type: OS::Neutron::Port
    properties:
      network: { get_param: public_net }
      fixed_ips:
        - subnet: { get_param: public_subnet }
      security_groups: [{ get_resource: server_security_group }]
      allowed_address_pairs:
        - ip_address: { get_attr: [server1_port1, fixed_ips, 0,
ip_address] }

  server2_port2:
    type: OS::Neutron::Port
    properties:
      network: { get_resource: private_net }
      fixed_ips:
        - subnet: { get_resource: private_subnet }
```

International Headquarters
1 Hayarden Street,
Airport City
Lod 7019900, Israel
Tel: +972-3-976-4000
Fax: +972-3-976-4040

AudioCodes Inc.
27 World's Fair Drive,
Somerset, NJ 08873
Tel: +1-732-469-0880
Fax: +1-732-469-2298

Contact us: https://www.audiocodes.com/corporate/offices-worldwide
Website: https://www.audiocodes.com/

Document #: LTRT-28662