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 7.4

Table of Contents

This page is intentionally left blank.

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
28663	Initial document release for Version 7.4.
28664	AWS, Google Cloud, AWS-compatible added; new sections (Network Configuration. Auxiliary Files, TLS Certificates); new hashtags (#network-interfaces, #reset-password, #cloud-init, #write-factory); miscellaneous

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.

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:

- Amazon Web Services (AWS)
- Azure
- Google Cloud
- OpenStack
- Other clouds that use AWS-compatible metadata service (e.g., Yandex cloud)

Other environments (e.g., VMware or KVM) are supported via "configuration drive emulation", as described in Section 3, Configuration Drive Emulation.

1.2 Boot Sequence

The SBC performs the following actions upon the *first* boot:

- Checks for the presence of local "configuration drive" and attempts to read "automatic configuration" from it.
- Attempts to acquire an IP address on each available network interface from the cloud infrastructure (through DHCP).
- Checks for the presence of cloud metadata service (http://169.254.169.254), identifies the environment in which it's deployed (e.g., OpenStack, AWS, or Azure), and attempts to read "automatic configuration" from it.
- If one of the supported cloud environments is detected (as described in Section 1.1, Supported Environments) or the **#dhcp-address** hashtag is found in "automatic configuration" data, it stores the acquired network configuration in persistent storage.
- Stores configuration (**#ini-file** hashtag) and all other elements from the acquired "automatic configuration" data in persistent storage.
- Continues normal boot sequence.

Upon subsequent reboots, SBC performs a different sequence:

- Attempts to acquire an IP address on the first network interface (eth0) from the cloud infrastructure (through DHCP).
- Checks for the presence of cloud metadata service (http://169.254.169.254), identifies the environment in which it's deployed (e.g., OpenStack, AWS, or Azure), and attempts to read "automatic configuration" from it.
- Processes one of the following special hashtags *only*:
  - **#reset-password**
  - **#write-factory**
  - **#cloud-init**
- Continues normal boot sequence.

The first boot sequence is repeated in the following cases:

- The SBC is restored to "factory default" configuration (using the `write factory` CLI command).
- A new SBC instance is created from the snapshot of another SBC instance.
- **#cloud-init** or **#write-factory** hashtags are specified in the "automatic configuration" provided by cloud metadata service.

1.3 Network Configuration

Automatic provisioning uses IP addresses acquired through DHCP to automatically populate an Interface table with the relevant information, as follows:

- Interfaces are named "eth0", "eth1", and so on.
- The first interface (eth0) is assigned with application type "OAM + Media + Control" (6).
- Other interfaces are assigned with application type "Media + Control" (5).
- If the interface has additional (secondary) IP addresses, the Interfaces table is populated with additional entries, named "ethX:0", "ethX:1", and so on.
- If the INI file contains the **HARemoteAddress** parameter, the last interface is assigned with application type "Maintenance" (99).
- Any interface that fails to acquire an IP address through DHCP is assigned with a "dummy" IP address ("192.168.<i>.100").

Network configuration can be customized or extended using the **#network-interfaces** hashtag. Refer to Section 0,

#network-interfaces for more information.

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 metadata service (http://169.254.169.254)
- Local configuration drive

When cloud metadata service is used, the following conditions should be fulfilled:

- The SBC is able to acquire an IP address through DHCP on the first network interface (eth0).
- Cloud metadata service is accessible through the first network interface (eth0).

"Automatic configuration" data should be formatted as follows:

```
#hashtag1
<data1>
#hashtag2
<data2>
#cloud-end
```

Most cloud environments impose a limit on the maximum size of user-data that may be provided through metadata service. For example, in the AWS environment, user-data is limited to 16 KB. Therefore, if you need to provide extensive SBC configuration, it's recommended to host configuration files on an external HTTP/HTTPS/FTP/SFTP/S3 storage and use URL hashtags (e.g., #ini-url or #ini-s3) to reference these files.

2.1 #ini-file

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

```
#ini-file
SyslogServerIP = 10.8.12.50
EnableSyslog = 1
TelnetServerEnable = 0
SSHServerEnable = 1
```

It's recommended to exclude the following configuration tables from the provided INI file:

- InterfaceTable
- EtherGroupTable
- DeviceTable
- PhysicalPortsTable

This ensures that the SBC uses networking configuration provided by the cloud through DHCP, as described in Section 1.3, Network Configuration.

If you still choose to include InterfaceTable in your configuration file, the table will be applied "as is", essentially overriding any network configuration that was acquired through DHCP. In this case, it's 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.

2.2 #ini-url

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

The syntax is as follows:

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

2.3 #ini-s3

The **#ini-s3** hashtag is used to specify the location on Amazon S3 cloud storage where the SBC configuration (in INI file format) is stored.

The syntax is as follows:

```
#ini-s3
https://sbc.s3.eu-central-1.amazonaws.com/config.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 the `"sbc"` bucket.

IAM > Roles > SbcS3Access:

```
Permissions policies: AccessINIBucket
```

IAM > Policies > AccessINIBucket:

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

2.4 Auxiliary Files

The following hashtags can be used to specify the location on the remote file server (HTTP, HTTPS, FTP or SFTP) where Auxiliary files are stored:

- **#cpt-file-url** – Call Progress Tones file
- **#dial-plan-csv-url** – Dial Plan file
- **#prt-file-url** – Pre-Recorded Tones file
- **#user-info-csv-url** – User Info file

The syntax is as follows:

```
#cpt-file-url
http://10.4.220.50/sbc/CPT.dat

#dial-plan-csv-url
http://10.4.220.50/sbc/DIALPLANRULES.csv

#prt-file-url
http://10.4.220.50/sbc/PRT.dat

#user-info-csv-url
http://10.4.220.50/sbc/USERINFO.csv
```

If Auxiliary files are hosted on Amazon S3 cloud storage, use the following hashtags instead:

- **#cpt-file-s3**
- **#dial-plan-csv-s3**
- **#prt-file-s3**
- **#user-info-csv-s3**

For example:

```
#dial-plan-csv-s3
https://sbc.s3.eu-central-1.amazonaws.com/DIALPLANRULES.csv
```

2.5 TLS Certificates

The following hashtags can be used to specify the location on the remote file server (HTTP, HTTPS, FTP or SFTP) where TLS certificates and/or private keys are stored:

- **#trusted-root-url** – trusted root certificates
- **#certificate-url** – device certificate
- **#private-key-url** – device's private key

The syntax is as follows:

```
#trusted-root-url <context>
http://10.4.220.50/sbc/root.pem

#certificate-url <context>
http://10.4.220.50/sbc/cert.pem

#private-key-url <context> [<password>]
http://10.4.220.50/sbc/key.pem
```

All TLS hashtags should be followed by the TLS Context ID number to which they apply. For example, to specify the trusted root certificate for TLS Context #0:

```
#trusted-root-url 0
http://10.4.220.50/sbc/root.pem
```

All TLS Contexts, except for the default TLS Context #0, must be explicitly created using the TLSContexts table in the configuration file provided through the **#ini-file**, **#ini-url** or **#ini-s3** hashtags. For example:

```
[ TLSContexts ]
FORMAT Index = Name, TLSVersion, ServerCipherString,
ClientCipherString, DHKeySize;
TLSContexts 1 = "BTALK", 4, "DHE-RSA-AES256-GCM-SHA384", "DHE-RSA-
AES256-GCM-SHA384", 2048;
[ \TLSContexts ]
```

The private key hashtag may also optionally be followed by a password string:

```
#private-key-url 0 mypassword
http://10.4.220.50/sbc/key.pem
```

All TLS files should be in PEM format.

If TLS files are hosted on Amazon S3 cloud storage, use the following hashtags instead:

- **#trusted-root-s3**
- **#certificate-s3**
- **#private-key-s3**

For example:

```
#trusted-root-s3 0
https://sbc.s3.eu-central-1.amazonaws.com/root.pem
```

2.6 #network-if

Note: Starting with Version 7.20A.254.202, the **#network-if** hashtag is deprecated; use the **#network-interfaces** hashtag instead.

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

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

Where:

- <idx> is the network interface index, starting from 0
- <name> is the interface name (without spaces)
- <app> is the 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> is the IPv4 address
- <prefix> is the prefix length
- <default gateway> is the default gateway or "0.0.0.0" if not defined
- <dns> is the DNS server or "0.0.0.0" if not defined
- <mtu> is the maximum transmission unit (MTU) size

Every line in the **#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.

Notes:

- **#network-if** hashtag can be used for primary IP addresses only.
- **#network-if** hashtag is not applicable for the AWS environment.

2.7 #network-interfaces

Notes:

- **#network-interfaces** hashtag is available in Version 7.20A.254.202 or later.
- Support for AWS environment is available in Version 7.40A.260.001 or later.

The **#network-interfaces** hashtag is used to complement and/or override automatic IP address assignment (through DHCP / cloud metadata) performed on the first boot. The syntax is as follows:

```
#network-interfaces
iface <name>
    <parameter> <value>
```

Where:

- <name> is the network interface name (e.g., "eth0" or "eth1:1")
- <parameter> is the parameter name (as described below)
- <value> is the parameter value

Supported parameters:

- name – interface name
- ip – IP address
- prefix – subnet prefix length
- gateway – default gateway
- mode – interface mode:
    - 3 – IPv6 manual prefix
    - 4 – IPv6 manual
    - 10 – IPv4 manual (default)
- app – application type:
    - 0 – OAM
    - 1 – Media
    - 2 – Control
    - 3 – OAM + Media
    - 4 – OAM + Control
    - 5 – Media + Control
    - 6 – OAM + Media + Control
    - 23 – Cluster
    - 99 – Maintenance (HA)
- dns – primary DNS server
- dns2 – secondary DNS server
- mtu – maximum transmission unit (MTU) size

The **#network-interfaces** section may contain configuration for multiple interfaces. For example:

```
#network-interfaces
iface eth0
    ip 10.2.0.65
    prefix 24
    gateway 10.2.0.1
iface eth1
    ip 10.3.0.12
    prefix 24
    gateway 10.3.0.1
```

Configuration is applied on top of the automatic IP address assignment (through DHCP / cloud metadata) performed on the first boot. Therefore, the following use cases are supported:

- Providing complete configuration for environments that lack DHCP agent or cloud metadata, as shown above.

- Updating configuration of the interfaces discovered by automatic IP address assignment, for example, changing their DNS server configuration:

```
#network-interfaces
iface eth0
    dns 8.8.8.8
```

- Providing configuration for secondary IP addresses. Note that in this case, it's enough to specify the IP address only; the rest of the configuration is automatically copied from the corresponding primary IP address. For example:

```
#network-interfaces
iface eth0:1
    ip 10.2.0.112
```

Note: The **#network-interfaces** hashtag is not applicable for the AWS environment in versions prior to 7.40A.260.001.

2.8 #dhcp-address

The SBC determines whether it’s deployed in a cloud environment or not by checking the availability of the cloud metadata service (typically located at 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 through DHCP.

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

```
#dhcp-address
```

Example of use cases:

- SBC is deployed in the VMware environment that has a DHCP server, but lacks metadata service. Configuration drive with the **#dhcp-address** hashtag may be used to memorize network acquired through DHCP on the first boot.
- SBC is deployed in the Hyper-V environment that lacks both DHCP and metadata services. Configuration drive with the **#network-interfaces** and **#dhcp-address** hashtags may be used to provide initial networking configuration for the SBC instance.

2.9 #no-dhcp-address

The **#no-dhcp-address** hashtag prevents the SBC from memorizing network configuration acquired through DHCP. It’s primarily used for troubleshooting, for example, for restoring SBC snapshots in the cloud environment, while preserving network configuration as specified in the snapshot.

2.10 #static-route

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

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

Where:

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

For example, to add a static route to the 10.3.0.0 subnet on the 1st interface, use the following:

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

2.11 #customer-id and #license-key

Customers who purchased "customer-id bound" bulk licenses from AudioCodes, should use the **#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.

2.12 #ini-default

The **#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.

2.13 #ini-incremental

The **#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 and therefore, this tag is not really useful. However, you can create snapshots of the SBC – with some pre-defined configuration – and use them to create new SBC instances. In such cases, the **#ini-incremental** hashtag may be used to adjust the image configuration instead of specifying a new configuration "from scratch".

2.14 #reset-password

The **#reset-password** hashtag resets all local users and passwords, while preserving the rest of the SBC configuration unchanged. The default user **Admin** with password **Admin** is created and may be used to log into the SBC. Make sure to remove the **#reset-password** hashtag from the VM's user-data after the SBC completes the reboot; otherwise, the Local Users table will be deleted again on every reboot.

2.15 #cloud-init

The **#cloud-init** hashtag causes SBC software to re-run the "first boot" cloud-init process, as described in Section 1.2, Boot Sequence. It may be used to trigger SBC re-configuration if VM's user data has changed. Make sure to remove the **#cloud-init** hashtag from the VM's user-data after the SBC completes the reboot; otherwise, the process will run on every reboot.

2.16 #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. Make sure to remove the **#write-factory** hashtag from the VM's user-data after the SBC completes the reboot; otherwise, the configuration will be deleted again on every reboot.

2.17 #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 that is irrelevant to the SBC.

3 Configuration Drive Emulation

Pure virtualized (non-cloud) environments (e.g., VMware or KVM) may use automatic provisioning as described in Section 2, "Automatic Configuration" Data, by emulating the "configuration drive" method of automatic provisioning.

Configuration drive is essentially a virtual CD-ROM attached to the running SBC instance that has a single FAT or ISO 9660 file system with the label `config-2` and configuration data 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-drive
rm -r /tmp/new-drive
```

Created `configdrive.iso` should be attached to the SBC instance as a virtual CD-ROM device during its first boot.

Consider using the **#dhcp-address** hashtag, as described in Section 2.7, #network-interfaces, to make the SBC memorize the network configuration acquired through DHCP.

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

AudioCodes Inc.
200 Cottontail Lane
Suite A101E
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-28664