

Custom Application Deployment Instructions

Release date: June 28, 2022

Contents

1. Overview	1
2. Difference Between v1.1.1 and v2.0.1 Container	2
3. Notes	3
4. Network Requirements	4
5. Install Container	6
5.1. Install Container via AC Console	6
5.2. Install Container via Gateway Console	7
6. Container and Gateway Interaction	9
7. Create APP Package	10
7.1. What Should be Included	10
7.2. Naming Rule	10
7.3. APP Package Example	10
7.4. APP Configuration Customization	11
8. Install and Delete APP	12
8.1. Install APP via AC Console	12
8.2. Install APP via Gateway Console	13
8.3. Delete APP	13
9. Container Port Forwarding	14
10. SSH into Container	14
10.1. Local SSH login	14
10.2. Remote SSH login with AC	15
11. Select a Specific Bluetooth Chip in Container	16

1. Overview

Cassia's Bluetooth gateways E1000, X1000, and X2000 support edge computing which can improve response time, reduce cost of data transmission & cloud service, and improve reliability, security and scalability. The user can run custom applications in the gateway's container.

The container and APP share CPU, memory, and storage with the host gateway. Please check the table below for more information. When there is no APP installed, the container CPU usage is usually lower than 5%, the container memory usage is usually lower than 1%, and there is about 1.1 GB storage free (container uses about 1.2 GB).

Type	S2000	E1000	X1000	X2000
Support edge computing	No	Yes	Yes	Yes
Maximum memory can be used by container and APP	N/A	128 MB	128 MB	700 MB
Maximum CPU can be used by container and APP	N/A	2 cores, 1.5 GHz	2 cores, 1.5 GHz	2 cores, 1.5 GHz
Maximum storage can be used by container and APP	N/A	2.3 GB	2.3 GB	2.3 GB

The first container version is v1.1.1 (can be installed on all gateway versions). The latest container version is v2.0.1 (can only be installed on gateway firmware 2.1.1 or higher version). Please check chapter 2 for the major difference. The user can keep using v1.1.1 container if they don't need the new features in v2.0.1 container. Please verify your APP together with the new container carefully before rolling out new container to more gateways, since the Ubuntu version changed and the pre-installed utilities and packages are different.

Please download the container firmware from the link below (this page is password protected, please get in touch with your Cassia sales representative for assistance):

<https://www.cassianetworks.com/knowledge-base/router-gateway-firmware/>

2. Difference Between v1.1.1 and v2.0.1 Container

Difference	Container v1.1.1	Container v2.0.1
Gateway firmware	Any version	v2.1.1 or higher version
Container OS	Ubuntu 16.04.3 LTS	Ubuntu 20.04.2 LTS
Default SSH password (username is "cassia")	cassia	cassia-xxxxxx (xxxxxx is the last 6 characters of gateway MAC in lowercase, e.g. the SSH password of gateway CC:1B:E0:E0:8E:B4 is cassia-e08eb4)
Default root password	Default password of root user is "cassia"	Please run command "sudo passwd root" to set password for root user before running "su -"
Pre-installed utilities and packages	Please check below table	Please check below table
ASP.NET support	ASP.NET Core not pre-installed	ASP.NET Core 3.1.16 pre-installed

Below is the list of utilities and packages pre-loaded in container v1.1.1 and v2.0.1.

Name	Container 1.1.1 Version	Container 2.0.1 Version
BlueZ	5.37	5.53
Bluetoothd	5.37	5.53
DBus	1.10.6	1.12.16
Python 2	2.7.12	2.7.18
Python 3	3.5.2	3.8.5
Python Pip	8.1.1	20.0.2
python-gobject	3.20.0	3.34.0
dbus	1.10.6	1.12.16
python3-dbus	No	1.2.16
Node	6.11.5	10.19.0

Name	Container 1.1.1 Version	Container 2.0.1 Version
Nodejs	4.2.6	10.19.0
NPM	3.10.10	No
node-gyp	6.11.5	No
noble	4.4.6	No
GCC	5.4.0	9.3.0
G++	5.4.0	9.3.0
curl	7.47.0	7.68.0
ASP.NET Core	No	3.1.16

3. Notes

Container functions

- Please remember to change the container's SSH password upon the first login.
- Cassia's gateway container uses a compact version of Ubuntu. Certain packages may not be pre-installed and/or available. Please see chapter 2 for information.
- Resetting the container will delete and re-create the container. The files under /opt will be kept. The custom APPs files not under /opt will be deleted.
- The user has SSH/root access to the Ubuntu container. However, Ubuntu is running as a container, so its core cannot be modified, and the properties of sysfs, e.g., /proc, is read-only.
- For security reasons, factory reset gateway will not impact the container, APP, and container files.
- From firmware 2.0.2, an option to enable and disable container local SSH login is added in the Container tab for security reasons. It is set to **OFF** by default. If you want to local SSH login the container, please turn it **ON**. Resetting the gateway will change this option into the default value **OFF**.

APP design

- Please implement APP log rotation to avoid flooding container storage. For the gateways with firmware versions below 1.4.3, if APP log floods gateway storage, the gateway may be offline and can't recover by a reboot. The user has to press the factory reset button to reset the gateway and then delete the container.
- It is suggested to keep the memory usage in the container below 70%. The other 30% is for peak hours and unusual cases.
- If the end user wants to upgrade an existing APP, please make sure that the name

and/or version is different.

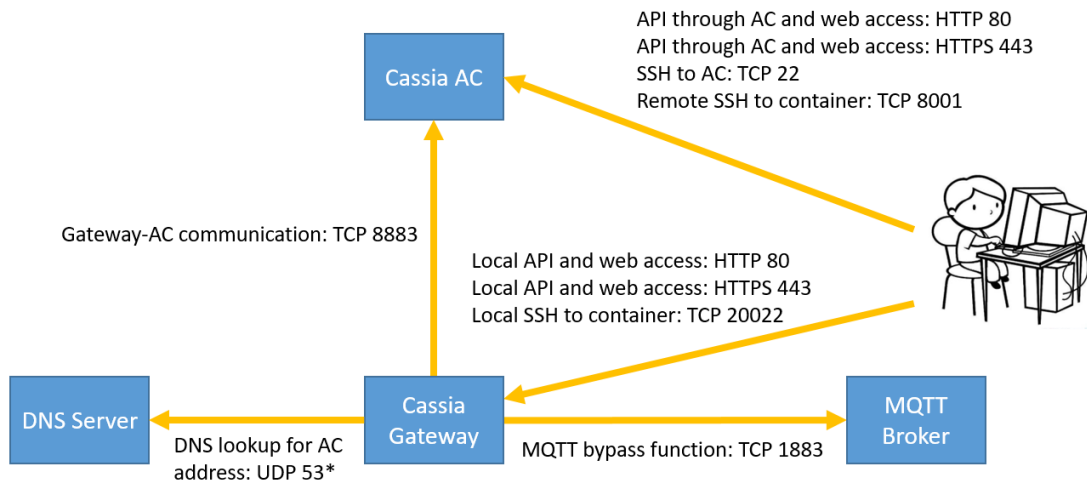
- Please consider compiling your APP code in a full development environment before loading and attempting to run in the container.
- Before firmware 2.0.3, when HTTPS is enabled on the gateway, the Cassia API URL used in the container APP should be updated to use port 443. From firmware 2.0.3, port 80 between container and gateway firmware is always enabled, so APP doesn't need to be updated regardless of the gateway's HTTPS settings.
- After upgrading gateway firmware to 2.0, if the APP uses BlueZ with Gatttool and Bluetoothd (e.g., noble or python Bluetooth lib) instead of Cassia Bluetooth stack and Cassia RESTful API, **please change 'Cassia Bluetooth Stack' to close** (in AC console -> Gateway -> Config-> Bluetooth Setting, or gateway console -> Other -> Bluetooth Setting). Otherwise, Bluetooth operations in the APP may return failure.

Cassia's Bluetooth stack and Cassia's RESTful API offer state of the art Bluetooth scan and connection performance. It is highly recommended to keep the Cassia Bluetooth stack open and use Cassia's RESTful API to achieve the best performance Bluetooth IoT system.

- From firmware 2.0, the output of the RESTful API to obtain gateway configuration from AC will be changed (GET `http://{your AC domain}/api/cassia/info?mac=<hubmac>`). The container status will be removed from the default API output, to avoid the oversized UDP packets problem. The user can get container status separately by the same API with the additional parameter 'fields=container'. Please refer to SDK WIKI for details.
- From gateway firmware 2.0, DNS name server in Cassia Bluetooth gateway will be propagated into container `/etc/resolv.conf`. Besides two default DNS name servers 8.8.8.8 and 114.114.114.114, the container will use the DNS setting in the Network section of the gateway webpage Config tab as an additional DNS name server. This feature solves the problem that the firewall blocks two default DNS servers.

4. Network Requirements

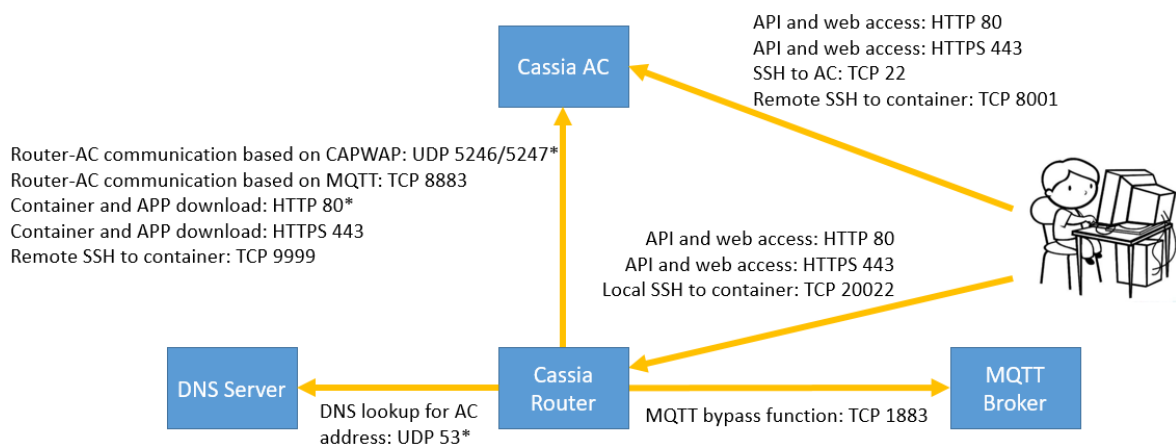
From v2.1.1, for the gateways that use MQTT to communicate with AC (default setting), the following ports are used and required for firewall configuration. TCP ports 80, 443 and 9999 are not required anymore.



Please make sure the following ports are opened outbound on the gateway side firewall.

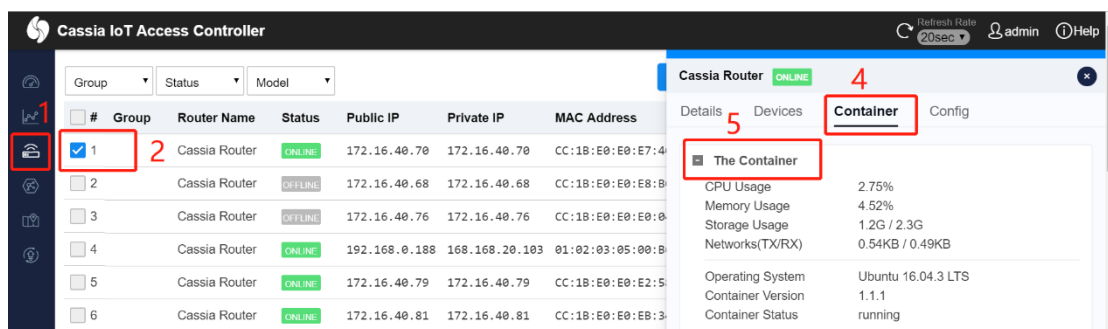
Type	Port	M/O	Description
TCP	8883	Mandatory	Gateway-AC communication
UDP	53	Mandatory*	DNS lookup for AC address. *Optional if internal DNS is specified in gateway network configuration
TCP	1883	Optional	For MQTT bypass function only

For the gateways that use CAPWAP to communicate with AC or the gateways using firmware below v2.1.1, the following ports may be used and required for firewall configuration.

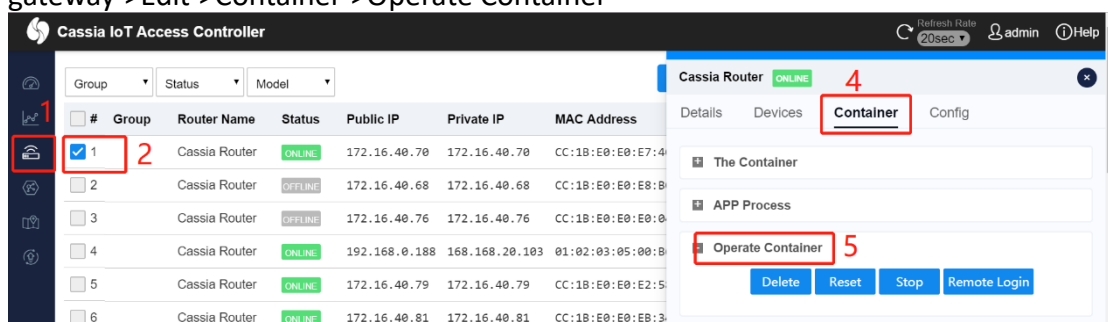


Please make sure the following ports are opened outbound on the gateway side firewall. The user can check if a TCP port is opened by using Netcat in gateway's local console.

Type	Port	M/O	Description
UDP	5246, 5247*	Mandatory	Gateway-AC communication based on CAPWAP. *Port 5246 and 5246 can be disabled after migrating gateway-AC communication to MQTT
TCP	8883		Gateway-AC communication based on MQTT (recommended from firmware v2.0.2)
HTTP	80*	Mandatory	Container and APP download from AC based on HTTP. *HTTP port 80 can be disabled if HTTPS is enabled
HTTPS	443		Container and APP download from AC based on HTTPS

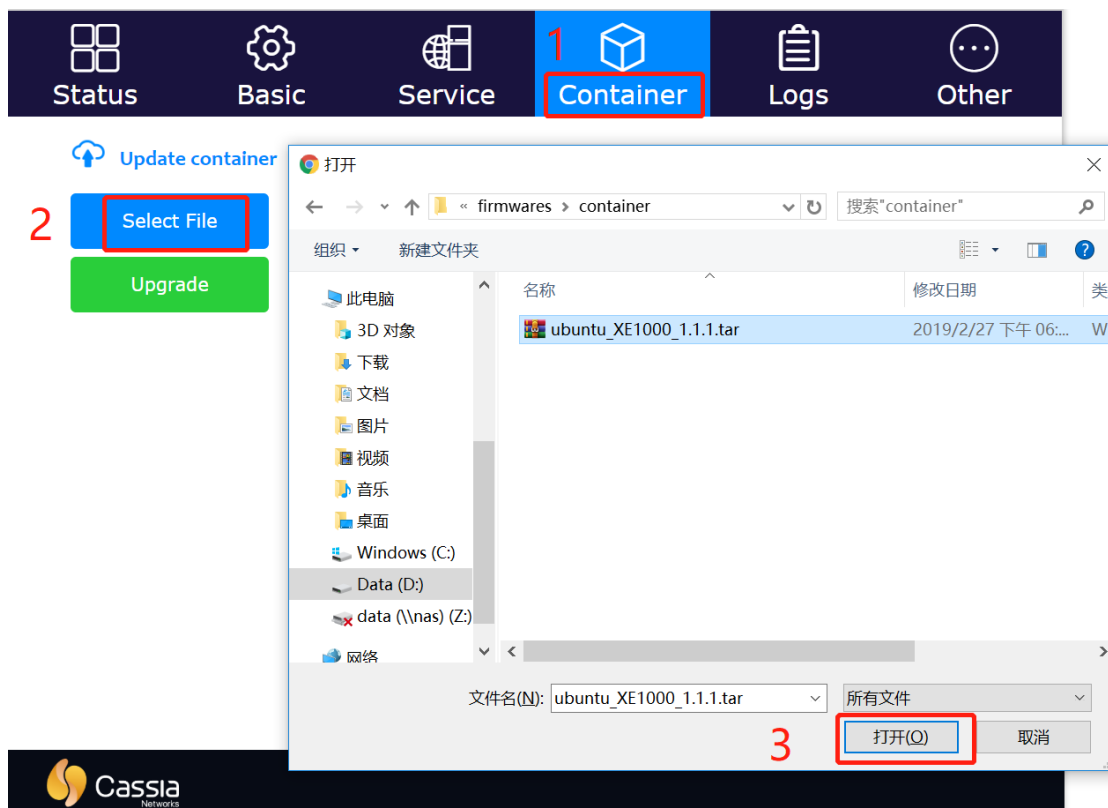


The user can also delete, reset, start, stop, and remote login containers from the AC. Please follow the steps on AC console: Gateways ->select gateway->Edit->Container->Operate Container

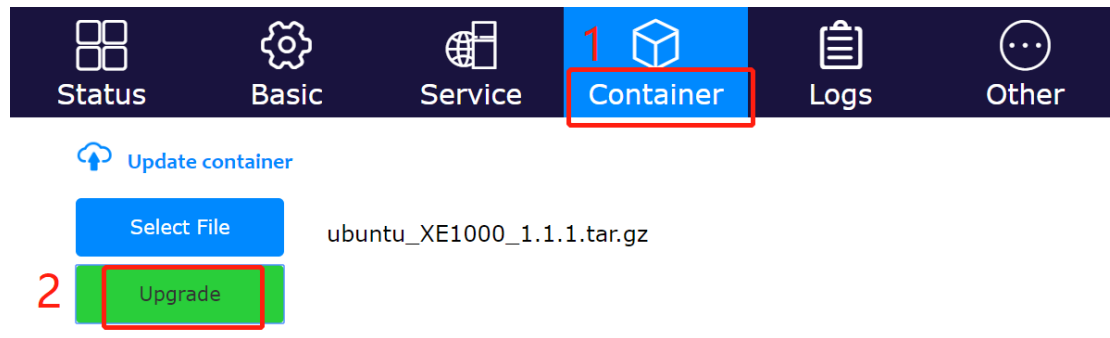


5.2. Install Container via Gateway Console

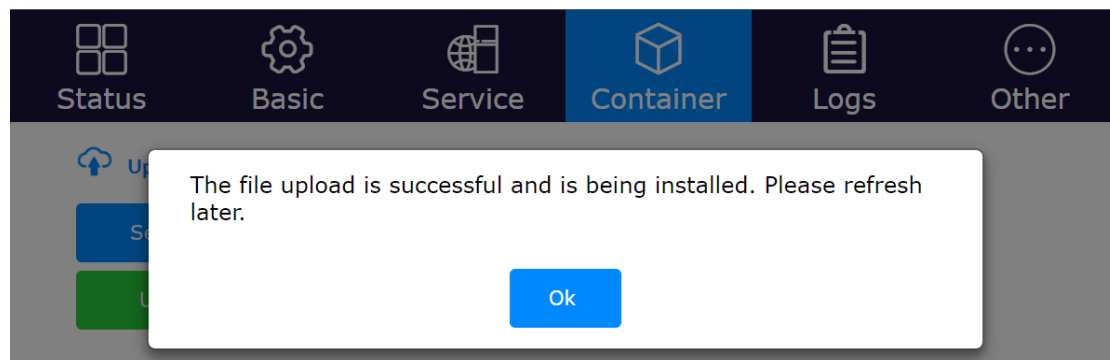
Please follow the steps on the gateway console: Container->Select File->Open.



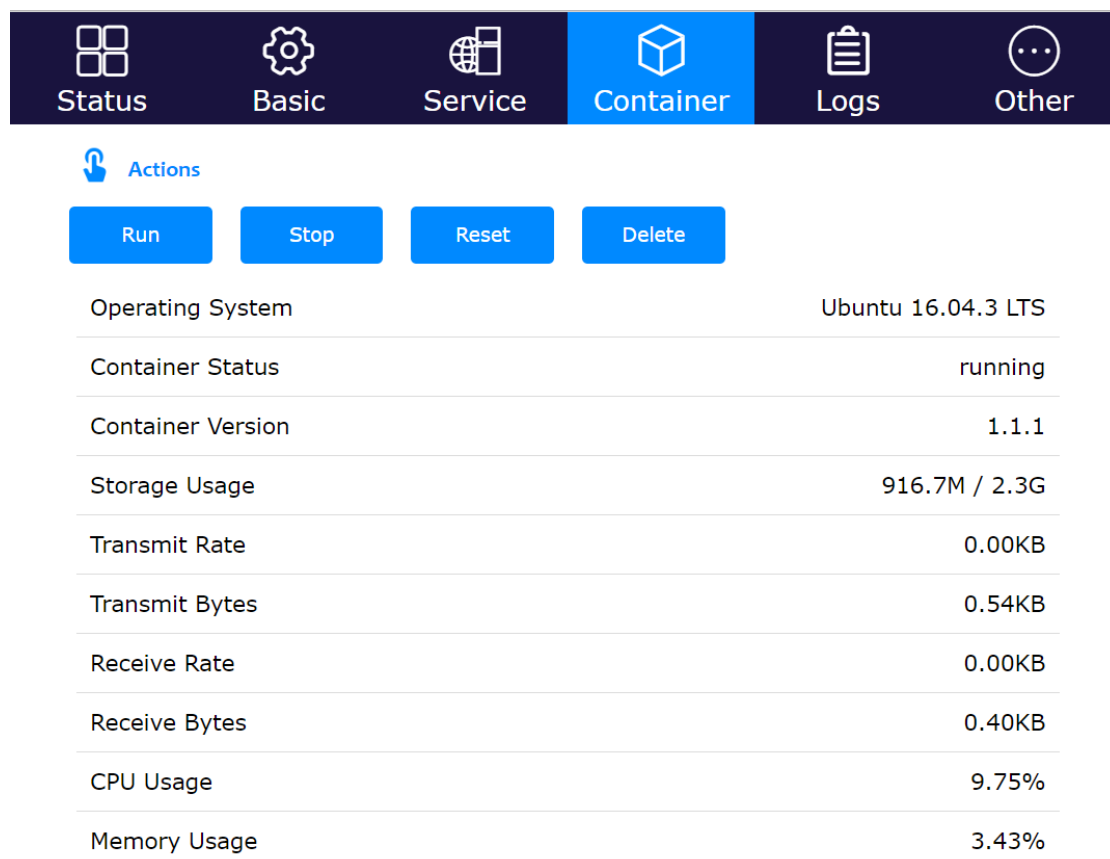
Click Container->Upgrade.



Now, the container will be uploaded and installed on the gateway.



Please refresh the web browser. You will see the container information in the Container tab. The user can install their APP or take actions for the container (Run, Stop, Reset, Delete).



6. Container and Gateway Interaction

After the container is installed, the gateway and container will be in the same subnet. The IP address of the gateway is 10.10.10.254/24. The IP address of the container is 10.10.10.253/24.

```
cassia@ubuntu:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:16:3e:28:51:9d
          inet addr:10.10.10.253  Bcast:10.10.10.255  Mask:255.255.255.0
          inet6 addr: fe80::216:3eff:fe28:519d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:883532 errors:0 dropped:0 overruns:0 frame:0
          TX packets:855270 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:210580223 (210.5 MB)  TX bytes:56505328 (56.5 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1250 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1250 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:116330 (116.3 KB)  TX bytes:116330 (116.3 KB)
```

The APP in the container can call local RESTful API like below (Turn on scanning as an example).

```
$curl -v 10.10.10.254/gap/nodes/?event=1&active=1
```

```
cassia@ubuntu:~$ curl -v http://10.10.10.254/gap/nodes/?event=1
* Trying 10.10.10.254...
* Connected to 10.10.10.254 (10.10.10.254) port 80 (#0)
> GET /gap/nodes/?event=1 HTTP/1.1
> Host: 10.10.10.254
> User-Agent: curl/7.47.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx
< Date: Fri, 27 Mar 2020 14:04:35 GMT
< Content-Type: text/event-stream
< Transfer-Encoding: chunked
< Connection: keep-alive
< Access-Control-Allow-Credentials: true
< Cache-Control: no-cache
< Access-Control-Allow-Headers: Content-Type
< X-Powered-By: Cassia
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Methods: GET, POST, PUT, DELETE
<
:keep-alive

data: {"name": "(unknown)", "evtType": 3, "rssi": -65, "adData": "1EFF06000109200236D61D3C1BD
A805DF34AFD924B0662B308CC3FDDA88702", "bdaddrs": [{"bdaddr": "19:36:03:67:DE:F7", "bdaddrT
ype": "random"}]}
```

7. Create APP Package

7.1. What Should be Included

The APP package should include all the program files, configuration files, and data files. It should also include a startup script (autorun.sh). From firmware v2.1.1, the user can include a delete script too (delete_app.sh).

First, the codes that the user wants to run automatically when the container starts up should be added in autorun.sh. Second, the user should save all the files following container root file system architecture, set access rights (read and write), and set user group permissions. Last, the user should use tar -zvcf command to build the APP package.

From firmware v2.1.1, the user can include a delete script (delete_app.sh) which will run automatically when deleting the APP from the AC or gateway console. The user can add codes in this script to clean up their APP logs, etc.

Please check APP sample code: <https://github.com/CassiaNetworks/CassiaSDKGuide>

7.2. Naming Rule

APP package name format: name.major.minor.tar.gz

- “Name” is the name of the APP, which is made up of the letter A-Z, a-z, digit 0-9, underline “_”, en dash “-”. The length of the name string should be equal to or less than 16 characters.
- “Major” is the major version number. Made up of 0-9 only. The length should be no more than 3 digits.
- “Minor” is the minor version number. Made up of 0-9 only. The length should be no more than 3 digits.
- tar.gz is the compression format of the APP package.

7.3. APP Package Example

Assuming the user wants to generate an APP named abcd with version 1.1., the APP includes startup script autorun.sh, delete script delete_app.sh, executable program file /usr/bin/eeee, configuration file /etc/ff.conf, and lib file /usr/lib/gg.so in folder abcd.1.1.

What is more, from firmware 1.4.1, the user can add meta.json under folder /root/config/<app_name>/ to define their own APP configuration console.

The file architecture looks like the below:

abcd.1.1/autorun.sh
abcd.1.1/delete_app.sh
abcd.1.1/etc/ff.conf
abcd.1.1/usr/bin/eeee
abcd.1.1/usr/lib/gg.so

[abcd.1.1/root/config/abcd/meta.json](#)

When a user runs the command: `tar-zcvf abcd.1.1.tar.gz abcd.1.1/`, the system will generate the APP package file `abcd.1.1.tar.gz`.

After installing this APP package on the gateway, the end user will find the below files in the container. The content of `abcd.sh` is the same as `autorun.sh`. The content of `abcd.uninst` is the same as `delete_app.sh`:

```
/etc/ff.conf
/usr/bin/eeee
/usr/lib/gg.so
/root/apps/abcd.sh
/root/apps/abcd.uninst
/root/apps/abcd.version
/root/config/abcd/meta.json
```

7.4. APP Configuration Customization

From firmware 1.4.1, the user can add their own APP configuration console in Cassia gateway's local console and the AC's console.

First, please create `meta.json` under folder `/root/config/<app_name>/`. This file defines the configuration items.

"type": "string" is mandatory. It is reserved for further usage. Now, we only support "string" for "type".

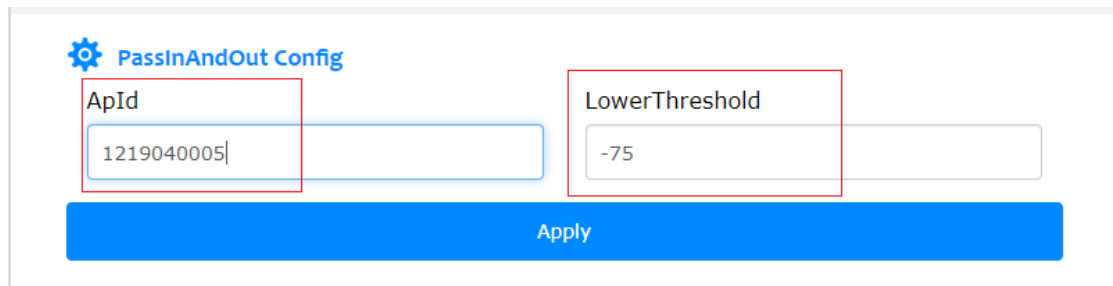
Below is the format:

```
{
  "name": "<app_name>",
  "config_items" : [
    {"name": "<configure_1>", "type": "string"},
    {"name": "<configure_2>", "type": "string"},
    {"name": "<configure_3>", "type": "string"},
    .....
    {"name": "<configure_n>", "type": "string"}
  ]
}
```

Below is an example:

```
{
  "name": "PassInAndOut",
  "config_items" : [
    {"name": "Apld", "type": "string"},
    {"name": "LowerThreshold", "type": "string"}
  ]
}
```

Then, on the gateway's local console and the AC's console, the user can find their own APP configuration console as seen below.



The image shows a configuration interface titled "PassInAndOut Config". It contains two input fields: "ApId" with the value "1219040005" and "LowerThreshold" with the value "-75". Both fields are highlighted with red boxes. Below these fields is a large blue button labeled "Apply".

After the configuration, the gateway will generate file config.json under folder /root/config/<app_name>/.

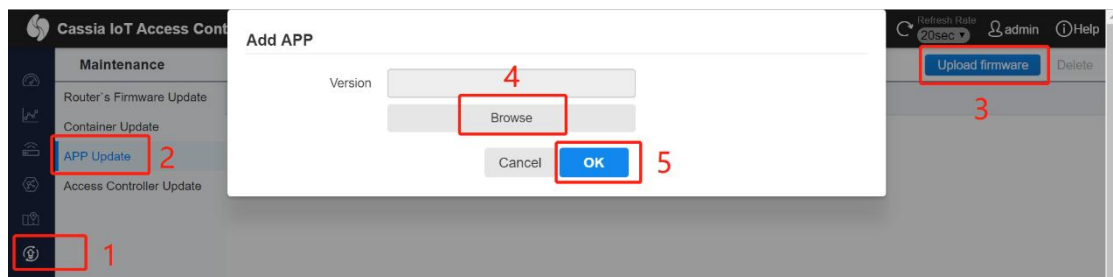
Below is an example:

```
{
  " ApId": "1219040005",
  " LowerThreshold": "-75"
}
```

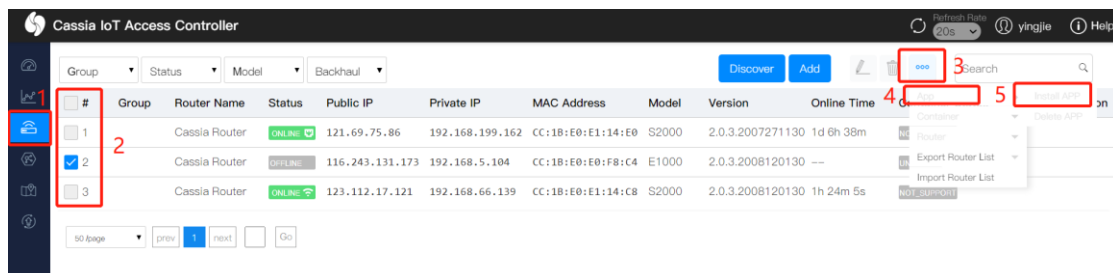
8. Install and Delete APP

8.1. Install APP via AC Console

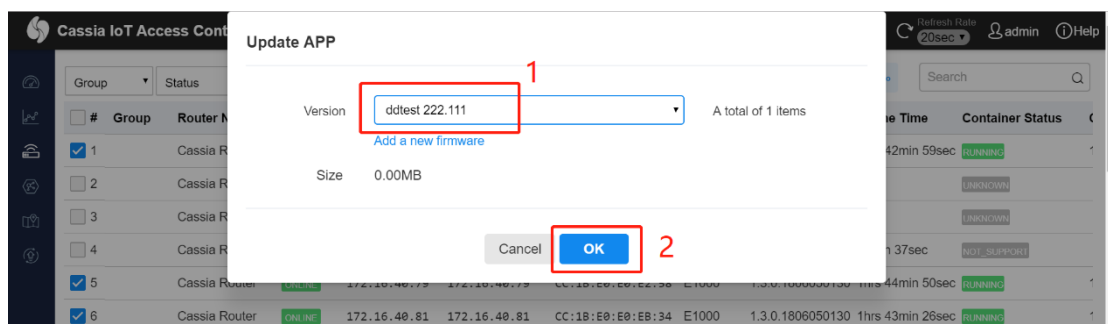
Please follow the steps on the AC console to upload the APP to the AC:
Maintenance->APP Update->Upload firmware->Browse->OK.



Please follow the steps on the AC console to install an APP to a batch of gateways:
Gateways->select the gateways that need to install APP->More->App->Install APP.



Please select the APP version and click the OK button. The APP will be installed or updated. From firmware 2.1.1, APP can be continuously downloaded after being disrupted (MQTT should be used between gateway and AC).

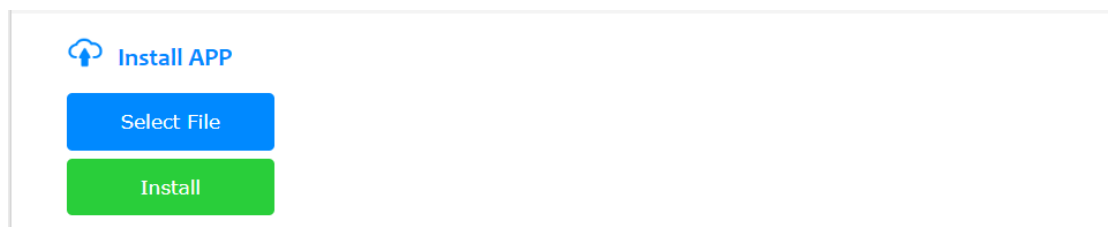


Now, the user can check the installed APPs on the container page of AC and gateway console.

The user can also check the first installed APP for a batch of gateways on the AC gateway page.

8.2. Install APP via Gateway Console

The user can also install their APP via the gateway console.



After installation, the user can check the installed APP and programs in operation.



8.3. Delete APP

From firmware 2.0.3, the user can delete the installed APP by clicking the Del button. This action will delete /root/apps/\${app_name}.sh and /root/apps/\${app_name}.version, but will remain all other APP files unchanged (avoid deleting important customer data).

Please delete the other APP files, if necessary, by adding delete script (recommended, available from firmware v2.1.1), adding codes in autorun.sh of the new APP, or by SSH into the container.

9. Container Port Forwarding

From firmware 1.4.1, the user can configure a maximum of four TCP or UDP ports for container port forwarding.

Port Forwarding Configuration

Protocol	Port
TCP	60000
N/A	
TCP	
UDP	
N/A	

Protocol: N/A Port:

Protocol: N/A Port:

Protocol: N/A Port:

Apply

By using this functionality, the user can set up a server (for example webserver) in the container and access it via the gateway's private IP address and the configured port. The port range is [60000, 65525]. N/A means the port is closed.

10. SSH into Container

The SSH username is "cassia". The default SSH password of the container version 1.1.1 is "cassia". The default SSH password of the container version 2.x is "cassia-xxxxxx" (xxxxxx is the last 6 characters of gateway's MAC in lowercase, e.g., the SSH password of gateway CC:1B:E0:E0:8E:B4 is cassia-e08eb4).

NOTE: Please remember to change the container's SSH password upon the first login.

10.1. Local SSH login

NOTE: From firmware 2.0.2, an option to enable and disable container local SSH login is added in the Container tab. The container local SSH login is OFF by default for security reasons. Please turn it **ON** before you want to local SSH login the container. Resetting the gateway will change this option into the default value **OFF**.

If localhost and gateway are in the same network, please follow the below examples (x.x.x.x is the IP address of gateway and cassia is the username).

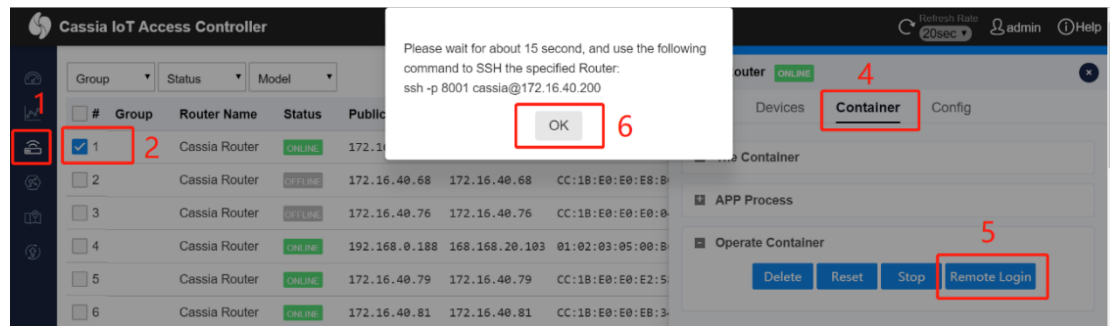
Login example with Linux Shell "ssh -p 20022 cassia@x.x.x.x"

Login example with Xshell "ssh cassia@x.x.x.x 20022"

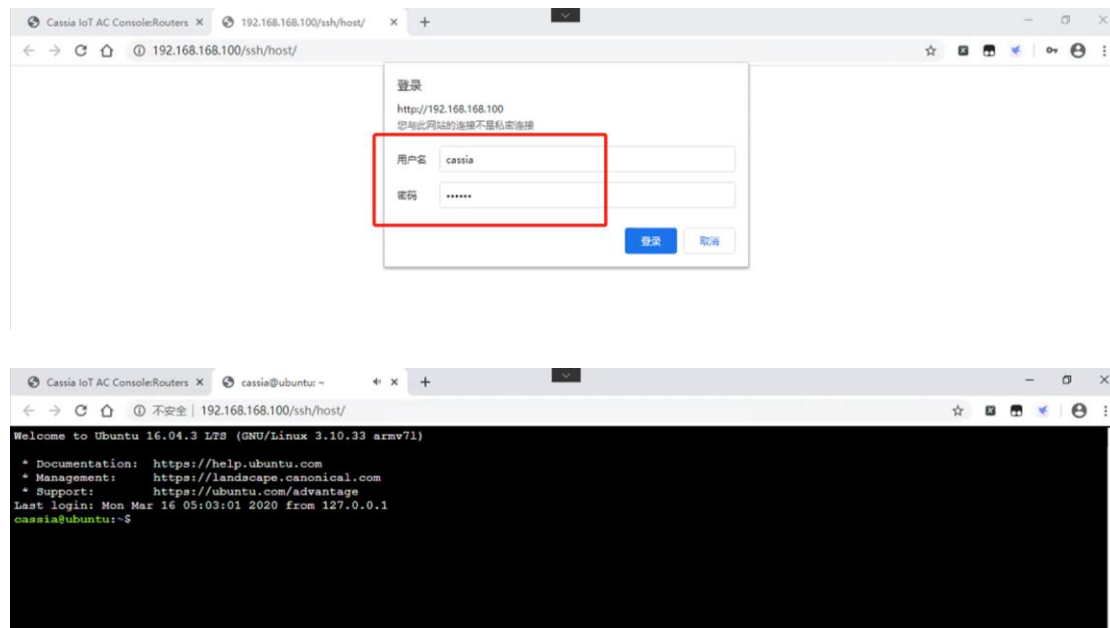
10.2. Remote SSH login with AC

If localhost and gateway are in a different network and the gateway is running in AC managed mode, please follow steps on the AC console: Gateways ->select gateway ->Edit->Container->Remote Login->OK.

Before firmware 2.0, please follow the guideline in the popup window.



From firmware 2.0, a new tab will pop up on the web browser as an SSH terminal. After pressing the “Remote Login” button, the user can still log in to the container by “ssh -p 8001 cassia@ac_ip_address” in other terminals.



After logging into the container with SSH, the user can control Bluetooth with RESTful API, noble, or dbus. Users can also use apt-get command to install Linux tools. For example, the user can run “apt-get install lrzsz” to install lrzsz.

In the AC console, users are limited to establishing and maintaining a single SSH session at any given time. It does not support concurrent SSH access by multiple administrative users.

11. Select a Specific Bluetooth Chip in Container

Cassia's E1000, X1000, and X2000 gateways include two Bluetooth chips. The user can select a specific Bluetooth chip in the container.

If the user is using Cassia's RESTful API, please add the parameter "-chip".

- chip (Optional): 0 or 1, indicates which chip of the Cassia gateway is used for scan and connect. By default, the gateway will pick up the chip automatically based on an internal algorithm. The S2000 only supports chip 0. The E1000, X1000 and X2000 support 0 and 1.

If the user is using BlueZ, please follow the below guideline. **NOTE:** It is highly recommended to use Cassia's RESTful API which has better performance and filters.

BlueZ has two ways to work with Bluetooth chips.

- Through BluetoothD and DBUS: BluetoothD will register interface on DBUS after starts up. It has a fixed interface.
- Call BlueZ command, such as hcitool, gatttool, etc. below are some examples.

```
hcitool -i hci0 lescan    // open scan for chip 0
hcitool -i hci1 lescan    // open scan for chip 1
gatttool -i hci0 -t public -b CC:1B:E0:01:02:03    // connect by chip 0
```