RAUC hawkBit Updater

Release 1.1

Lasse Klok Mikkelsen, Enrico Jörns, Bastian Krause

Nov 15, 2021

CONTENTS

CHAPTER

ONE

USING THE RAUC HAWKBIT UPDATER

1.1 Authentication

As described on the hawkBit Authentication page in the "DDI API Authentication Modes" section, a device can be authenticated with a security token. A security token can be either a "Target" token or a "Gateway" token. The "Target" security token is specific to a single target defined in hawkBit. In the RAUC hawkBit updater's configuration file it's referred to as `auth_token`.

Targets can also be connected through a gateway which manages the targets directly and as a result these targets are indirectly connected to the hawkBit update server. The "Gateway" token is used to authenticate this gateway and allow it to manage all the targets under its tenant. With RAUC hawkBit updater such token can be used to authenticate all targets on the server. I.e. same gateway token can be used in a configuration file replicated on many targets. In the RAUC hawkBit updater's configuration file it's called `gateway_token`. Although gateway token is very handy during development or testing, it's recommended to use this token with care because it can be used to authenticate any device.

1.2 Plain Bundle Support

RAUC takes ownership of plain format bundles during installation. Thus rauc-hawkbit-updater can remove these bundles after installation only if it they are located in a directory belonging to the user executing rauc-hawkbit-updater.

1.2.1 systemd Example

To store the bundle in such a directory, a drop-in `rauc-hawkbit-updater.service.d/10-plain-bundle.conf` can be created:

```
[Service]
ExecStartPre=/bin/mkdir -p /tmp/rauc-hawkbit-updater/
```

The bundle location needs to be set in rauc-hawkbit-updater's config:

```
bundle_download_location = /tmp/rauc-hawkbit-updater/bundle.raucb
```

CHAPTER

TWO

REFERENCE

2.1 Configuration File

Example configuration:

```
[client]
hawkbit_server            = 127.0.0.1:8080
ssl                       = false
ssl_verify                = false
tenant_id                 = DEFAULT
target_name               = test-target
auth_token                = bhVahL1Il1shie2aj2poojeChee6ahShu
bundle_download_location  = /tmp/bundle.raucb

[device]
key1                      = valueA
key2                      = valueB
```

[client] section

Configures how to connect to a hawkBit server, etc.

Mandatory options:

hawkbit_server=<host>[:<port>] The IP or hostname of the hawkbit server to connect to (Punycode representation must be used for host names containing Unicode characters). The `port` can be provided optionally, separated by a colon.

target_name=<name> Unique `name` string to identify controller.

auth_token=<token> Controller-specific authentication token. This is set for each device individually. For details, refer to https://www.eclipse.org/hawkbit/concepts/authentication/.

> **Note:** Either `auth_token` or `gateway_token` must be provided

gateway_token=<token> Gateway authentication token. This is a tenant-wide token and must explicitly be enabled in hakwBit first. It is actually meant to authenticate a gateway that itself manages/authenticates multiple targets, thus use with care. For details, refer to https://www.eclipse.org/hawkbit/concepts/authentication/.

Note: Either `auth_token` or `gateway_token` must be provided

bundle_download_location=<path> Full path to where the bundle should be downloaded to. E.g. set to `/tmp/_bundle.raucb` to let rauc-hawkbit-updater use this location within `/tmp`.

Optional options:

tenant_id=<ID> ID of the tenant to connect to. Defaults to `DEFAULT`.

ssl=<boolean> Whether to use SSL connections (`https`) or not (`http`). Defaults to `true`.

ssl_verify=<boolean> Whether to enforce SSL verification or not. Defaults to `true`.

connect_timeout=<seconds> HTTP connection setup timeout [seconds]. Defaults to `20` seconds.

timeout=<seconds> HTTP request timeout [seconds]. Defaults to `60` seconds.

retry_wait=<seconds> Time to wait before retrying in case an error occurred [seconds]. Defaults to `60` seconds.

low_speed_time=<seconds> Time to be below `low_speed_rate` to trigger the low speed abort. Defaults to `60`. See https://curl.se/libcurl/c/CURLOPT_LOW_SPEED_TIME.html.

low_speed_rate=<bytes per second> Average transfer speed to be below during `low_speed_time` seconds to consider transfer as "too slow" and abort it. Defaults to `100`. See https://curl.se/libcurl/c/CURLOPT_LOW_SPEED_LIMIT.html.

resume_downloads=<boolean> Whether to resume aborted downloads or not. Defaults to `false`.

post_update_reboot=<boolean> Whether to reboot the system after a successful update. Defaults to `false`.

Important: Note that this results in an immediate reboot without contacting the system manager and without terminating any processes or unmounting any file systems. This may result in data loss.

log_level=<level> Log level to print, where `level` is a string of

- `debug`
- `info`
- `message`
- `critical`
- `error`
- `fatal`

Defaults to `message`.

[device] section

This section allows to set a custom list of key-value pairs that will be used as config data target attribute for device registration. They can be used for target filtering.

Important: The [device] section is mandatory and at least one key-value pair must be configured.

CHAPTER

THREE

CONTRIBUTING

Thank you for thinking about contributing to RAUC hawkBit Updater! Various backgrounds and use-cases are essential for making RAUC hawkBit Updater work well for all users.

The following should help you with submitting your changes, but don't let these guidelines keep you from opening a pull request. If in doubt, we'd prefer to see the code earlier as a work-in-progress PR and help you with the submission process.

3.1 Workflow

- Changes should be submitted via a GitHub pull request.
- Try to limit each commit to a single conceptual change.
- Add a signed-of-by line to your commits according to the *Developer's Certificate of Origin* (see below).
- Check that the tests still work before submitting the pull request. Also check the CI's feedback on the pull request after submission.
- When adding new features, please also add the corresponding documentation and test code.
- If your change affects backward compatibility, describe the necessary changes in the commit message and update the examples where needed.

3.2 Code

- Basically follow the Linux kernel coding style

3.3 Documentation

- Use semantic linefeeds in .rst files.

3.4 Developer's Certificate of Origin

RAUC hawkBit Updater uses the Developer's Certificate of Origin 1.1 with the same process as used for the Linux kernel:

> Developer's Certificate of Origin 1.1
>
> By making a contribution to this project, I certify that:
>
> (a) The contribution was created in whole or in part by me and I have the right to submit it under the open source license indicated in the file; or
>
> (b) The contribution is based upon previous work that, to the best of my knowledge, is covered under an appropriate open source license and I have the right under that license to submit that work with modifications, whether created in whole or in part by me, under the same open source license (unless I am permitted to submit under a different license), as indicated in the file; or
>
> (c) The contribution was provided directly to me by some other person who certified (a), (b) or (c) and I have not modified it.
>
> (d) I understand and agree that this project and the contribution are public and that a record of the contribution (including all personal information I submit with it, including my sign-off) is maintained indefinitely and may be redistributed consistent with this project or the open source license(s) involved.

Then you just add a line (using `git commit -s`) saying:

> Signed-off-by: Random J Developer <random@developer.example.org>

using your real name (sorry, no pseudonyms or anonymous contributions).

CHAPTER

FOUR

CHANGES IN RAUC HAWKBIT UPDATER

4.1 Release 1.1 (released Nov 15, 2021)

Enhancements

- RAUC hawkBit Updater does now handle hawkBit cancellation requests. This allows to cancel deployments that were not yet received/downloaded/installed. Once the installation has begun, cancellations are rejected. [#89]
- RAUC hawkBit Updater now explicitly rejects deployments with multiple chunks/artifacts as these are conceptually unsupported by RAUC. [#103]
- RAUC hawkBit Updater now implements waiting and retrying when receiving HTTP errors 409 (Conflict) or 429 (Too Many Requests) on DDI API calls. [#102]
- Enable TCP keep-alive probing to recognize and deal with connection outages earlier. [#101]
- New configuration options `low_speed_time` and `low_speed_time` allow to adjust the detection of slow connections to match the expected environmental conditions. [#101]
- A new option `resume_downloads` allows to configure RAUC hawkBit Updater to resume aborted downloads if possible. [#101]
- RAUC hawkBit Updater now evaluates the deployment API's 'skip' options for download and update (as e.g. used for maintenance window handling). Depending on what attributes are set, this will skip installation after download or even the entire update. [#111]

Testing

- replaced manual injection of temporary env modification by monkeypatch fixture
- test cases for all new features were added

Documentation

- Added note on requirements for storage location when using plain bundle format

4.2 Release 1.0 (released Sep 15, 2021)

This is the initial release of RAUC hawkBit Updater.

The RAUC hawkBit updater is a simple commandline tool / daemon written in C (glib). The daemon runs on your target and operates as an interface between the RAUC D-Bus API and the hawkBit DDI API.