Welcome

Introducing Afero >

Tutorials >

Profile Editor User Guide >

Inspector User Guide

Developer Hub Setup

Cloud API >

Firmware Reference

MCU to ASR Communication >

Device Attribute Message Protocol

Device Attribute Registry

Setting Time on the MCU

After ASR links with the Afero Cloud, it updates two attribute values related to the time-of-day. These two attributes, “Linked Timestamp” and “UTC Offset”, are delivered to an attached MCU rapidly enough that the local time can be set on the MCU with a reasonable degree of accuracy.

Two Time-Relevant Attributes

The two attributes you’ll use to set local time on the MCU are described below:

- 65015 - LINKED_TIMESTAMP (signed 32-bit long)

  This is a UNIX Epoch (number of seconds since 00:00:00 1/1/1970) UTC timestamp marking when ASR successfully last linked to the Afero Cloud. When the timestamp is returned shortly after ASR reboots, it should be reasonably close to actual current time.

  The latency from the Cloud back to the MCU should typically be less than a couple of seconds, but don’t rely on this timestamp being more accurate than +/- one minute or so. If your TOD requirements are not critical, it’s a usable timestamp that’s given to you automatically.

- 65001 - UTC_OFFSET_DATA (byte[8])

  This byte array contains three pieces of information: the current local timezone offset from UTC, the next UTC offset, and a UNIX Epoch timestamp of when the “next” UTC offset is valid. Specifically:

  - [0-1] - Little-endian signed int containing the local timezone offset from UTC in minutes
  - [2-5] - Little-endian unsigned long containing an Epoch timestamp (UTC) for “next” offset validity
  - [6-7] - Little-endian signed int containing the “next” local timezone offset from UTC in minutes

Attribute Value Change Rules

afLib2 for Arduino API >

MCU Coding Tips

Setting Time on the MCU

Hardware Reference >

Tech & App Notes

Training Labs >

Release Notes >

UTC Offset is determined by the Location attributes set (by you) for the ASR and **are not dynamic** in any way. The UTC Offset can and will be wrong if the Location in the device configuration is incorrect. You can set the correct Location for your Afero device using the Afero mobile app, or by using the Afero Inspector developer tool at https://inspector.afero.io.

Let's look at examples of these attributes and what their values mean. These attribute values:

```
LINKED_TIMESTAMP=0x5a2b06c3
UTC_OFFSET_DATA=0xD4FEF0D3A45A10FF
```

Translate to:

```
Link Time: 0x5a2b06c3 (Fri 2017-12-08 21:40:19 GMT)
Current UTC Offset: -300 minutes (0xFED4) (Eastern Standard Time, in this example)
UTC Offset Change: Sun 2018-03-11 07:00:00 GMT (0x5AA4D3F0) (Spring 2018 USA DST Time Change,
2am local time)
Next UTC Offset: -240 minutes (0xFF10) (Eastern Daylight Time, in this example)
```

Only assume that LINKED_TIMESTAMP is current when ASR state transitions to a **connected state**. You can use `af_lib_get_attribute` to return the Linked Timestamp at any point, but remember the result returned will be the time ASR first linked or re-linked to the Cloud, which could be significantly in the past if the device has been connected for a long time.

aflib_time_check Example Code

In afLib2, we've included the `aflib_time_check` example application, which provides a simple method for:

- Listening for the Linked Timestamp and UTC Offset attributes in `attrNotifyHandler`, and
- Setting a C time struct that can be manipulated within your application.

Example Code Notes

- For best results, use a Real-Time Clock (RTC) chip and use the Linked Timestamp and UTC Offset attributes to set the RTC. By doing this you'll have access to a reasonably accurate clock. Remember, the Linked Timestamp is only sent to the MCU when ASR links or re-links to the Afero Cloud.
- The device profile created using the Afero Profile Editor doesn't need any specific support for Linked Timestamp or UTC Offset attributes, as they are system attributes and will be presented to the MCU as long as the Afero device profile has MCU support **enabled**.
- The example code provided in afLib2 includes some code that protects against accepting LINKED_TIMESTAMP when an MCU application requests that data via `af_lib_get_attribute`. Since the value returned is the timestamp of when ASR last linked to the Afero Cloud, this value will not update except when ASR re-links its connection. If this level of paranoia isn't needed, all references and checks to "timestamp_read" can be removed.
- When ASR first boots, it will send an SYSTEM_UTC_OFFSET_DATA attribute update to the MCU; however, all data in this attribute will be zero. Since UTC Offset=0 is valid, you should check for the value of the UTC Offset Change Timestamp (the middle four bytes of the attribute value). If this timestamp is zero then the UTC Offset data is invalid and can be ignored. Once ASR links, you will receive the Linked Timestamp attribute and another UTC Offset attribute with valid data. In the case of a device with no Location or local timezone defined, the UTC_OFFSET_DATA attribute will remain all zeroes and can be safely ignored.
- It is possible to receive a new SYSTEM_UTC_OFFSET_DATA attribute update at any time, in response to the location of the device being updated in the Afero Cloud, or the passing of a Daylight Saving Time boundary. As long as the Offset Change Time Timestamp is greater than the Linked Timestamp (or the Current Timestamp if you have an RTC), then the update data should be considered valid and the RTC updated accordingly (typically by subtracting the "old" offset and adding the "new" offset to re-adjust the clock to the new timezone).
- In C/C++, the `strftime()` call can be used to format either timestamp into human-readable forms.
- Data types for the timestamp attribute values are:
    - int32_t linked_timestamp
    - int16_t utc_offset_now

- int16_t utc_offset_next
- int32_t utc_offset_change_time

Updated June 6, 2018