The afLib Lifecycle: The transports, create(), and destroy()

Every call to an afLib function requires a pointer to afLib that has been initialized for the communications protocol to be used. This means that initializing afLib is a basic task you'll perform in each of your scripts. Typically, within `setup()`, you will:

- Call `arduino_transport_create_<protocol>()` to set up the protocol of choice.
- Call `af_lib_create_with_unified_callback()` to initialize afLib.

At the *end* of your use of afLib, you may want to tear it down. This is not common, but if your application involves a complete shutdown and re-initialization of afLib within a continuous script run, you may call the `destroy()` method (presumably in preparation for initializing afLib again).

This page describes the afLib functions related to initializing and terminating afLib.

arduino_transport_create_<*protocol*>()

Description

Initialize an `af_transport_t` appropriate for the communications protocol to be used by the application. A pointer to the `af_transport_t` is a required argument for `af_lib_create_with_unified_callback()`.

There is a specific form of `af_transport_t` for each of the communication protocols available, SPI and UART. You will need to create the `af_transport_t` corresponding to the protocol used in your project, and pass it to `af_lib_create_with_unified_callback()`.

SPI Syntax

```
af_transport_t* arduino_transport_create_spi(int chipSelect);
```

Parameters for arduino_transport_create_spi

`chipSelect`	The number of the pin to be used for SPI Chip Select. It is 10 for Teensy and Uno.

UART Syntax

```
af_transport_t* arduino_transport_create_uart(uint8_t rxPin, uint8_t txPin)
```

Parameters for arduino_transport_create_uart

`rxPin`	The number of the pin to be used for UART RX. It is 7 for Teensy and Uno.
`txPin`	The number of the pin to be used for UART TX. It is 8 for Teensy and Uno.

Returns

Pointer to an `af_transport_t`.

af_lib_create_with_unified_callback()

Description

Initialize an instance of afLib for the application. The parameters include a callback function to let your code know when things have changed, and a pointer to the transport specific to your communication protocol.

Syntax

```
af_lib_t* af_lib_create_with_unified_callback(af_lib_event_callback_t event_cb, af_transport_t *transport)
```

Parameters

`event_cb`	The callback (`attrEventCallback()`) to let your MCU code know that one of the ASR attributes has changed.
`transport`	Pointer to an `af_transport_t` to handle communications.

Returns

Pointer to an instance of afLib to handle communications.

SPI Example

The SPI example below shows how we create a new instance of afLib for SPI:

- Instantiate an `af_transport_t` for SPI, supplying the pin number for Chip Select.
- Supply `af_transport_t` as an argument to `af_lib_create_with_unified_callback()`, along with `attrEventCallback` as the callback method.
- Call `arduino_spi_setup_interrupts` to give afLib access to interrupts.

```
#include <SPI.h>

#include "af_lib.h"
#include "arduino_spi.h"
#include "af_module_commands.h"
#include "af_module_states.h"
#include "arduino_transport.h"

// Pin Defines assume Arduino Uno
#define CS_PIN                    10
#define INT_PIN                    2

af_lib_t *af_lib;

void attrEventCallback(const af_lib_event_type_t eventType,
                       const af_lib_error_t error,
                       const uint16_t attributeId,
                       const uint16_t valueLen,
                       const uint8_t* value) {
  // Any application-specific actions
}

void setup() {
  // Initialize afLib
  af_transport_t* arduinoSPI = arduino_transport_create_spi(CS_PIN);
  af_lib = af_lib_create_with_unified_callback(attrEventCallback, arduinoSPI);
  af_lib_error_t err = arduino_spi_setup_interrupts(af_lib, digitalPinToInterrupt(INT_PIN));
  if (err != AF_SUCCESS) {
      Serial.print("*** SPI setup failed with error: "); Serial.println(err);
  }
}

void loop() {
  // Give afLib processing time by calling af_lib_loop(af_lib); in sketch's loop()
  af_lib_loop(af_lib);
}
```

UART Example

In the UART example, we create a new instance of afLib for UART. In this case we:

- Instantiate an `af_transport_t` for UART, supplying RX and TX pin numbers.
- Call `af_lib_create_with_unified_callback()`, with `attrEventCallback` as the callback method and the UART transport pointer.

```
#include "af_lib.h"
#include "arduino_uart.h"
#include "af_module_commands.h"
#include "af_module_states.h"
#include "arduino_transport.h"

#define RX_PIN                7
#define TX_PIN                8

af_lib_t *af_lib;

void attrEventCallback(const af_lib_event_type_t eventType,
                       const af_lib_error_t error,
                       const uint16_t attributeId,
                       const uint16_t valueLen,
                       const uint8_t* value) {
  // Any application-specific actions
}

void setup() {
  // Initialize afLib
  af_transport_t *arduinoUART = arduino_transport_create_uart(RX_PIN, TX_PIN);
  af_lib = af_lib_create_with_unified_callback(attrEventCallback, arduinoUART);
}

void loop() {
  // Give afLib processing time by calling af_lib_loop(af_lib); in sketch's loop()
  af_lib_loop(af_lib);
}
```

af_lib_destroy()

Description

Destroy a previously created af_lib_t instance

Syntax

```
void af_lib_destroy(af_lib_t* af_lib)
```

Parameters

`af_lib`	Pointer to an af_transport_t to be destroyed.

Returns

None.

Next: *The Main Loop and the Request Queue*

Updated September 25, 2019