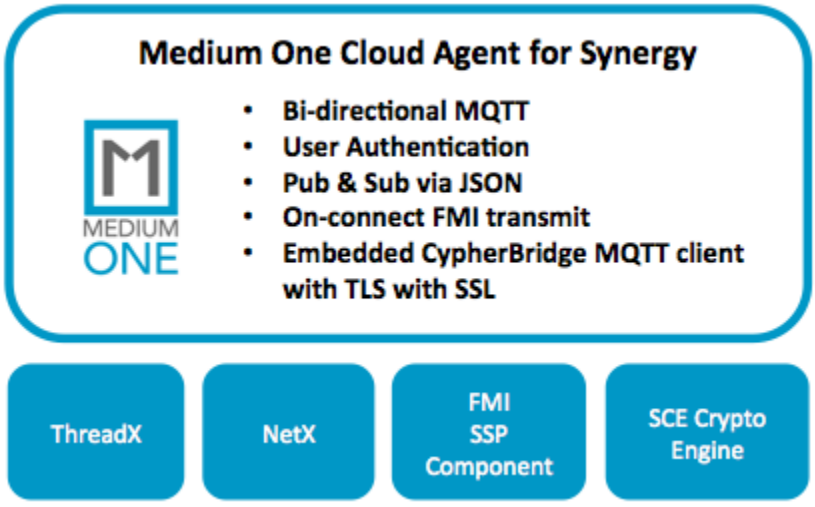
Medium One Cloud Agent for Synergy Documentation

Revision Date: 11/03/16
Supported VSA Version: 1.16.11.03

Table of Contents

Introduction

Medium One's Cloud Agent for Synergy ("Agent") provide APIs to enable bi-directional data communication to Medium One's cloud services including Renesas IoT Sandbox and the Synergy Platform. The Value Software Add-on (VSA) will provide an agent that securely communicates over MQTT.



Revision 1.1

System Requirements

Supported kits:

- S3A7 (IoT Fast Prototyping Kit)
- S7G2 (SK-S7G2)

Resources:

- 183k code rom
- 40k data ram
- 48 MHz

Pre-requisites

The following guides are recommended to become familiar with Medium One cloud concepts.

Getting Started with Renesas IoT Sandbox

Pre-built Demos

The following demos utilize the Agent and provide a great way to see a sample implementation.

Smart Chef Demo on S3A7 Fast Prototyping Kit

API Documentation

```
int m1_connect(char * mqtt_url,
        int mqtt_port,
        char * mqtt_user_id,
        char * password,
        char * mqtt_project_id,
        char * api_key,
        char * device_id,
        int retry_limit,
        int retry_delay,
        int mqtt_heart_beat,
        int tls_enabled)
```


Revision 1.1

- Description: Specifies API credentials and establishes a MQTT connection to Medium One. During m1_connect(), if connection is successful, the following data is transmitted to the Medium One *raw* datastream
    - “connected”: true
    - "ip_address": <WAN_IP>
    - "unique_id": <Unique ID> (from FMI)
    - "product_name": <Product Name> (from FMI)
    - "product_market": <Product Market> (from FMI)
    - "mask_revision": <Mask Revision> (from FMI)
    - "quality_code": <Quality Code> (from FMI)
    - "ssp_version": <SSP Version> (ssp_version_t) (from FMI)
    - "mac_address": <mac address> (if available) (from NetX)
    - "lan_address": <lan_address> (if available) (from NetX)
- Parameters:
    - *mqtt_url*: MQTT broker
    - *mqtt_port*: MQTT port
    - *mqtt_user_id:* MQTT user ID (provided by docs)
    - *password*: API user password
    - *mqtt_project_id*: MQTT project ID (provided by docs)
    - *device_id*: Device identifier
    - *retry_limit*: The number of times to retry (recommend 5 times)
    - *retry_delay*: The time delay (in seconds) between retry (recommend 5 seconds)
    - *mqtt_heart_beat*: period (s) between mqtt keep alive signal (recommend 60 seconds)
    - *tls_enabled*: 1 for using TLS and 0 for using None TLS
- Errors:
    - M1_ERROR_INVALID_URL
    - M1_ERROR_UNABLE_TO_CONNECT
- Success:
    - return 0

void m1_disconnect()

- Description: Disconnects from Medium One MQTT broker

int m1_publish_event(char * json_payload, char * observed_at)

- Description: This sends a json event to Medium One.
- Parameters:
    - *json_payload*: serialized JSON object to send to Medium One
        - Example: "{\"my_key\": 12345}"
    - *observed_at*: ISO8601-format datetime string. If NULL, not included in event (*observed_at* generated by M1 broker on reception)
        - Example: "2016-08-29T16:03:12.794925-07:00"

- Errors:
  - M1_ERROR_NOT_CONNECTED
  - M1_ERROR_NULL_PAYLOAD
  - M1_ERROR_UNABLE_TO_PUBLISH
- Success:
  - return 0

int m1_register_subscription_callback(void (* subscription_callback)(int type, char * topic, char * msg, int length))

- Description: This function registers a callback function which is called when a message is received from Medium One.
- Parameters:
  - *subscription_callback*: function pointer to callback
  - Callback function parameters:
    - *type*: type of message received (1 - cloud to device; 2 - cloud to group)
    - *topic:* full topic identifier for received message (null-terminated string)
    - *msg*: message content
    - *length:* message length
- Errors:
  - M1_ERROR_NULL_CALLBACK

Error Codes

M1_ERROR_INVALID_URL = 1
M1_ERROR_UNABLE_TO_CONNECT = 2
M1_ERROR_UNABLE_TO_DISCONNECT = 3
M1_ERROR_NOT_CONNECTED = 4
M1_ERROR_NULL_PAYLOAD = 5
M1_ERROR_UNABLE_TO_PUBLISH = 6
M1_ERROR_NULL_CALLBACK = 7

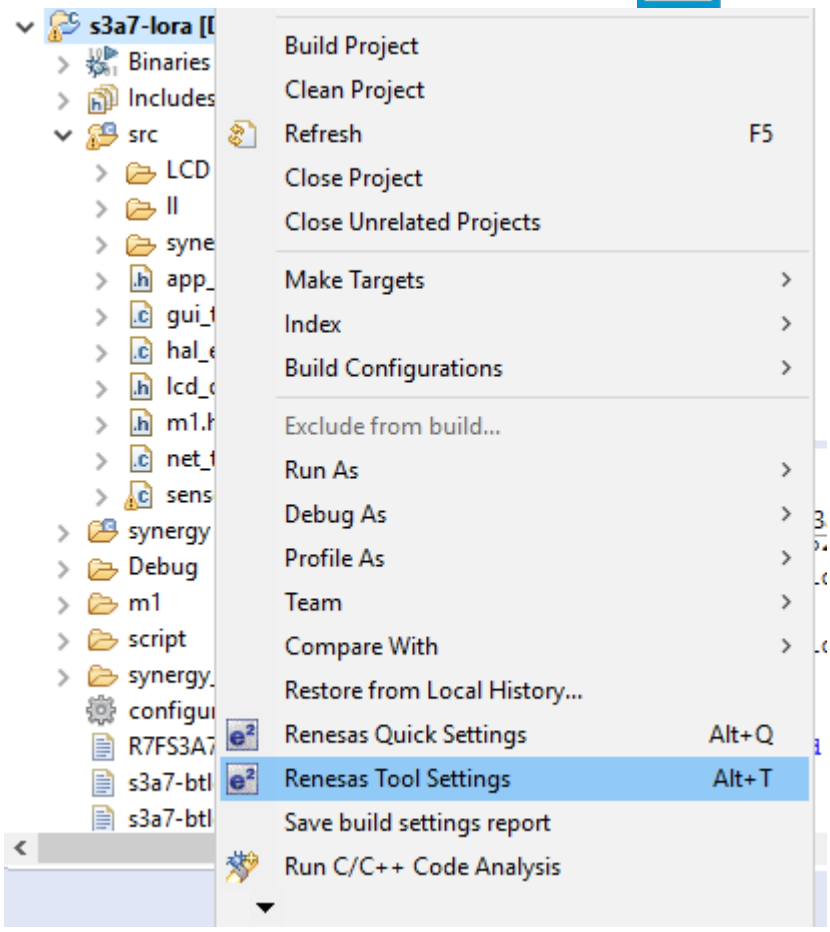
SSP Interoperability

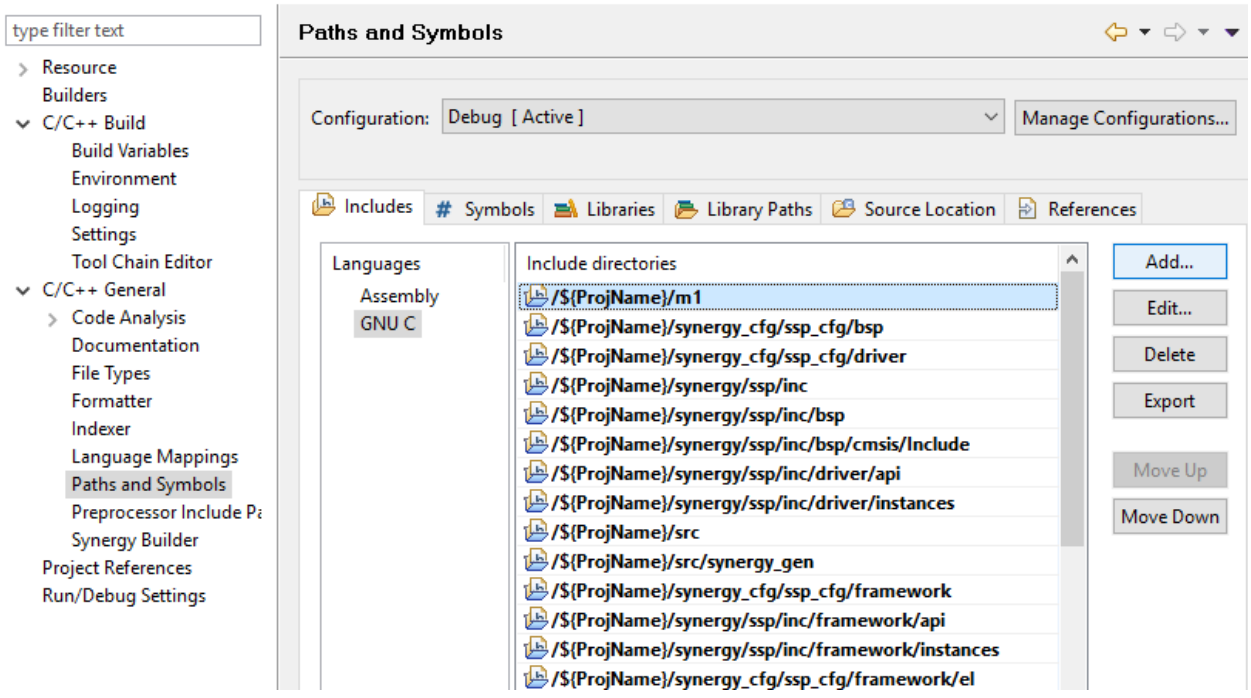
- SSP resources used:
  - ThreadX: 1 thread used, priority 15
    - Queues + mutexes: used for communication from APIs to M1 VSA
  - Factory MCU Information (FMI) Module: to collect identification information for diagnostic use
  - CypherBridge uMQTT resources (ThreadX, NetX usage, etc.)
    - NetX: nx, nx_dhcp, nx_dns, etc.
    - Not using NetX BSD socket interface
    - sf_el_nx (NetX Synergy Port) on SK, sf_wifi_gt202+sf_wifi_nsal_nx on S3A7

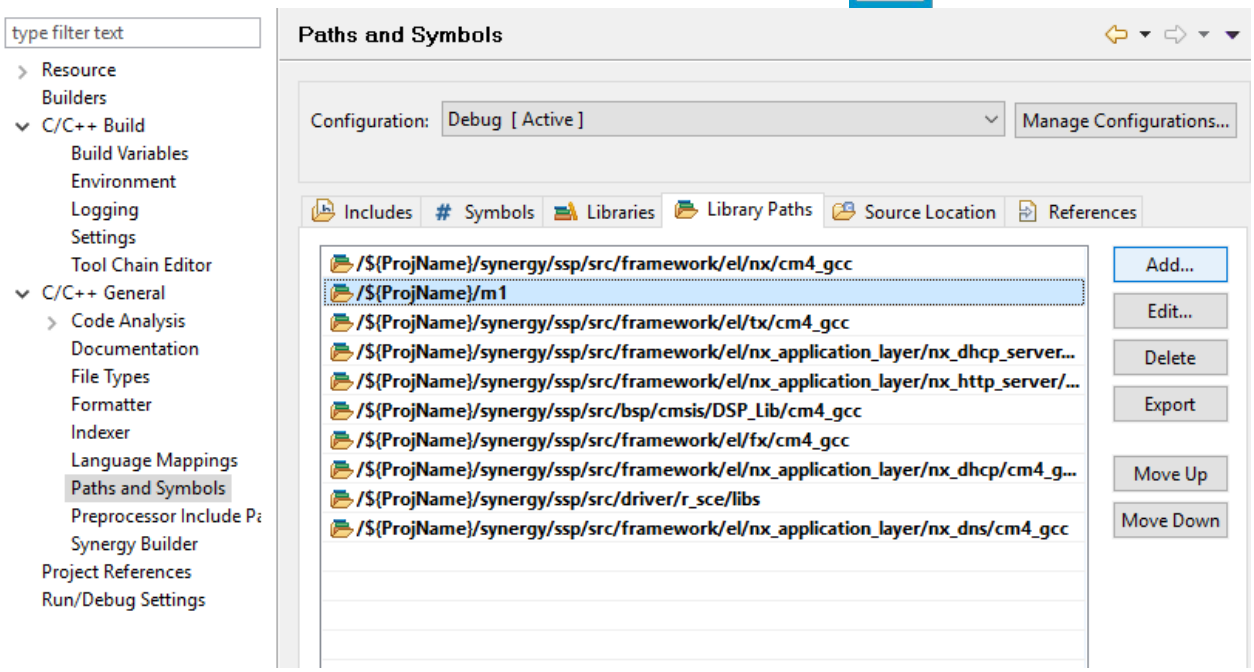
- SSP APIs used (to interface between the software add-on and the SSP):
    - ThreadX APIs
        - tx_thread_sleep/suspend/resume()
        - tx_queue_send/receive()
        - tx_mutex_get/put()
    - FMI
        - productInfoGet()
        - versionGet()
    - NetX
        - uMQTT usage (init, sockets, binding, packet pool, dhcp)

How To Install Medium One Cloud Agent On Your Synergy Project

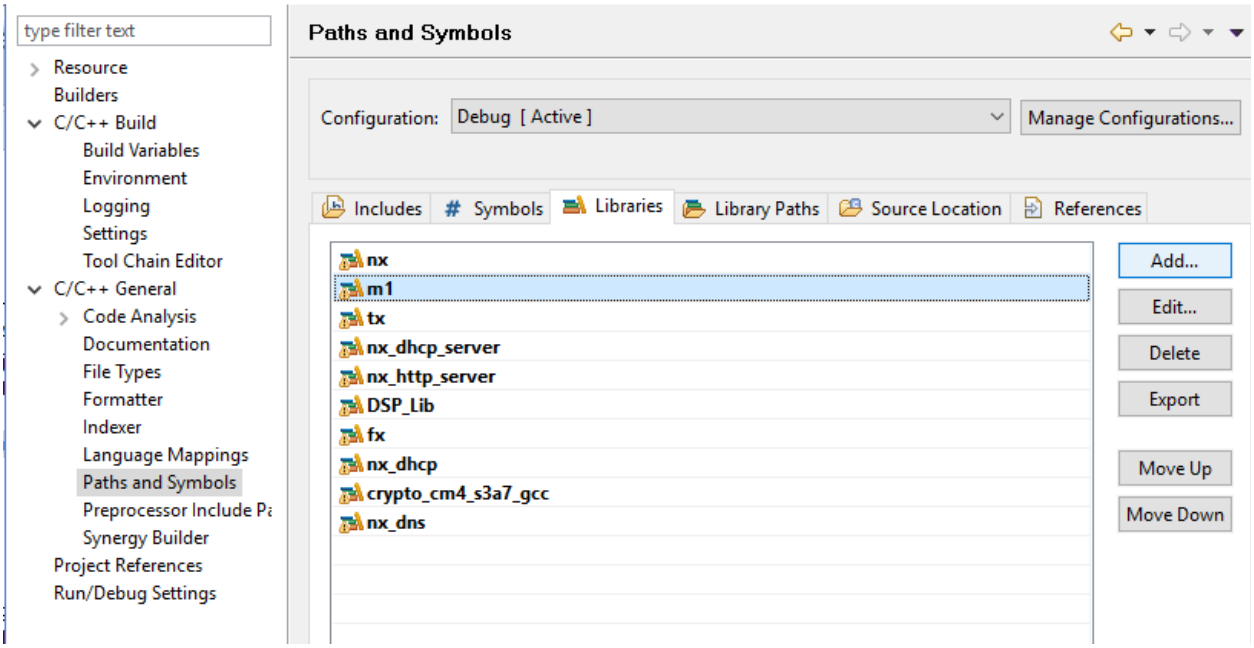
1. Download the Medium One Cloud Agent from the Renesas Synergy Gallery at synergygallery.renesas.com -> Software-Adds
2. Unzip the package.
3. Place the *m1* folder in your project.
4. Add the *m1* folder to your include and library paths.







5. Add *libm1* to the libraries list.



6. Open the Synergy Configuration for your project
7. On the Threads tab, for HAL/Common, add Driver -> Connectivity -> UART Driver on r_sci_uart
    a. Configure it appropriately for an available SCI. This UART port will transmit debug information regarding the MQTT connection.
8. On the Components tab, ensure that:
    a. nx_dns is selected
    b. r_fmi is selected

   c. r_sce is selected
   d. r_rtc is selected
9. Before calling *m1_connect()*, ensure that your application has setup an NX_IP instance named ***g_http_ip***, using a NX_PACKET_POOL named ***g_http_packet_pool***. All initialization, including resolving an IP address for the interface if necessary, must be done prior to calling *m1_connect()*.