cFS Basecamp
Remote Operations Guide

Version 1.8

July 2023

Introduction

- **Basecamp supports using the python GUI to interface with a cFS target that is running on a remote processor**
  - Basecamp also supports receiving telemetry from any remote application that complies with Basecamp's remote telemetry topic definition standard
- **The objectives of this tutorial are to**
  - Describe Basecamp's remote operations architecture and how the components that can be configured to remotely control a cFS target
  - Describe how a *"remote target process"* can be controlled by Basecamp's remote ops infrastructure
  - Describe how to configure a Raspberry Pi so Basecamp's *remote target process* tools are started during the Pi's boot sequence which allows Basecamp to communicate with a Pi that does not have a keyboard and screen attached
  - Describe how the architecture can be extended with new targets and *remote target process*

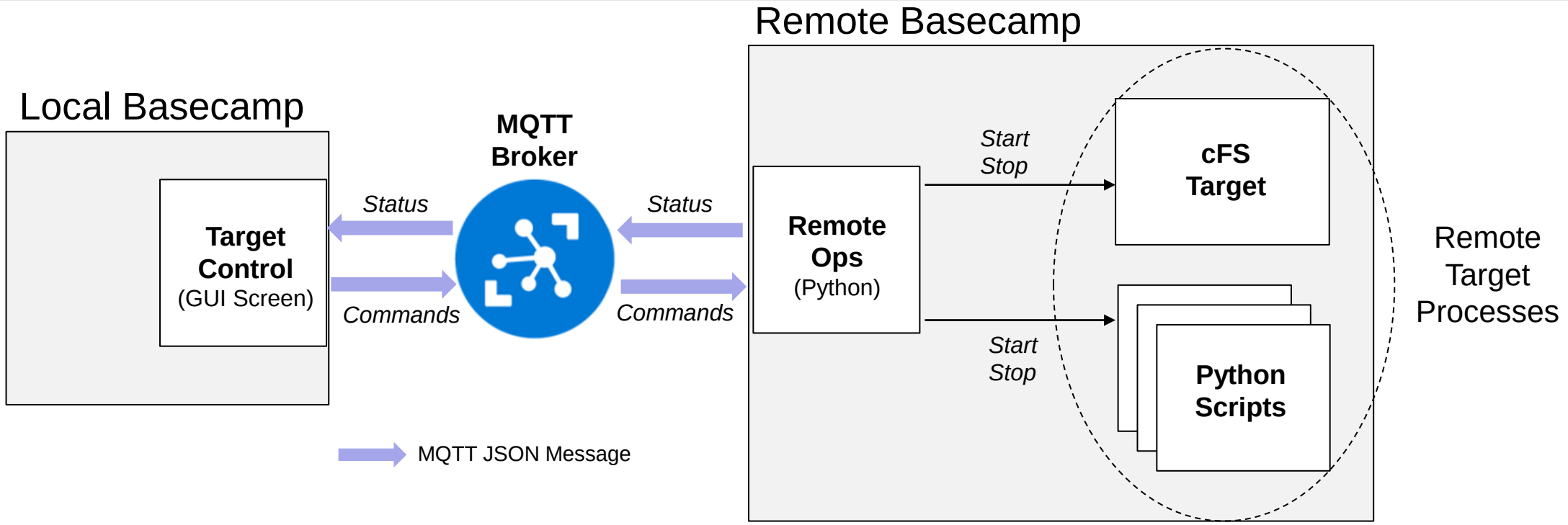
Notes:

1. Raspberry Pi is the only supported remote target in Basecamp v1.0
2. The MQTT Gateway (MQTT_GW) app docs directory has app-specific information. There's also an MQTT gateway inrtduction project project at **https://openmissionstack.com/projects_read/mqtt_gateway**

Terminology

- **Basecamp**
  - Refers to all the software components that are included in the cfs-basecamp git repo
  - Not all the components need to be running in every Basecamp instance so Basecamp on computer X could just be the cFS target running without a GUI. The GUI may have been used to add applications to the cFS target, but the GUI isn't required to run the cFS target.
- **Local Basecamp**
  - A desktop computer environment that has Basecamp installed
  - Users typically run the Basecamp GUI for command and telemetry
- **Remote Target Process**
  - A cFS target or a python script that can be started and stopped remotely by Basecamp's remote ops tool
  - A remote target process has telemetry that complies with one or more Basecamp remote telemetry topic definition standard
- **Remote Basecamp**
  - Any computer or processor board that is configured to have a *remote target process* controlled by a *local Basecamp*
- **MQTT**
  - *"MQTT is an OASIS standard messaging protocol for the Internet of Things (IoT). It is designed as an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth.",* https://mqtt.org/
- **MQTT Topic**
  - An UTF-8 string that an MQTT broker uses to filter messages for each connected client
- **Remote Telemetry Topic**
  - An MQTT Topic that is defined using a Basecamp standard topic definition structure
- **BinHex**
  - A binary encoding method where each 8-bit binary value is encoded using two hexadecimal text values

Remote Target Process Control (1 of 2)



- **Target control (screenshot on next slide) communicates to Basecamp's Remote Ops using an MQTT broker**
  - *Remote Ops* is a headless (no GUI) Python application
  - Multiple organizations provide free MQTT broker services such as EMQX https://www.emqx.com/en) and HiveMQ (https://www.hivemq.com/.hivemq.com)
- **Target Control and Remote Ops manage starting/stopping Remote Target Processes. The Remote Ops script does not report operational status for each Remote Target App after it is started**
  - Remote Ops telemetry provides status on Remote Target Process operational state
  - This status is not the telemetry from the Remote Target Process, this communication path is described in a later lesson

Remote Target Process Control (2 of 2)

- Automatically connects to the MQTT Broker defined in basecamp.ini

- Manual MQTT connect/disconnect
- Manage target platform, NOOP sends event message reported in status
- Start/stop a cFS remote target process
- Start/stop a Python remote target process**

- Remote IP address used for sending commands to the remote target process
- Telemetry message sequence counter; Incrementing value indicates good connection
- Count of commands received
- Report significant events, all commands generate an event
- Is remote cFS target executing?
- What apps were loaded by cFS startup script
- Is a Python target executing?
- List of running Pythons apps

** Multiple Python processes can run and are managed via a popup dialogue.


STEM
Open ware
cFS
Remote Target
“Hello World” Python Script

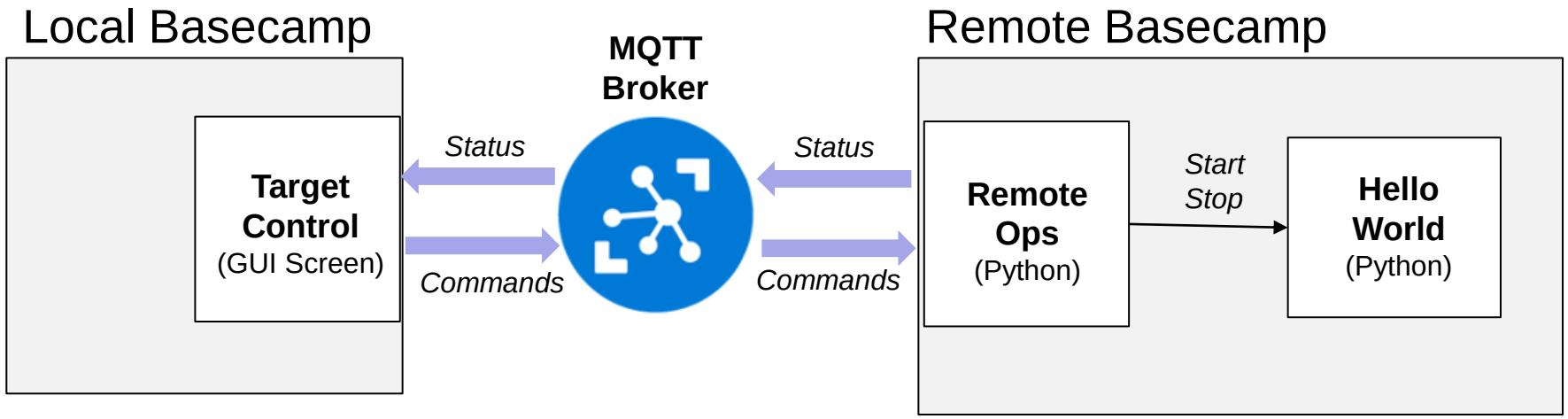
Agenda

1. Introduce MQTT how the Hello World remote python script uses MQTT
2. Run the ‘Hello World’ python script from the command line
3. Monitor the MQTT messages using a browser
4. Stop Hello World and start the Remote Ops script from the command line
5. Start Basecamp from another terminal window or on another computer
6. Start Hello World from Basecamp and monitor the telemetry

Notes:

1. You do not need two computers to work through this tutorial. The remote target can be configured in a second terminal window.
2. If you use two computers, then the remote system will either need a keyboard/terminal or SSH access

Remote Basecamp Source Files




```
cfs-basecamp
├─. . .
├─gnd-sys/app/
│  └─remote-ops/ . . . . . .  Top-level remote ops directory, one subdirectory for each remote target
│     │ ├─remoteops.py  . . .  Manage the MQTT interface for starting/stopping remote target processes
│     │ ├─remoteops.ini . . .  Remote ops configuration settings
│     ├─demo-target/  . . . .  Demonstration target
│     │ ├─helloworld.ini .  .  Configuration settings for the hello world python script
│     │ └─helloworld.py . . .  Python script that publishes an MQTT JSON message displayed by the Local Basecamp
│     ├─raspberry-pi/ . . . .  Raspberry Pi target
│     └─future targets/
└─. . .
```

Introduction to MQTT

- **Do not be concerned if you've never even heard of MQTT**
  - As a Basecamp user you will only need to understand some basic concepts that are explained in this tutorial
  - Basecamp uses MQTT because it is easy to learn, the de facto standard for the Internet of Things (IoT), large active open-source community, freely available MQTT brokers (messaging servers), its pub/sub model is compatible with the cFS, and it is scalable
- **MQTT is a publish/subscribe messaging protocol**
  - Message delivery is decoupled from the applications so they can "fire-and-forget"
- **MQTT topics uniquely identify (or address) messages**
  - Topics are text strings structured in a hierarchy like folders in a file system using a forward slash as a delimiter
  - An example topic from the IoT domain may be something like *myhome/groundfloor/livingroom/temperature*
- **MQTT clients publish topics containing their data and subscribe to topics containing data they need to process**
  - The local and remote basecamps are each MQTT clients
- **MQTT message payloads contain application-specific data**
  - Basecamp uses JSON payloads
- **MQTT message brokers deliver messages between clients**
  - Basecamp uses EMQX https://www.emqx.io as its default broker
  - EMQX has a browser hosted web socket that will be used in this tutorial to monitor basecamp messages
- **There's plenty of freely available MQTT learning resources**
  - For example, https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/

Hello World

- **Hello world publishes the MQTT topic *basecamp/demo/sc/rate* with the following JSON payload**

```
{"x": 0.130900, "y": 0.000000, "z": 0.000000}
```

- **It has a 1Hz control loop that sets one axis equal to a fixed rate of 18 deg/s for 5 seconds**
  - This rate results in a 90-degree rotation before changing the rate to a new axis
- **The MQTT configurations are defined in helloworld.ini**

```
[MQTT]
TARGET_ID   = demo/sc
SENSOR_ID   = rate
BROKER_ADDR = broker.emqx.io
BROKER_PORT = 1883
```

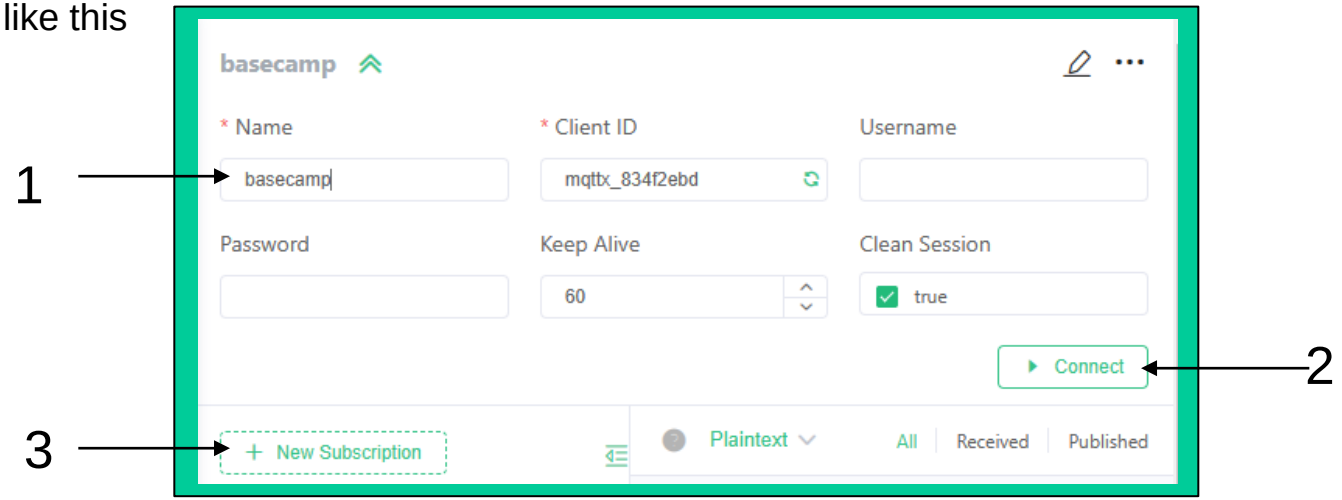
- **To run hello world, change your directory and issue the command as shown:**

```
Publishing telemetry basecamp/demo/sc/rate, {"x": 0.000000, "y": 0.000000, "z": 0.000000}
Publishing telemetry basecamp/demo/sc/rate, {"x": 0.000000, "y": 0.314159, "z": 0.000000}
Publishing telemetry basecamp/demo/sc/rate, {"x": 0.000000, "y": 0.314159, "z": 0.000000}
```

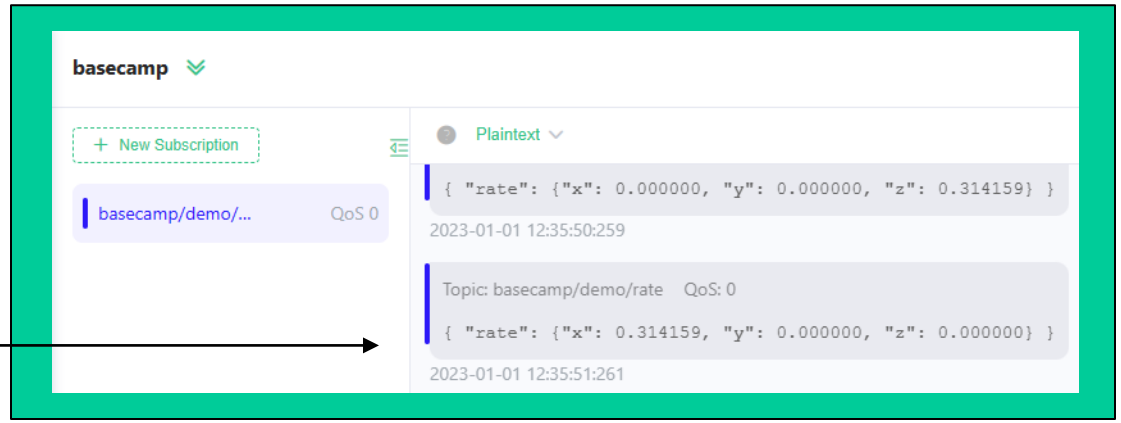
- **The hello world is connected as a client to the EMQX broker on port 1883**
- **The topic name is formed by concatenating with ini file definitions: basecamp/TARGET_ID/SENSOR_ID**
- **During the initial 5 seconds all axis rates are set to zero**

Hello World Telemetry (1 of 2)

- **EMQX provides a browser web socket where you can subscribe to hello world's topic**
  - Go to https://www.emqx.com/en/mqtt/public-mqtt5-broker and select the "Online MQTT Client" link
  - You will be directed to a page that looks like this

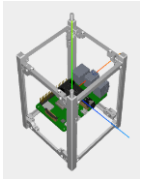


1. Enter a name, this example used "basecamp"
   - Client ID is supplied for you
2. Click "Connect"
3. Click "New Subscription" and enter "basecamp/demo/sc/rate" in the popup dialogue
   - You should see the rate messages displayed in the received window

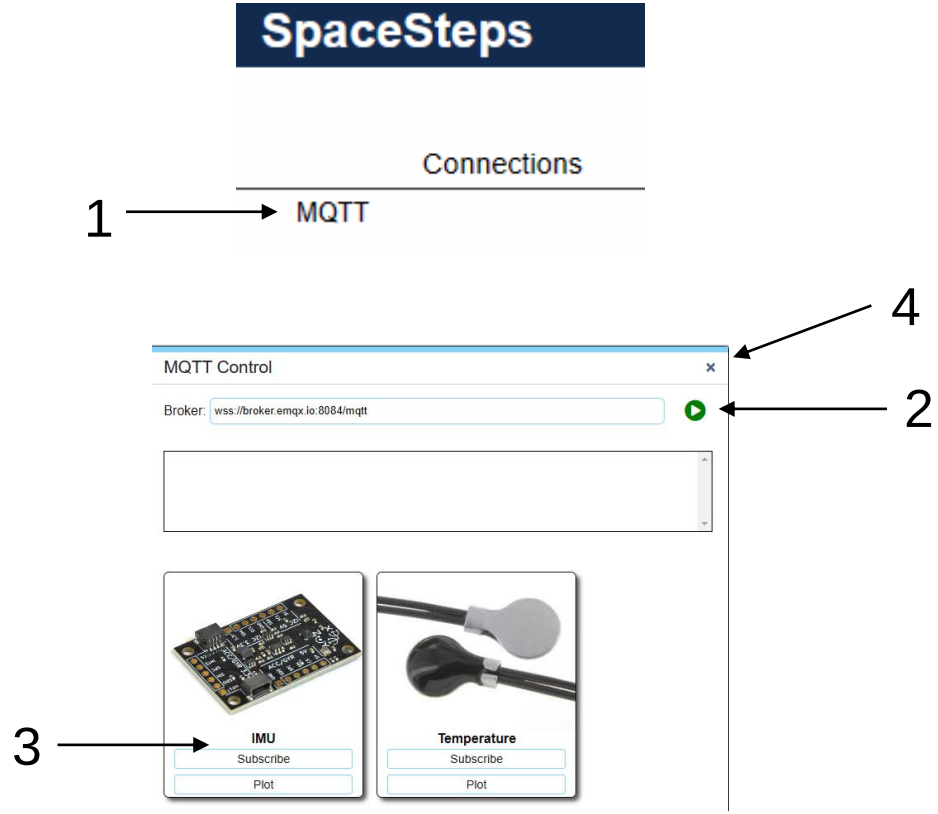


Hello World Telemetry (2 of 2)

- Space Steps https://spacesteps.com/ is a website that provides free space industry learning resources
- These resources include tools for visualizing some MQTT topics
- https://spacesteps.com/mqtt_receive can ingest hello world's basecamp/demo/sc/topic and apply the rates to the following image

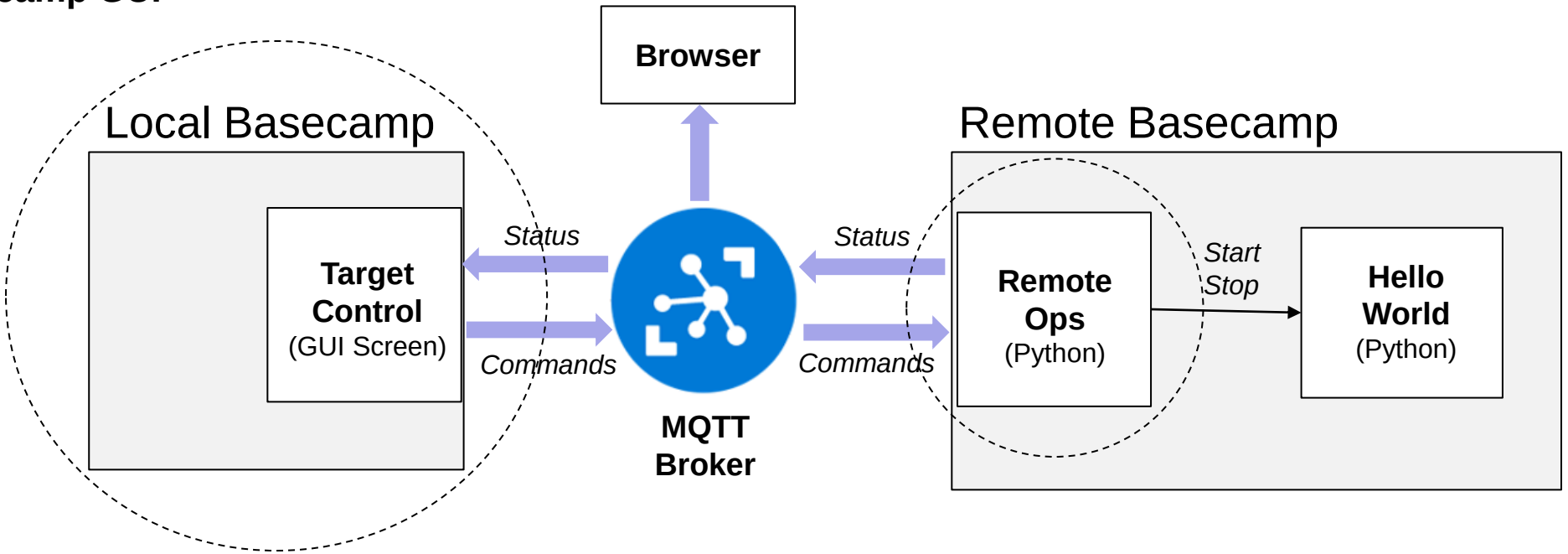


- After you go to the mqtt_receive site
  1. Select MQTT from the upper left which displays the MQTT Control window
  2. Select the play button to connect to the MQTT broker
  3. Subscribe to IMU data
  4. Close the window
  5. Start hello world from the command line and the CubeSat image will perform a sequence of rotations about each axis



Remotely Operate Hello World

- ~~In the previous slides, you ran Hello World from the command Line and observed the telemetry using a browser~~
- **In this section you will run the remoteops.py script from that command line and start/stop helloworld.py from a Basecamp GUI**




```
cfs-basecamp
├─. . .
├─gnd-sys/app/
│  └─remote-ops/ . . . . . .  Top-level remote ops directory, one subdirectory for each remote target
│     ├─remoteops.py . . . . .  Manage the MQTT interface for starting/stopping remote target processes
│     ├─remoteops.ini . . . . .  Remote ops configuration settings
│     ├─demo-target/ . . . . . .  Demonstration target
│     │  ├─helloworld.ini . . . . .  Configuration settings for the hello world python script
│     │  └─helloworld.py . . . . . .  Python script that publishes an MQTT JSON message displayed by the Local Basecamp
│     . . .
└─. . .
```

Remote Target Process Control

1. **On the Remote Basecamp (terminal window or physically separate computer), start the remote ops script**
   - Stop hello world if it is still running
   - Start remoteops.py as follows

```
osk@Open-STEMware:~/cfs-basecamp/gnd-sys/app/remoteops$ python3 remoteops.py
IP Adress: 172.25.0.255
Remote Ops defaults broker.emqx.io:1883//basecamp/demo
Client initialized on broker.emqx.io:1883
```

2. **The default remoteops.ini configurations should work to run hello world**

```
[MQTT]
TARGET_ID   = demo
BROKER_ADDR = broker.emqx.io
BROKER_PORT = 1883
```

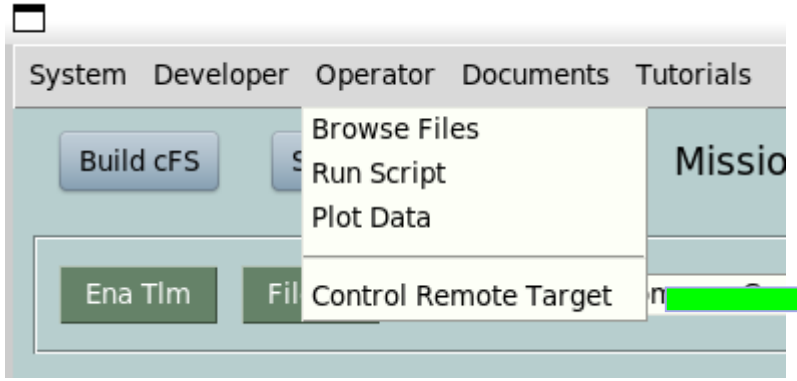
- **TARGET_ID is added to a 'basecamp/' prefix to form the MQTT client ID as well as the root of the MQTT topic**
- **When hello world is run it will concatenate 'rate' to the MQTT topic**

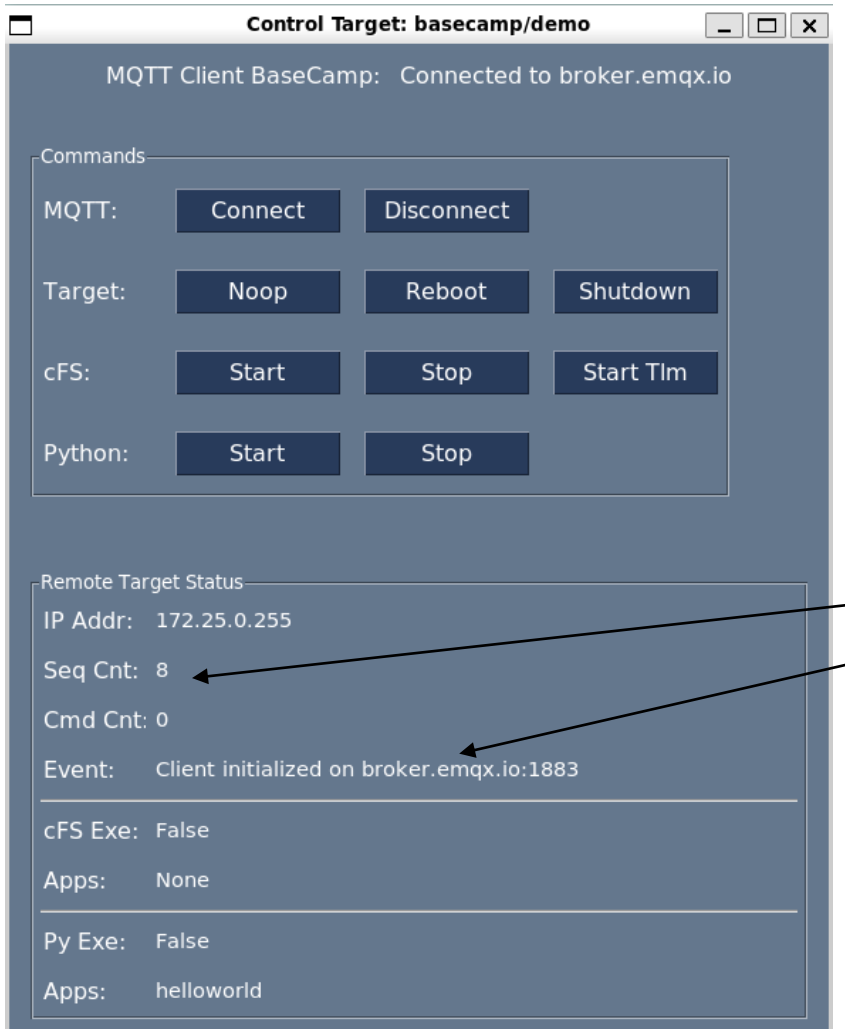
```
[APPS]
PYTHON_PATH = demo-target
PYTHON_APPS = helloworld
```

- **PYTHON_PATH identifies the remote target directory**
- **PYTHON_APPs is a comma separated list of available python apps that can be started and stopped**

Control Target Screen

1. **On the Local Basecamp start the Basecamp python GUI**
   a. From the menu select Operator->Control Remote Target



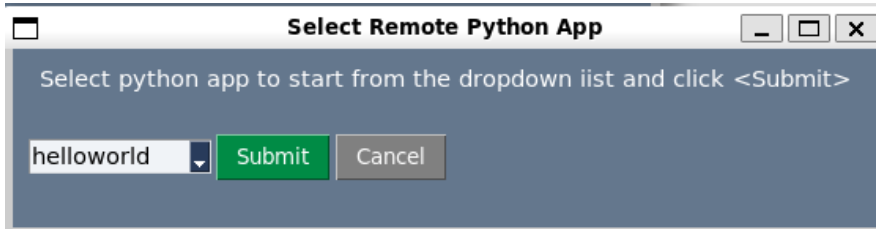


The remoteops.py script sends an MQTT message that contains this data

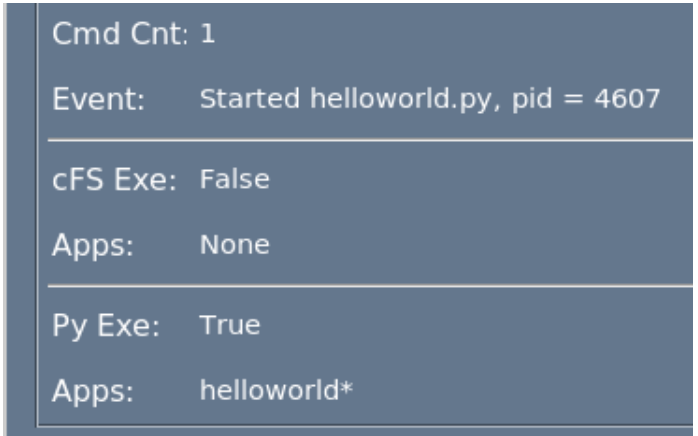
The sequence count should increment and an initialization event message

Start Hello World

1. **Select Start on the Python row to launch the following dialogue window**
   - The dropdown menu contains the apps listed in remoteops.ini

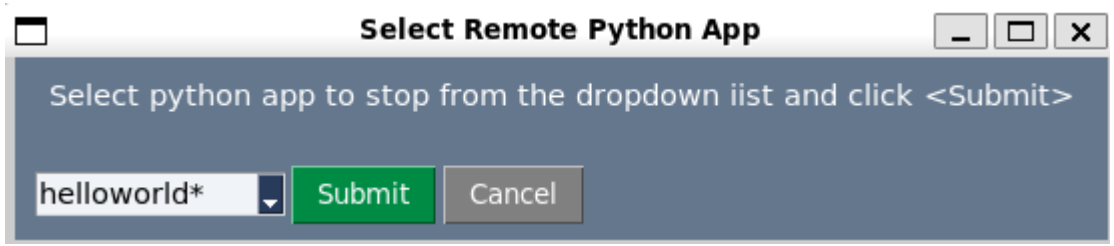


2. **Click <Submit> to start hello world**
   - The command counter will increment, and a success event message will be displayed
   - Python executing will be set to true and an asterisk will appear next to the python app that is executing

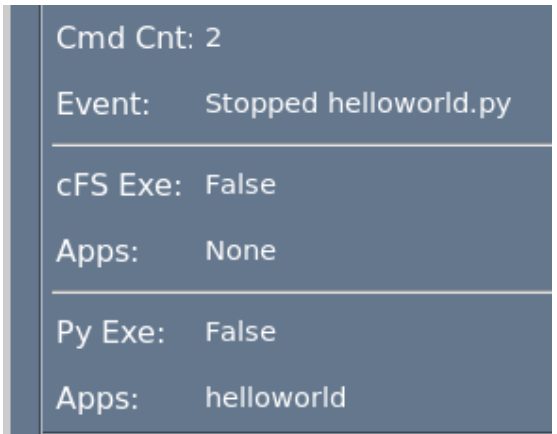


Stop Hello World

1. **Select Stop on the Python row to launch the following dialogue window**
   - The dropdown menu contains python apps that are running



2. **Click <Submit> to stop hello world**
   - The command counter will increment, and a success event message will be displayed
   - Python executing will be set to false and the asterisk wil be removed from helloworld



Closing Remarks

- **Multiple remote target Python apps can be defined**
  - It's the user's responsible to ensure multiple remote target applications can run concurrently
- **Remotes Ops also supports halting and rebooting the remote target**
  - The commands to perform these functions are target dependent so they are defined in remoteops.ini' 'EXEC' section
- **Basecamp can display an MQTT topic's payload in a telemetry screen**
  - This is accomplished using the cFS MQTT Gateway app
  - This is covered in a later lesson


STEM
Open ware
cFS
Configure & Control
Remote cFS Target

Introduction

Objectives

- Describe how to configure and control a remote cFS target
- Describe the different options for the Local Basecamp to receive Telemetry from the remote

Introduction

In the previous lesson a single python script with no commands and a single MQTT telemetry message was the remote target process being controlled. In this lesson a cFS target with a suite of apps is being controlled. MQTT is used for commands/telemetry and the MQTT Gateway app (MQTT_GW) makes this possible.

Approach

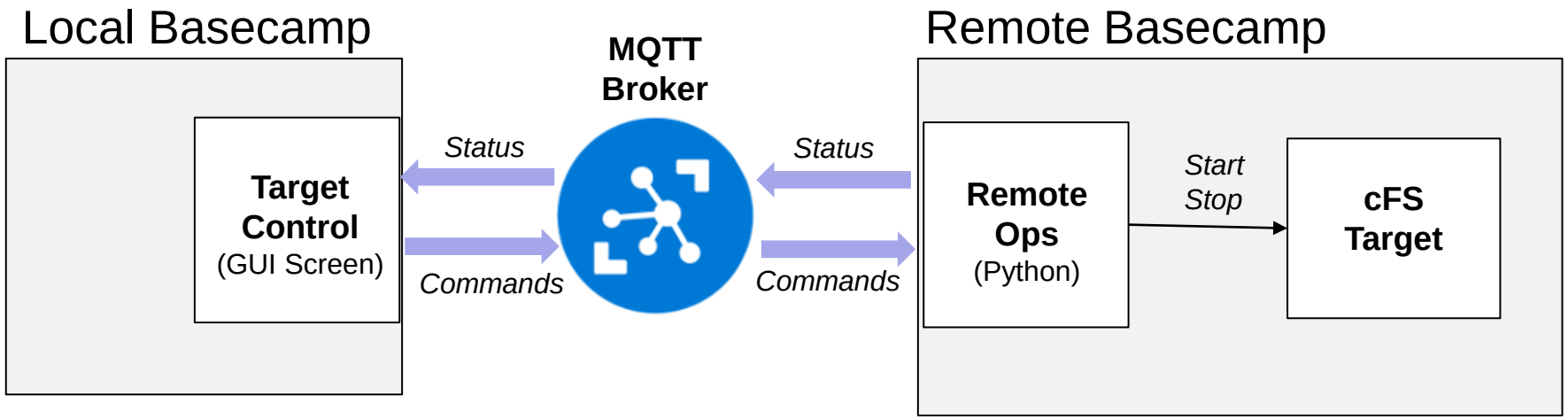
1. Add MQTT library (MQTT_LIB) and MQTT Gateway app (MQTT_GW) to the remote Basecamp cFS

TODO

- **Notes when running gpio_demo remotely**

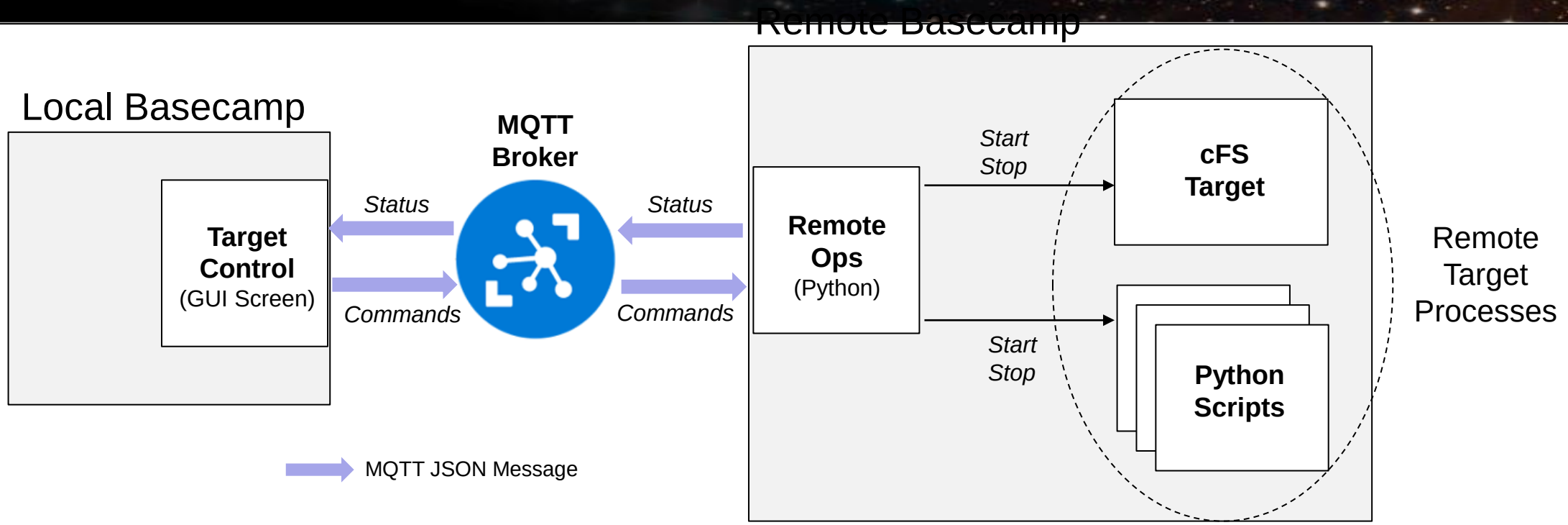
1. **Create local EDS python libraries**
   a. Host must be able to build the remote system for all of the components with EDS specs to generate the library
   b. Python libraries are object files native to the remote system so you can't simply copy them from the remote target to the host
2. **Start/stop remote target**
   a. Direct
      i. Have access (monitor/keyboard, ssh, etc.) to start the remote cFS target from the command line
   b. Basecamp Remote Ops
      i. The remote target must be running Basecamp's remote ops
      ii. Local Basecamp GND IP should be set to 127.0.0.1
3. **Remote target commanding**
   a. Direct
      i. Set basecamp.ini CFS_IP_ADDR to remote target IP
      ii. Issue commands from GUI
   b. MQTT
      i. @@Need to enhance CI to provide a wrapped message output that MQTT can subscribe to
      ii. Add MQTT library and app
      iii. Configure
4. **Start telemetry**

Remote Basecamp Source Files




```
cfs-basecamp
├-. . .
├-gnd-sys/
|  └-remote-ops/ . . . . . Top-level remote ops directory, one subdirectory for each remote target
|     | ├-remoteops.py  . . . Manage the MQTT interface for starting/stopping remote target processes
|     | ├-remoteops.ini . . . Remote ops configuration settings
|     └-. . . targets
├-usr/
|  └-apps/
|     ├-mqtt_gw/  .     . . . . Manage the MQTT interface for starting/stopping remote target processes
|     └-mqtt_lib/  . . . . Manage the MQTT interface for starting/stopping remote target processes
└-. . .
```

Remote Target Process Control



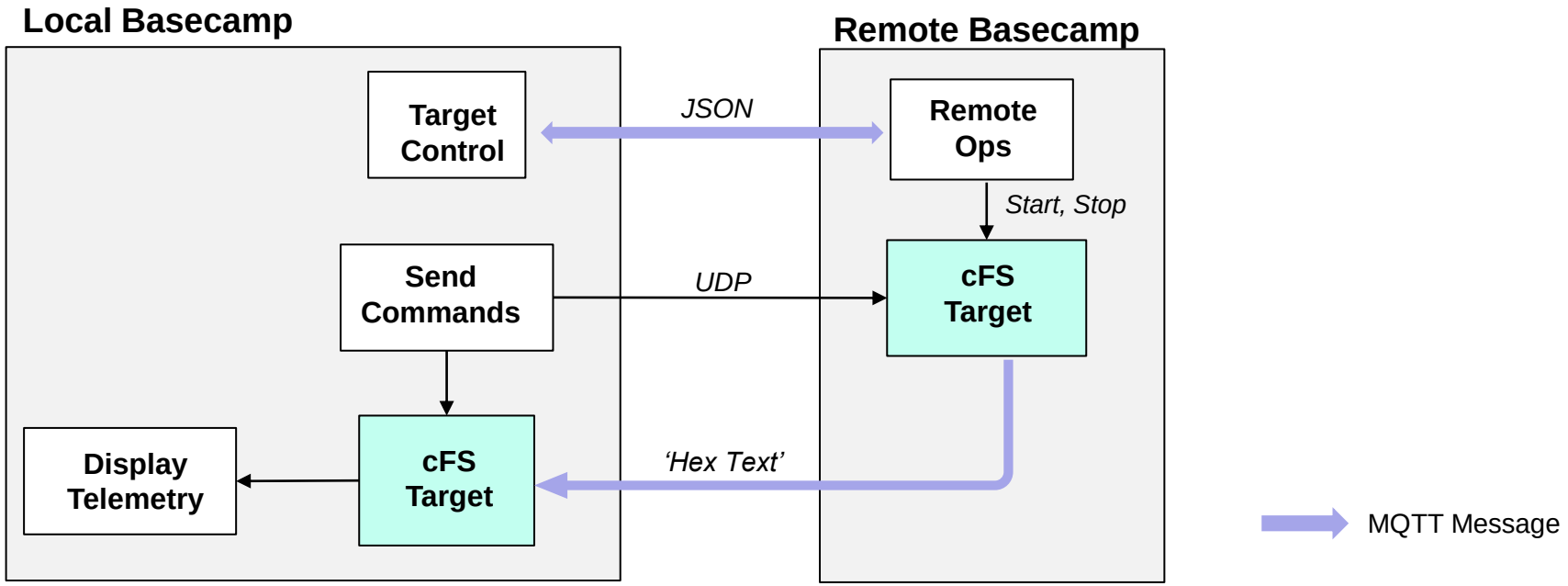
- **Multiple remote target Python apps can be defined**

  Multiple organizations provide free MQTT broker services

  If your operational environment allows access to one of these free brokers, you can use Basecamp's remote operations

  - This is useful when you have separate Python scripts for different hardware components
  - The user is responsible for ensuring multiple remote target applications can run concurrently
- **The Remote Ops script does <u>not</u> report operational status for each Remote Target App after it is started**
  - This should

Remote cFS Operations (1 of 2)

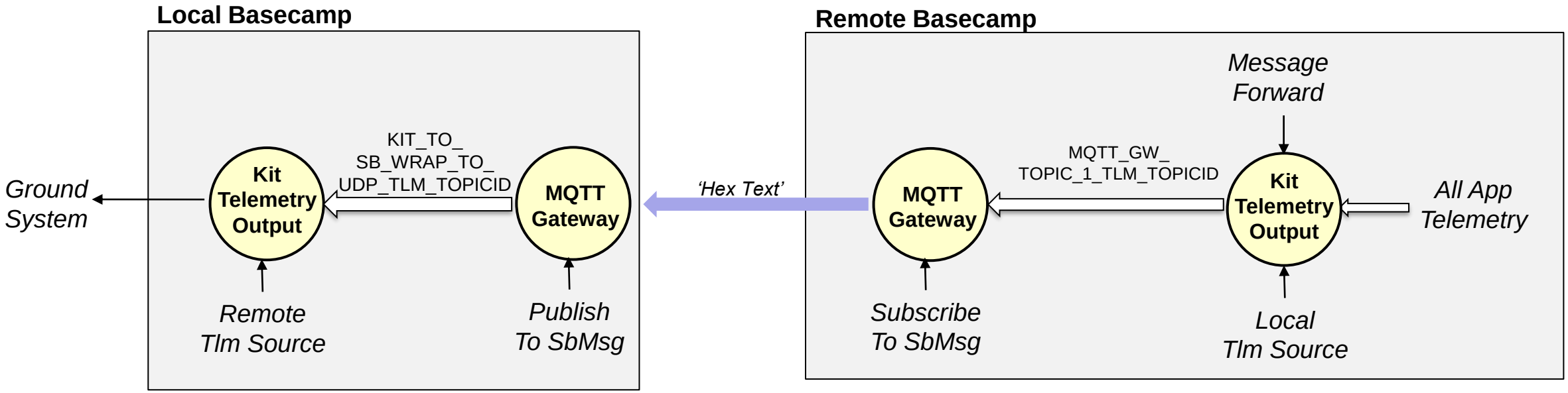


- **The same EDS versions must be used on both systems**
  - This is useful when you have separate Python scripts for different hardware components
  - The user is responsible for ensuring multiple remote target applications can run concurrently
- **The Remote Ops script does <u>not</u> report operational status for each Remote Target App after it is started**
  - This should
- **Remotes Ops supports other functions like rebooting the remote target**

Remote cFS Operations (2 of 2)

- Commanding
  - Assumes ability to use UDP directly to
  - Commanding hex text support
- Telemetry requires the MQTT_GW app and KIT_TO features that described later
  - Install MQTT_LIB and MQTT_GW
  - Modify MQTT Topic table to subscribe to sbmsg
  - Modify MQTT ini table to have a different client name
  - Modify KIT_TO ini table to forward packets
- Describe switch sequence
  - Set local KIT_TO to remote
  - Set command IP address
  - Send enable telemetry command to remote cFS KIT_TO

MQTT Hex Text Telemetry



- KIT_TO table identifies which packets to wrap and forward to MQTT_GW
- Local KIT_TO configured in remote and remote KIT_TO is configured as local which doesn't really matter

Remote Basecamp cFS Setup

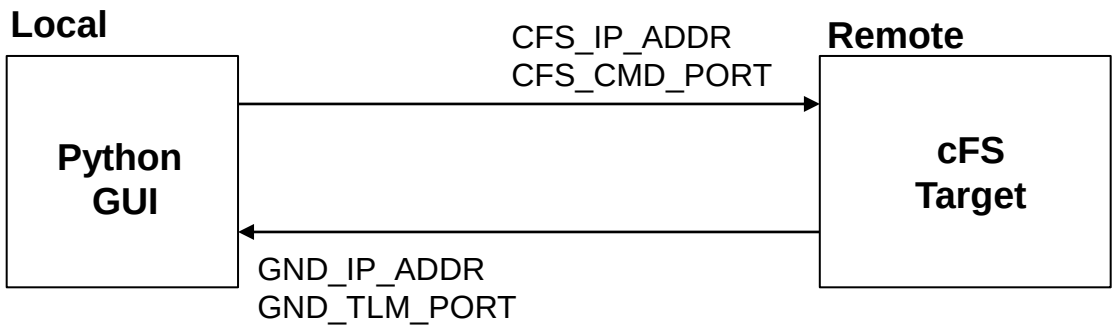
1. **Install Basecamp following the instructions at https://github.com/cfs-tools/cfs-basecamp**
   - If you are working on Raspberry Pi you must follow the instructions in Tutorial #5
2. **Start the Basecamp GUI and clone/install MQTT_LIB and MQTT_GW using the tools in the Developer menu**
3. **Edit MQTT_GW's topic table, cfe-eds-framework/basecamp_defs/cpu1_mqtt_gw_ini.json**
   - Change "MQTT_CLIENT_NAME" so it is different than the default name used by the local Basecamp
4. **Edit MQTT_GW's topic table, cfe-eds-framework/basecamp_defs/cpu1_mqtt_gw_topics.json,**
   - Set topic cfe-1 sbmsg's "sb-role" to "sub" which instructs MQTT_GW to
     1. Subscribe to MQTT_GW_TOPIC_1_TLM
     2. Unwrap the SB messages that are contained in the MQTT_GW_TOPIC_1_TLM SB payload
     3. Encode the unwrapped SB message in hex text and publish to an MQTT broker in a basecamp/sbmsg topic
5. **Edit KIT_TO's packet table cfe-eds-framework/basecamp_defs/cpu1_kit_to_pkt.json**
   - Set "forward" to true for each packet you want forwarded to your local Basecamp
   - KIT_TO copies forwarded messages into MQTT_GW_TOPIC_1_TLM's payload and publishes them on the SB
   - Minimum: ES HK for time, EVS long messages
6. **Run "make install" to install the modified tables**

Directory & File Highlights

```
cfs-basecamp
├─apps/
│
├─cfe-eds-framework/
│
├─gnd-sys/
│  └─remote-ops/ . . . . . . . . . . . . . Top-level directory, one subdirectory for each remote target
│     │ ├─remoteops.py  . . . . . . . . . . Manage the MQTT interface for starting/stopping remote target applications
│     │ ├─remoteops.ini . . . . . . . . . . Remote ops configuration settings
│     ├─raspberry-pi/
│     │ ├─aareadme.txt  . . . . . . . . . . Provides detailed installation instructions for the Pi Control Service
│     │ └─picontrol.service . . . . . . . . Systemd file used to start the remoteops.py script during the Pi boot process
│     └─future targets/
│
└─usr/
```

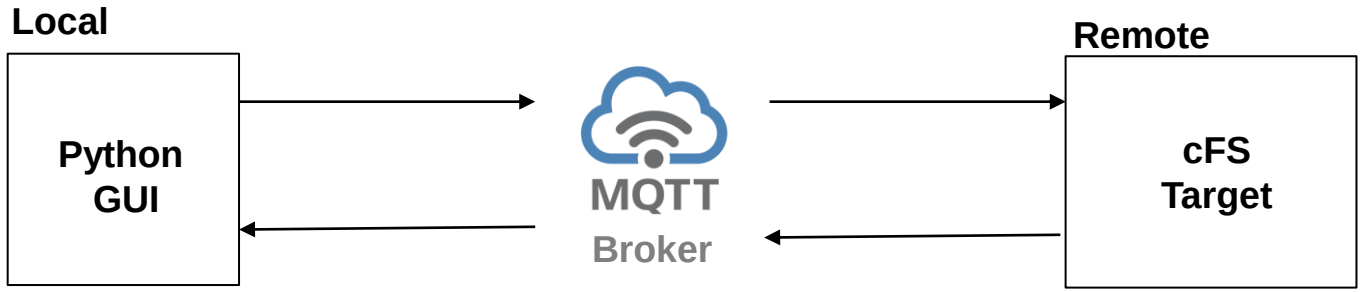
Remote Command & Telemetry

Direct Remote Command & Telemetry



1. Build same cFS target on Local and Remote to generate the same Python libraries
2.

MQTT Remote Command & Telemetry



GND_IP_ADDR
GND_TLM_PORT

CFS_IP_ADDR
CFS_CMD_PORT

1. Build same cFS target on Local and Remote to generate the same Python libraries
2. Local
   - Set commad to MQTT remote
   - Set telemetry to remote
   - Set MQTT telemetry to pub
   - Set KIT_TO to sub
3. Remote
   - Set KIT_TO to packet forward
   - Set MQTT command to pub
   - Set MQTT telemetry to sub
   - Set KIT_TO to pub

TODO

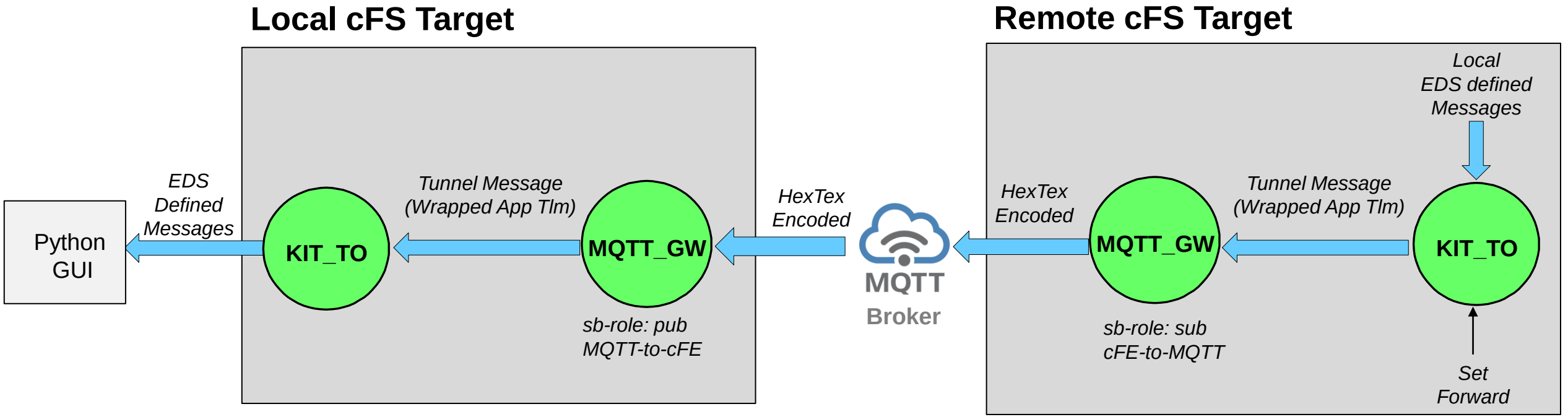
- **Describe how to run Basecamp's Raspberry Pi Python and the cFS GPIO demo in a remote ops configuration**

- **Notes when running gpio_demo remotely (June 2023)**

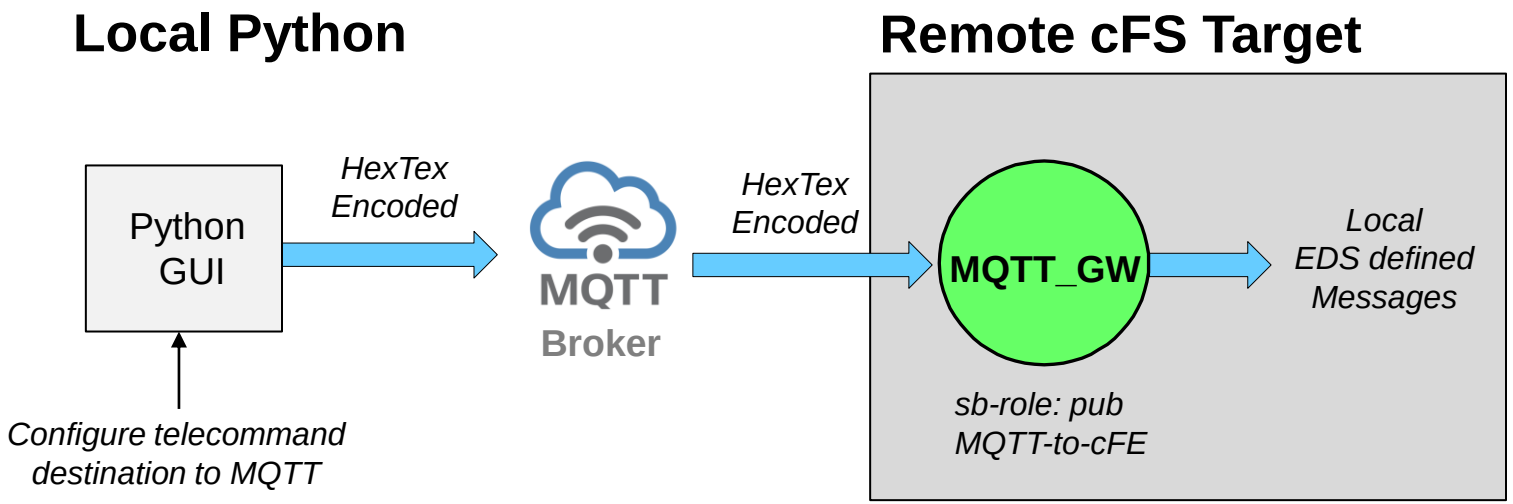
1. Create local EDS python libraries
   a. Host must be able to build the remote system for all of the components with EDS specs to generate the library
   b. Python libraries are object files native to the remote system so you can't simply copy them from the remote target to the host
2. Start/stop remote target
   a. Direct
      i. Have access (monitor/keyboard, ssh, etc.) to start the remote cFS target from the command line
   b. Basecamp Remote Ops
      i. The remote target must be running Basecamp's remote ops
      ii. Local Basecamp GND IP should be set to 127.0.0.1
3. Remote target commanding
   a. Direct
      i. Set basecamp.ini CFS_IP_ADDR to remote target IP
      ii. Issue commands from GUI
   b. MQTT
      i. @@Need to enhance CI to provide a wrapped message output that MQTT can subscribe to
      ii. Add MQTT library and app
      iii. Configure
4. Start telemetry
   a.

Can plugin have an ini file?
Ini files specify dependencies
EDS defines Interfaces

Remote-to-Local HexTex Telemetry



Local-to-Remote HexTex Telecommand

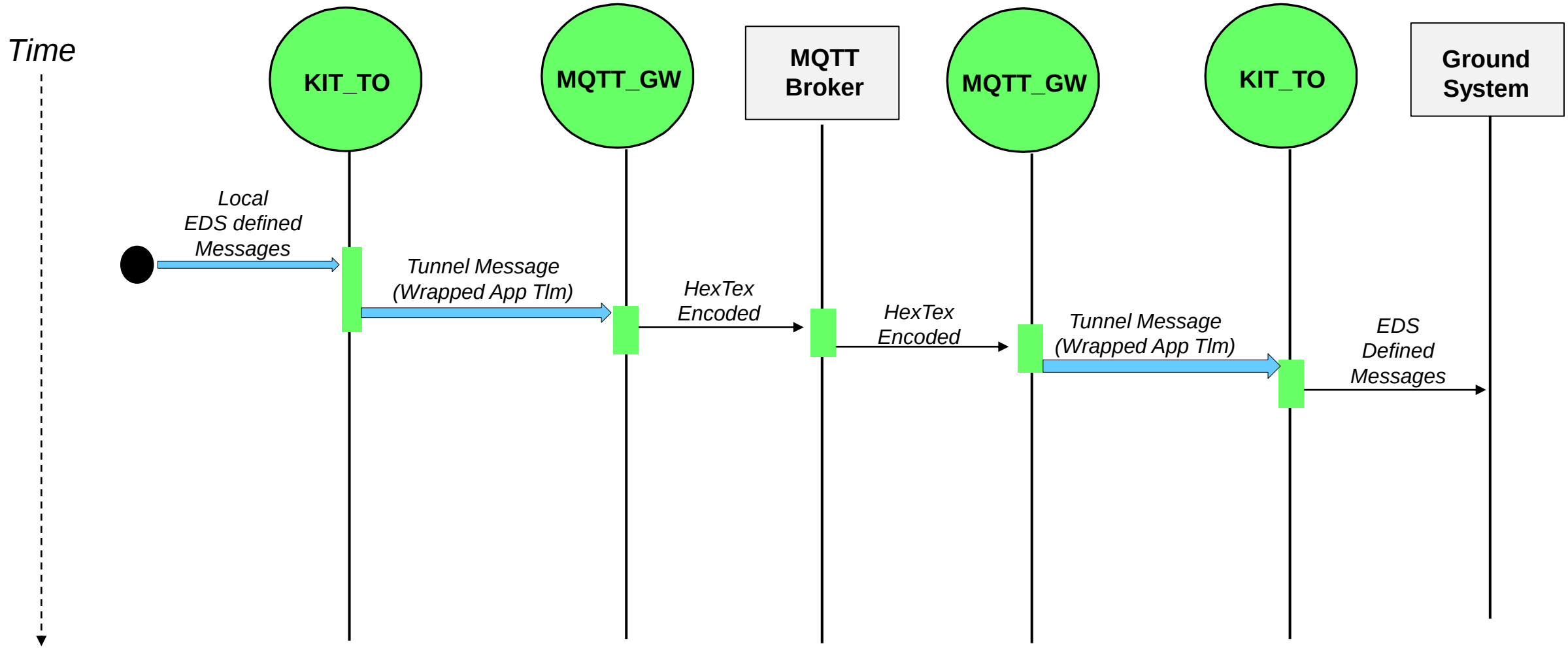


- MQTT broker telecommand destination configuration status:

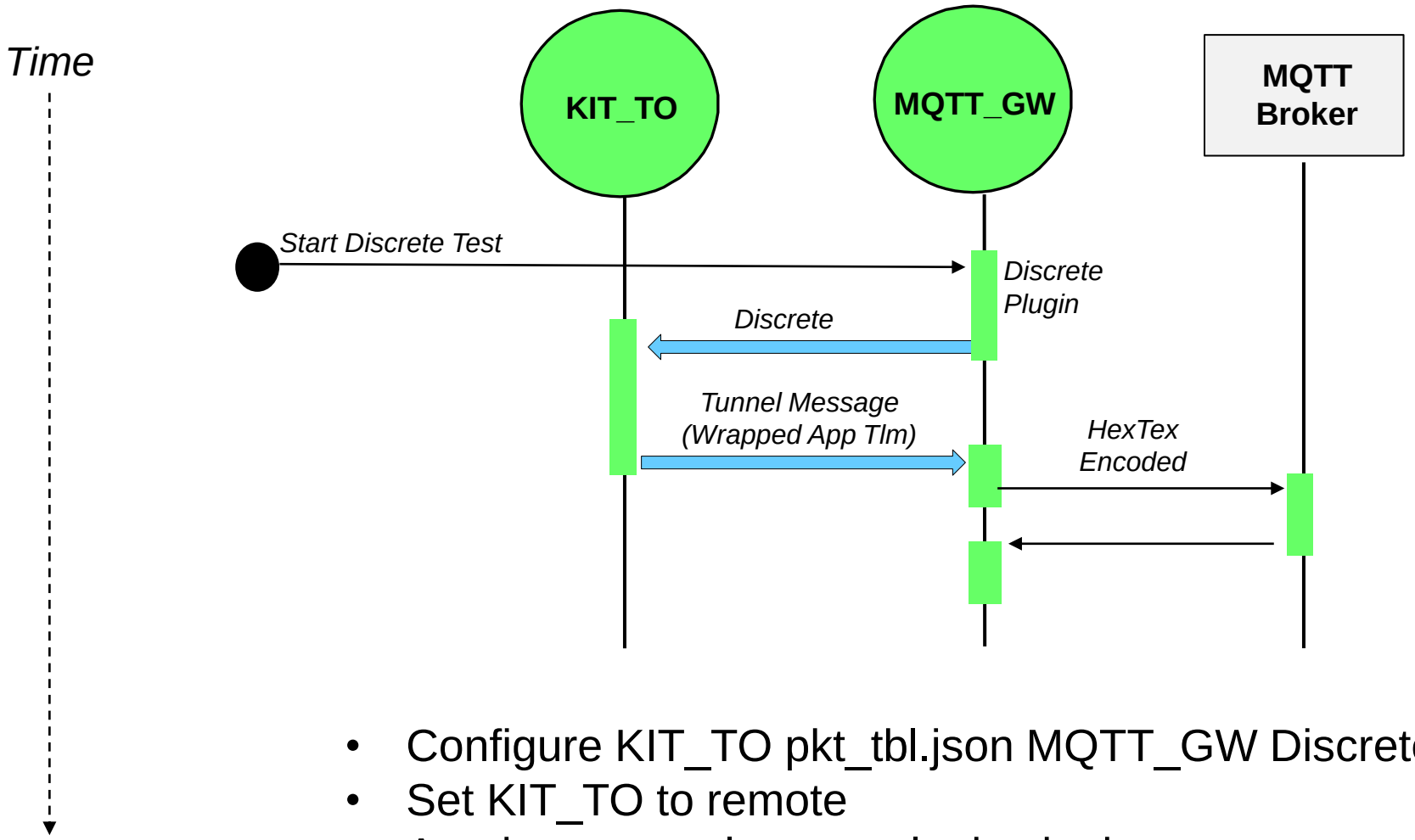
```
Telecommand: broker.hivemq.com:1883/basecamp/cfs/cmd
```

- Remote MQTT_GW command plugin performs same EDS functions as Command Ingest

MQTT Hex Text Telemetry



MQTT Telemetry Plugin Test



- Configure KIT_TO pkt_tbl.json MQTT_GW Discrete message to forward
- Set KIT_TO to remote
- Are there two telemetry plugins in the same system?

Start Event Software Bus Message Internet

- **Multiple remote target Python apps can be defined**
  - The user is responsible for ensuring multiple remote target applications can run concurrently
- **Remotes Ops supports other functions like rebooting the remote target**

Basecamp can be configured to start and stop remote target applications

Remote target applications include cFS targets and Python scripts

a INI file and broker definition

This is useful when you have separate Python scripts for different hardware components

```
cfs-basecamp
├-. . .
├-gnd-sys/
|  └-remote-ops/ . . . . . Top-level remote ops directory, one subdirectory for each remote target
|     | ├-remoteops.py  . . . . . Manage the MQTT interface for starting/stopping remote target processes
|     | ├-remoteops.ini . . . . . Remote ops configuration settings
|     ├-demo-target/  . . . . . Demonstration target
|     | ├-helloworld.ini .  . . . . . Configuration settings for the hello world python script
|     | └-helloworld.py . . . . . Python script that publishes an MQTT JSON message displayed by the Local Basecamp
|     ├-raspberry-pi/ . . . . . Raspberry Pi target
|     | ├-aareadme.txt  . . . . . Provides detailed installation instructions for the Pi Control Service
|     | ├-adafruitimu.py  . . . . . RTP that reads Inertial Measurement Unit (IMU) data and publish it in a MQTT message
|     | ├-discretedemo.py  . . . . . RTP that controls an LED and reports the LED state in a MQTT message
|     | └-picontrol.service . . . . . Systemd file used to start the remoteops.py script during the Pi boot process
|     └-future targets/
└-
```

Adding a Remote Target Process

- **Describe how to add remote target processes**

Notes:

1. Refer to tutorial TBD if you would like to create a Basecamp tutorial for your remote ops processes

- **Next steps**
  - Add your own targets and RTPs