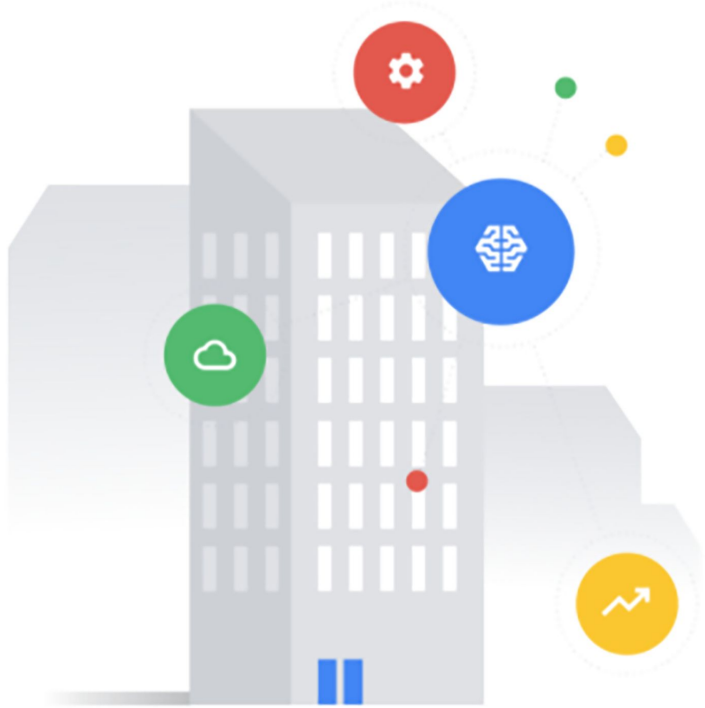
Google

Lesson 6

Sequences



Before you get started

This onboarding deck has interactive features and activities that enable a self-guided learning experience. To help you get started, here are two tips for viewing and navigating through the deck.

1 View this deck in presentation mode.

- To enter presentation mode, you can either:
  - Click the **Present** or **Slideshow** button in the top-right corner of this page.
  - Press **Ctrl+F5** (Windows), **Cmd+Enter** (macOS), or **Ctrl+Search+5** (Chrome OS) on your keyboard.
- To exit presentation mode, press the **Esc** key on your keyboard.

2 Navigate by clicking the buttons and links.

- Click the **Back** or **Next** buttons to go backward or forward in the deck. Moving forward, you'll find them in the bottom corners of every slide.
- Click blue text to go to another slide in this deck or open a new page in your browser.
- For the best learning experience, using your keyboard or mouse wheel to navigate is discouraged.

Ready to get started? Let's go!

Lesson 6

Sequences

What you'll learn about:

- Advanced device features with bidirectional exchange

By the end of this lesson, you'll be able to:

- Run sequence tester against an individual pubber device.
- Examine result messages from a sequence test.
- Run sequence tester against pubber with a bad serial number.
- Run and debug an individual sequence test.

Back

Next

Google

What are sequences?

Sequences are defined device behaviors which are formulated from a series of events or communication messages.

Sequences are forms of more advanced devices, and are considered bidirectional exchanges.

They define how a particular device should behave and represent a conversation with the device, asking it questions, telling it information, and expecting certain responses.

Different subsections define different bits of functionality. For example, a suite of writeback sequences details how a device writes a data point to a device from the cloud.

Back

Next

Google

Sequences in UDMI

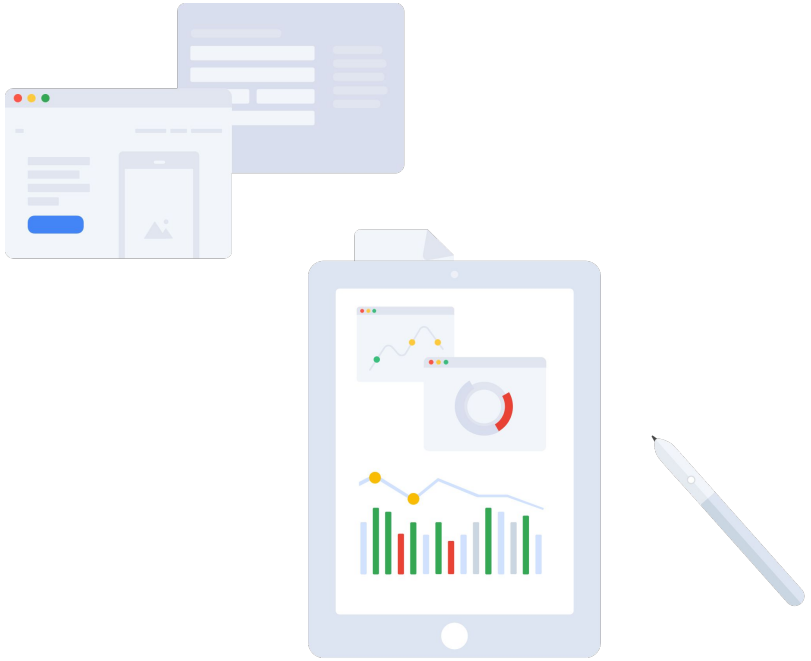
There are a number of defined sequences in UDMI.

Some of those defined sequences are:

- Config and State
- Discovery
- Writeback

For specific details on these sequences, refer to this link.

Let's look at a few example sequences. Each example will provide a general overview of how a device and the cloud work together.



Back

Next

Google

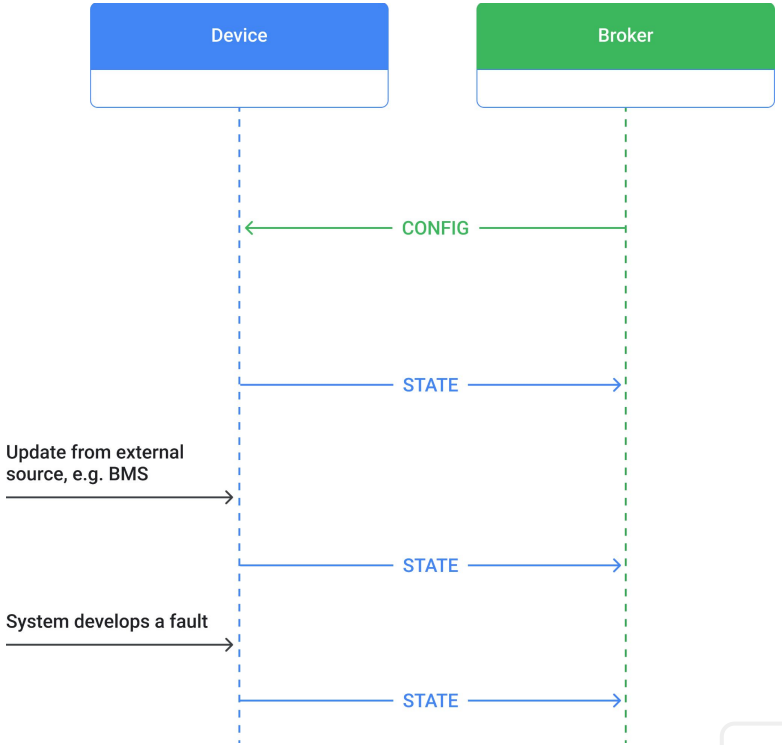
Sequences in UDMI (continued)

Last Updated Config

This sequence defines how `state` and `config` messages work together to represent a transactional state between the cloud and device.

In its simplest form, this sequence displays:

1. The device first receiving the config.
2. The device then parsing the config.
3. Then, if the config is valid, the device accepts the config and sends a state message with the value of **`last_update`** matching the timestamp of the config message.



Back

Next

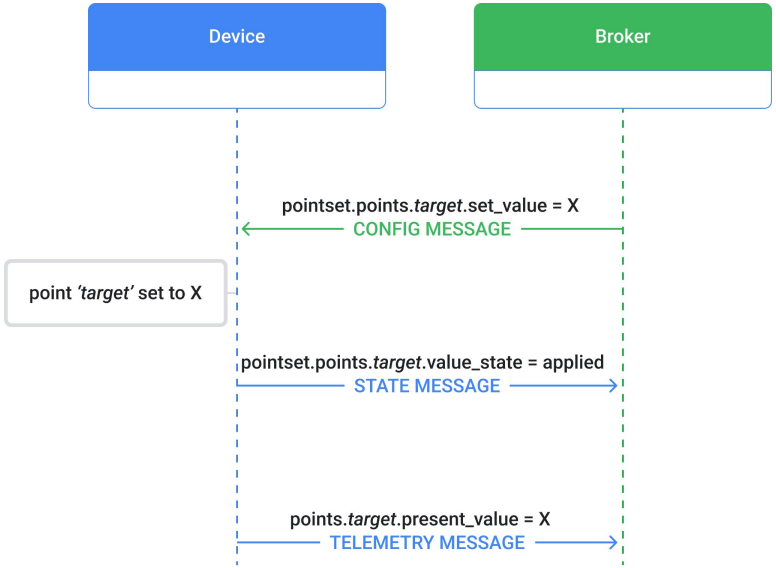
Sequences in UDMI (continued)

Writeback

This sequence defines the UDMI specification for cloud to device control (cloud actuation). This comprises several subsequences which each define different aspects of writeback.

The core writeback function is as follows:

1. The device receives a config message with a **`set_value`** field for a specific point.
2. The device applies the change, setting the value of the point to the value of the **`set_value`** field.
3. The device then sends a state message with the **`value_state`** of the point as applied. The point should now be reporting as telemetry the value defined in the writeback set value.



Back

Next

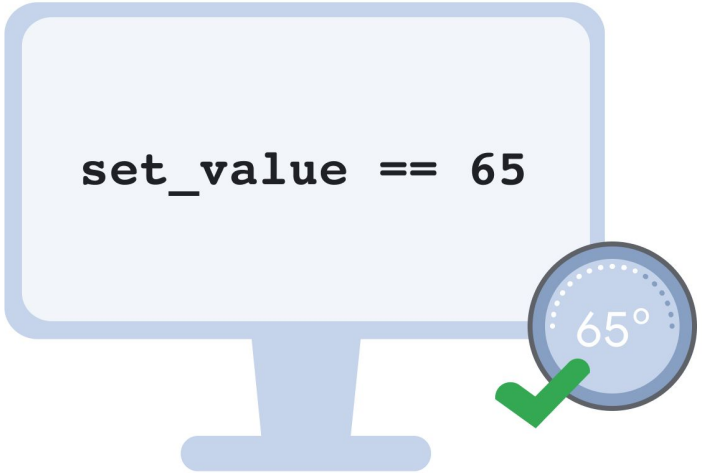
Google

Sequences in UDMI (continued)

If the writeback is successful, the device will apply the change and set the value of the point to the value of the **`set_value`** field.

Example

If the device receives a config message with a **`set_value`** of a thermostat temperature of 65 degrees, the device needs to apply that temperature change by setting the value of the point to the value of the **`set_value`** field. If the writeback was a success, the temperature on the thermostat would change to 65 degrees!



Back

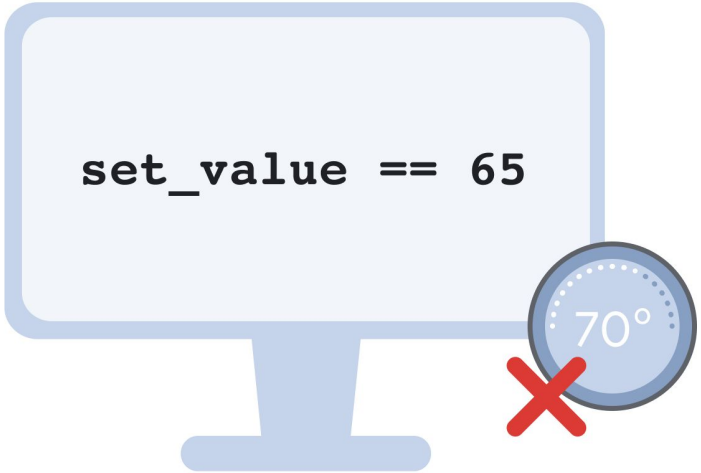
Next

Sequences in UDMI (continued)

If the writeback is unsuccessful, a state message with the **`value_state`** of the point as **`failure`** or **`invalid`** is sent.

- A state message with the **`value_state`** of the point as **`failure`** is sent if this was due to an error on the device side.
- A state message with the **`value_state`** of the point as **`invalid`** is sent if the requested value cannot be applied to the point (e.g. the point is not writable or the requested value is out of the operating bounds of the point).

> **Example**
>
> If we apply this to the previous scenario we discussed about the thermostat, but instead set the temperature to “blue,” the thermostat would not be changed to 65 degrees, as requested, because the requested value is considered invalid.



In both scenarios a status entry is also included in the point with a more detailed error.

Refer to Writeback for full specification.

Back

Next

Sequence testing with sequencer

Sequencer is one of the UDMI tools. It is invoked on an individual device, to test specific device behaviors and is used to validate that a device implements the different sequences and complies with expected standards.

The sequencer comprises different tests, each testing a different functionality. Each test is programmed with a series of sequential events, such as the examples in the previous section.

Each test is also allocated a time period. If the sequence does not complete in the time period, the test is considered a fail. A test can pass, fail, or skip (e.g., the test did not run due to missing configuration parameters).

For more information, refer to Sequencer Setup.

Running sequencer to test sequences

Let's practice running sequencer to test sequences. Use the following template of commands to run this process.

Sequencer usage:

```
bin/sequencer SITE_PATH PROJECT_ID DEVICE_ID [SERIAL_NO] [TEST_NAMES...]
```

- `SITE_PATH` is the path to your site model directory.
- `PROJECT_ID` is your Google Cloud Platform (GCP) project ID.
- `DEVICE_ID` is the DEVICE_ID of the device to test.
- `[SERIAL_NO]` (optional) is the serial number of the device under test. This is checked against the serial number disclosed by the device in its state message.
- `[TEST_NAMES]` (optional) are the specific test names.

To run sequencer to test sequences, start pubber so that there is an online device to test.

```
bin/pubber sites/udmi_site_model udmi-learning AHU-1 123
```

To run sequencer against the pubber instance started with the command above, use:

```
bin/sequencer sites/udmi_site_model udmi-learning AHU-1 123
```

Back

Note: Running sequencer with all tests will take at least several minutes against pubber and likely longer with a real device.

Next

Google

Sequencer test output

The sequence tester outputs a running log to the console.

At the end, a results summary is provided which includes the result and a summary for each test. Let's look at an example.

Example

```
XXXX-XX-XXTXX:XX:XXZ NOTICE sequencer RESULT pass broken_config Sequence complete
XXXX-XX-XXTXX:XX:XXZ NOTICE sequencer RESULT pass extra_config Sequence complete
XXXX-XX-XXTXX:XX:XXZ NOTICE sequencer RESULT pass periodic_scan Sequence complete
XXXX-XX-XXTXX:XX:XXZ NOTICE sequencer RESULT pass self_enumeration Sequence complete
XXXX-XX-XXTXX:XX:XXZ NOTICE sequencer RESULT pass single_scan Sequence complete
XXXX-XX-XXTXX:XX:XXZ NOTICE sequencer RESULT pass system_last_update Sequence complete
XXXX-XX-XXTXX:XX:XXZ NOTICE sequencer RESULT pass valid_serial_no Sequence complete
XXXX-XX-XXTXX:XX:XXZ NOTICE sequencer RESULT pass writeback_states Sequence complete
SUMMARY pass pass pass pass pass pass pass pass
```

Sequencer test output (continued)

The examples below describe what some of the common log messages are.

Select each box to display its description.

```
NOTICE sequencer starting test XXXXXXX
```

```
INFO sequencer Updated config with timestamp XXXX-XX-XXTXX:XX:XXZ
```

```
INFO sequencer start waiting for <condition>
```

```
INFO sequencer finished waiting for <condition>
```

```
INFO sequencer Updated state has last_config XXXX-XX-XXTXX:XX:XXZ
```

```
NOTICE sequencer RESULT <result> <test name> <reason>
```

Back

Next

Sequencer test output (continued)

A sequencer test has started.

```
2022-06-21T09:08:24Z NOTICE sequencer starting test system_last_update
```

```
NOTICE sequencer starting test XXXXXXX
```

```
INFO sequencer Updated config with timestamp XXXX-XX-XXTXX:XX:XXZ
```

```
INFO sequencer start waiting for <condition>
```

```
INFO sequencer finished waiting for <condition>
```

```
INFO sequencer Updated state has last_config XXXX-XX-XXTXX:XX:XXZ
```

```
NOTICE sequencer RESULT <result> <test name> <reason>
```

Back

Next

Google

Sequencer test output (continued)

Sequencer has sent a new config to the device with the timestamp.

2022-06-21T09:08:37Z INFO sequencer Updated config with timestamp 2022-06-21T09:08:21Z

```
NOTICE sequencer starting test XXXXXXX
```

```
INFO sequencer Updated config with timestamp XXXX-XX-XXTXX:XX:XXZ
```

```
INFO sequencer start waiting for <condition>
```

```
INFO sequencer finished waiting for <condition>
```

```
INFO sequencer Updated state has last_config XXXX-XX-XXTXX:XX:XXZ
```

```
NOTICE sequencer RESULT <result> <test name> <reason>
```

Back

Next

Google

Sequencer test output (continued)

This is very common. Sequencer is waiting for a condition. In this example, sequencer is waiting for a device config reset.

2022-06-21T09:08:37Z INFO sequencer start waiting for device config reset

```
NOTICE sequencer starting test XXXXXXX
```

```
INFO sequencer Updated config with timestamp XXXX-XX-XXTXX:XX:XXZ
```

INFO sequencer start waiting for <condition>

```
INFO sequencer finished waiting for <condition>
```

```
INFO sequencer Updated state has last_config XXXX-XX-XXTXX:XX:XXZ
```

```
NOTICE sequencer RESULT <result> <test name> <reason>
```

Back

Next

Google

Sequencer test output (continued)

Sequencer has finished waiting for the specified condition because it has transpired (e.g., the device has sent a state message if it was waiting for a device state update).

2022-06-21T09:08:41Z INFO sequencer finished waiting for device state update

```
NOTICE sequencer starting test XXXXXXX
```

```
INFO sequencer Updated config with timestamp XXXX-XX-XXTXX:XX:XXZ
```

```
INFO sequencer start waiting for <condition>
```

INFO sequencer finished waiting for <condition>

```
INFO sequencer Updated state has last_config XXXX-XX-XXTXX:XX:XXZ
```

```
NOTICE sequencer RESULT <result> <test name> <reason>
```

Back

Next

Google

Sequencer test output (continued)

A state message has been received, with a **`last_config`** field value in the message.

`2022-06-21T09:08:41Z INFO sequencer Updated state has last_config 2022-06-21T09:08:09Z`

```
NOTICE sequencer starting test XXXXXXX
```

```
INFO sequencer Updated config with timestamp XXXX-XX-XXTXX:XX:XXZ
```

```
INFO sequencer start waiting for <condition>
```

```
INFO sequencer finished waiting for <condition>
```

`INFO sequencer Updated state has last_config XXXX-XX-XXTXX:XX:XXZ`

```
NOTICE sequencer RESULT <result> <test name> <reason>
```

Back

Next

Sequencer test output (continued)

A test has concluded with the given result (pass/fail/skip) and reason.

2022-06-21T09:08:57Z NOTICE sequencer RESULT pass system_last_update Sequence complete

```
NOTICE sequencer starting test XXXXXXX
```

```
INFO sequencer Updated config with timestamp XXXX-XX-XXTXX:XX:XXZ
```

```
INFO sequencer start waiting for <condition>
```

```
INFO sequencer finished waiting for <condition>
```

```
INFO sequencer Updated state has last_config XXXX-XX-XXTXX:XX:XXZ
```

NOTICE sequencer RESULT <result> <test name> <reason>

Back

Next

Google

Sequencer test output (continued)

Sequencer will produce some files and directories into the out directory within the site model:

<SITE_MODEL>/out/<DEVICE_ID>, where **DEVICE_ID** is the identifier for the device tested. These include the latest messages received, log messages, summaries, and other files which are useful to diagnosing results.

The following are files and directories created by sequencer:

- **RESULT.log** – overall test results from sequencer
- **tests/** – directory, containing sub directories for each sequencer test which was run, e.g. **tests/system_last_update**

Back

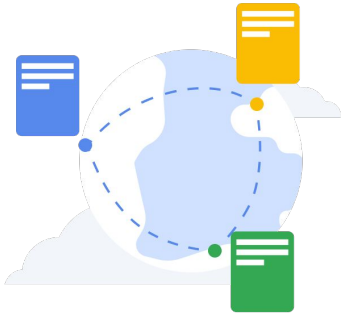
Next

Google

Sequencer test output (continued)

Each test subdirectory contains the following files:

- **`sequencer.log`** – logging messages generated by sequencer during a test run
- **`system.log`** – consolidation of all log entries (system events) sent by a device whilst the test was running
- **`config_update.json`** – latest config update sent to device
- **`state_update.json`** – latest state message sent by the device
- **`event_pointset.json`, `event_system.json`, `event_<SUBTYPE>.json`** – latest event messages of subtype sent by device



Back

Next

Running sequencer with an invalid serial

Remember, the command to run sequencer (shown below) offers serial number as an optional parameter.

When the serial number is provided:

- The sequencer test **`valid_serial_no`** will check that the serial number given by the device in the state message matches the serial number entered by the user.
- If the device serial does not match, a fail result is given:
  `2022-05-30T20:49:10Z NOTICE sequencer RESULT fail valid_serial_no received serial no expected:<[567]> but was:<[123]>`
- If a serial number is not provided, a **skip** result is given.

This can be useful for verifying that a device being tested is the actual device intended (i.e. accidentally providing sequencer with the wrong device ID, or a device which has been incorrectly registered, etc.).

Sequencer usage:

```
bin/sequencer SITE_PATH PROJECT_ID DEVICE_ID [SERIAL_NO] [TEST_NAMES...]
```

Back

Next

Google

Running sequencer with an invalid serial (continued)

Let's practice! To try running sequencer with an invalid serial, follow the instructions below:

1. Start pubber with a serial number of "123."

```
bin/pubber sites/udmi_site_model udmi-learning AHU-1 123
```

2. Start the sequencer, and enter an invalid serial number of 567 (this means that you're expecting the serial number to be 567).

```
bin/sequencer sites/udmi_site_model udmi-learning AHU-1 567
```

3. Look for the test result in the output.

```
2022-05-30T20:49:10Z NOTICE sequencer RESULT fail valid_serial_no received serial no expected:<[567]> but was:<[123]>
```

Back

Next

Google

Diagnosing sequencer results

To diagnose the sequencer test results, take a look at the message captures and sequencer logs.

The message captures and test logs are located in:
<SITE_MODEL>/out/AHU-1/tests/valid_serial_no

Look at the latest state message received in state_update.json, and notice the serial number reported was **123**.


```
{
 "pointset" : {
   "points" : {
     "filter_alarm_pressure_status" : { },
     "filter_differential_pressure_setpoint" : { },
     "filter_differential_pressure_sensor" : { }
   }
 },
 "system" : {
   "last_config" : "2022-06-10T15:10:35Z",
   "operational" : true,
   "serial_no" : "123",
   "hardware" : {
     "make" : "BOS",
     "model" : "pubber"
   },
   "software" : {
     "firmware" : "v1"
   }
 },
 "timestamp" : "2022-06-10T15:10:47Z",
 "version" : "1.3.14"
}
```


Back

Next

Google

Diagnosing sequencer results (continued)

To diagnose sequencer results, it is possible to:

- Run a single test.
- Toggle the verbosity level with a DEBUG or TRACE level output.

Run a single test

To run a single test, provide the name of the test after the serial number in the start command.

```
bin/sequencer SITE_PATH PROJECT_ID DEVICE_ID [SERIAL_NO] [TEST_NAMES...]
```

The possible tests are provided in the Sequencer Setup page.

For example, to run the **valid_serial_no** test only against the device AHU-1, you can enter:

```
bin/sequencer sites/udmi_site_model udmi-learning AHU-1 123 valid_serial_no
```

This helps save time when diagnosing the result of a single test.

Diagnosing sequencer results (continued)

Toggle verbosity levels

There are three verbosity levels when running sequencer:

- **INFO** (default) – Provides feedback for normal logging with important event updates
- **DEBUG** – Additionally shows:
  - The raw config messages sent to devices
  - The raw state message received from devices
  - The notification when events are received
- **TRACE** – Additionally shows:
  - Config and state sub-blocks values
  - Every message received, not just "most recent"

DEBUG would be beneficial to someone diagnosing an unexpected sequencer result.
TRACE would be beneficial to someone programming a sequence into a device (e.g. a device manufacturer).

These can be toggled with with the **-v** flag in the sequencer start command

```
bin/sequencer [-v] [-vv]  SITE_PATH PROJECT_ID DEVICE_ID [SERIAL_NO] [TEST_NAMES...]
```

- **Debug** level verbosity with a single v (-v)
- **Trace** level verbosity with two v's (-vv)

Back

Next

Diagnosing sequencer results (continued)

Select each box to display the description of example log messages you'll see when running in **TRACE** or **DEBUG** modes. The message will be displayed below the first line presented in the sequencer output logs.

```
DEBUG sequencer received event_system_2022-05-31T16:24:59Z
```

```
DEBUG sequencer Updated state:
```

```
DEBUG sequencer Updated config:
```

```
DEBUG sequencer update config_localnet
```

Back

Next

Google

Diagnosing sequencer results (continued)

Select each box to display the description of example log messages you'll see when running in **TRACE** or **DEBUG** modes. The message will be displayed below the first line presented in the sequencer output logs.

DEBUG sequencer received event_system_2022-05-31T16:24:59Z

Received an event at that time of type event_<event_type>.

DEBUG sequencer Updated state:

DEBUG sequencer Updated config:

DEBUG sequencer update config_localnet

Back

Next

Google

Diagnosing sequencer results (continued)

Select each box to display the description of example log messages you'll see when running in **TRACE** or **DEBUG** modes. The message will be displayed below the first line presented in the sequencer output logs.

```
DEBUG sequencer received event_system_2022-05-31T16:24:59Z
```

```
DEBUG sequencer Updated state:
```

Received an updated state message.

```
DEBUG sequencer Updated config:
```

```
DEBUG sequencer update config_localnet
```

Back

Next

Google

Diagnosing sequencer results (continued)

Select each box to display the description of example log messages you'll see when running in **TRACE** or **DEBUG** modes. The message will be displayed below the first line presented in the sequencer output logs.

```
DEBUG sequencer received event_system_2022-05-31T16:24:59Z
```

```
DEBUG sequencer Updated state:
```

```
DEBUG sequencer Updated config:
```

```
DEBUG sequencer update config_localnet
```

A new config has been sent to the device.

Back

Next

Google

Diagnosing sequencer results (continued)

Select each box to display the description of example log messages you'll see when running in **TRACE** or **DEBUG** modes. The message will be displayed below the first line presented in the sequencer output logs.

DEBUG sequencer received event_system_2022-05-31T16:24:59Z

DEBUG sequencer Updated state:

DEBUG sequencer Updated config:

DEBUG sequencer update config_localnet

A change to a config sub-block (localnet in this case, but could be pointset, system, etc) has occurred. The new config has not yet been sent to the device.

Back

Next

Lesson 6

Knowledge check

Let's take a moment to reflect on what you've learned so far.

- The next slides will have questions about the concepts that were introduced in this lesson.
- Review each question and select the correct response.

If there are more than two answer options, you won't be able to move forward until the correct answer is selected.

Back

*Click **Next** when you're ready to begin.*

Next

Google

Knowledge check 1

After running sequencer, sequence tester outputs a running log to the console. A results summary is provided, which includes the result and a summary for each test.

What does it mean if you see this common log message?

`INFO sequencer start waiting for <condition>`

Select the best answer from the options listed below.

- Sequencer has sent a new config to the device with the timestamp.
- A state message has been received, with a last_config field value in the message.
- Sequencer is on hold while something (e.g., a system log entry) is being received.
- Sequencer has finished waiting for specified conditions.

Back

Next

Google

Knowledge check 1

After running sequencer, sequence tester outputs a running log to the console. A results summary is provided, which includes the result and a summary for each test.

What does it mean if you see this common log message?

`INFO sequencer start waiting for <condition>`

Select the best answer from the options listed below.

- Sequencer has sent a new config to the device with the timestamp.
- A state message has been received, with a last_config field value in the message.
- Sequencer is on hold while something (e.g., a system log entry) is being received.
- Sequencer has finished waiting for specified conditions.

Close... but not quite right! 🤔

If sequencer has sent a new config to the device with a timestamp, you will see the following message:

```
INFO sequencer Updated config with timestamp
XXXX-XX-XXTXX:XX:XXZ
```

Try again

Back

Next

Google

Knowledge check 1

After running sequencer, sequence tester outputs a running log to the console. A results summary is provided, which includes the result and a summary for each test.

What does it mean if you see this common log message?

`INFO sequencer start waiting for <condition>`

Select the best answer from the options listed below.

Sequencer has sent a new config to the device with the timestamp.

A state message has been received, with a last_config field value in the message.

Sequencer is on hold while something (e.g., a system log entry) is being received.

Sequencer has finished waiting for specified conditions.

Close... but not quite right! 🤔

If a state message has been received with a last_config field value in the message, you will see the following message:

`INFO sequencer Updated state has last_config XXXX-XX-XXTXX:XX:XXZ`

Try again

Back

Next

Knowledge check 1

After running sequencer, sequence tester outputs a running log to the console. A results summary is provided, which includes the result and a summary for each test.

What does it mean if you see this common log message?

`INFO sequencer start waiting for <condition>`

Select the best answer from the options listed below.

Sequencer has sent a new config to the device with the timestamp.

A state message has been received, with a last_config field value in the message.

Sequencer is on hold while something (e.g., a system log entry) is being received.

Sequencer has finished waiting for specified conditions.

That's right! 🎉

This message is very common, and it means that sequencer is waiting for a condition. There are several conditions that could apply in this situation.

Back

Next

Knowledge check 1

After running sequencer, sequence tester outputs a running log to the console. A results summary is provided, which includes the result and a summary for each test.

What does it mean if you see this common log message?

`INFO sequencer start waiting for <condition>`

Select the best answer from the options listed below.

- Sequencer has sent a new config to the device with the timestamp.
- A state message has been received, with a last_config field value in the message.
- Sequencer is on hold while something (e.g., a system log entry) is being received.
- Sequencer has finished waiting for specified conditions.

Close... but not quite right! 🤔

If sequencer has finished waiting for a specific condition, you will see the following message:

`INFO sequencer finished waiting for <condition>`

Try again

Back

Next

Knowledge check 2

When running sequences, what does the failure of an invalid serial mean?

Select the best answer from the options listed below.

It means a serial number was not provided.

It means that the device being tested is not the actual device intended.

It means that the device being tested was the actual device intended.

Back

Next

Google

Knowledge check 2

When running sequences, what does the failure of an invalid serial mean?

Select the best answer from the options listed below.

It means a serial number was not provided.

It means that the device being tested is not the actual device intended.

It means that the device being tested was the actual device intended.

Close... but not quite right! 🤔

If a serial number is not provided while running sequences, a skip result is given, not a **failure** result.

Try again

Back

Next

Knowledge check 2

When running sequences, what does the failure of an invalid serial mean?

Select the best answer from the options listed below.

It means a serial number was not provided.

It means that the device being tested is not the actual device intended.

It means that the device being tested was the actual device intended.

That's right! 🎉

When running sequences leads to a failure result with an invalid serial, it means that the device being tested is not the intended device. For example, sequencer could have received the wrong device ID, or the device might have been incorrectly registered.

Back

Next

Knowledge check 2

When running sequences, what does the failure of an invalid serial mean?

Select the best answer from the options listed below.

It means a serial number was not provided.

It means that the device being tested is not the actual device intended.

It means that the device being tested was the actual device intended.

Close... but not quite right! 🤔

If the device being tested was the actual device intended, the serial numbers would have matched, and there would be no **failure** result.

Try again

Back

Next

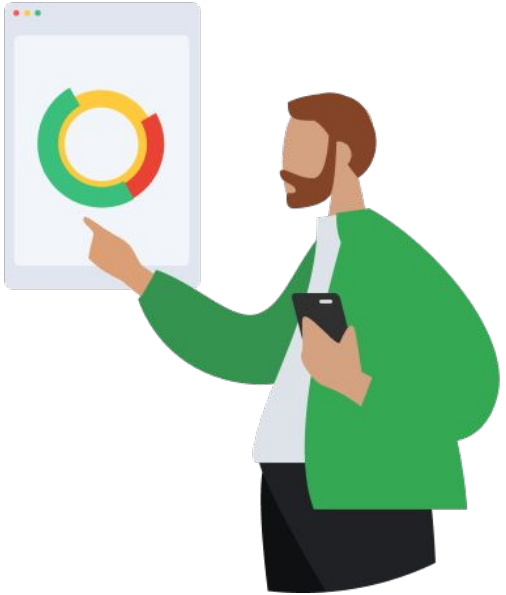
Lesson 6 summary

Let's review what you learned about:

- Advanced device features with bidirectional exchange

Now you should be able to:

- Run sequence tester against an individual pubber device.
- Examine result messages from a sequence test.
- Run sequence tester against Pubber with a bad serial number.
- Run and debug an individual sequence test.



Back

Next

Google

You completed Lesson 6!

Back

*Press the **Esc** key on your keyboard to exit presentation mode.*

Google

Helpful resources

Bookmark these resources for future reference.

- UDMI Project GitHub
  Contains specification for management and operation of IoT systems.
- Git Documentation
  Contains various sources of information about Git contributed by Git community.