Start Lab 01:30:00

Ingesting HL7v2 Data with the Healthcare API

Lab 1 hour No cost Introductory

GSP628

Overview

In this lab you will discover and use the basic functionality of Cloud Healthcare API using HL7v2 messages.

In this lab you will:

- Gain a general understanding of Cloud Healthcare API and its role in managing healthcare data.
- Learn how to create Cloud Healthcare API datasets and stores.
- Send test HL7 messages to a data store using the Cloud Healthcare API.

Setup and requirements

Overview

In this lab you will discover and use the basic functionality of Cloud Healthcare API using HL7v2 messages.

In this lab you will:

- Gain a general understanding of Cloud Healthcare API and its role in managing healthcare data.
- Learn how to create Cloud Healthcare API datasets and stores.
- Send test HL7 messages to a data store using the Cloud Healthcare API.

Setup and requirements

Overview

In this lab you will discover and use the basic functionality of Cloud Healthcare API using HL7v2 messages.

In this lab you will:

- Gain a general understanding of Cloud Healthcare API and its role in managing healthcare data.
- Learn how to create Cloud Healthcare API datasets and stores.
- Send test HL7 messages to a data store using the Cloud Healthcare API.

Setup and requirements

Overview

In this lab you will discover and use the basic functionality of Cloud Healthcare API using HL7v2 messages.

In this lab you will:

- Gain a general understanding of Cloud Healthcare API and its role in managing healthcare data.
- Learn how to create Cloud Healthcare API datasets and stores.
- Send test HL7 messages to a data store using the Cloud Healthcare API.

Setup and requirements

Overview

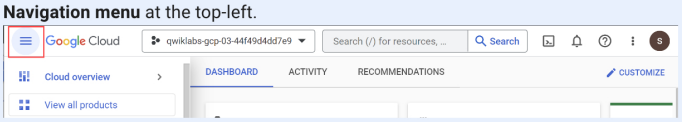
In this lab you will discover and use the basic functionality of Cloud Healthcare API using HL7v2 messages.

In this lab you will:

- Gain a general understanding of Cloud Healthcare API and its role in managing healthcare data.
- Learn how to create Cloud Healthcare API datasets and stores.
- Send test HL7 messages to a data store using the Cloud Healthcare API.

Setup and requirements

Note: To view a menu with a list of Google Cloud products and services, click the **Navigation menu** at the top-left.



Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

1. Click **Activate Cloud Shell** at the top of the Google Cloud console.

When you are connected, you are already authenticated, and the project is set to your **Project_ID**, `PROJECT_ID`. The output contains a line that declares the **Project_ID** for this session:

```
Your Cloud Platform project in this session is set to "PROJECT_ID"
```

`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

2. (Optional) You can list the active account name with this command:

```
gcloud auth list
```

3. Click **Authorize**.

Output:

```
ACTIVE: *
ACCOUNT: "ACCOUNT"

To set the active account, run:
    $ gcloud config set account `ACCOUNT`
```

4. (Optional) You can list the project ID with this command:


```
Your Cloud Platform project in this session is set to "PROJECT_ID"
```

`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

2. (Optional) You can list the active account name with this command:

```
gcloud auth list
```

3. Click **Authorize**.

Output:

```
ACTIVE: *
ACCOUNT: "ACCOUNT"

To set the active account, run:
    $ gcloud config set account `ACCOUNT`
```


4. (Optional) You can list the project ID with this command:

Cloud Healthcare API provides a managed solution for storing and accessing healthcare data in Google Cloud, providing a critical bridge between existing care systems and applications hosted on Google Cloud. Using the API, you can unlock significant new capabilities for data analysis, machine learning and application development, and use

these capabilities to build the next generation of healthcare solutions.

The API is comprised of three modality-specific interfaces that implement key industry-wide standards for healthcare data:

- FHIR, an emerging standard for health data interchange
- HL7v2, the most widely adopted method for health systems integration
- DICOM, the dominant standard for radiology and imaging-related disciplines

Each interface is backed by a standards-compliant data store that provides read, write, search, and other operations on the data.

The Cloud Healthcare API provides a number of key features that are critical to bridging current technologies to the next generation of healthcare systems and applications:

- **Standards conformance** - Google supports the use of standards-based interoperability through its participation in a number of healthcare standards bodies. In the Cloud Healthcare API each modality-specific data store and its associated API is substantially conformant with its relevant standard. For example, FHIR stores implement STU3, the current version of the FHIR specification, and DICOM stores implement DICOMweb, a web-based standard for exchanging medical images. In future updates, we expect to support additional versions of these specifications as well as the ability to request a resource in a different version than its canonical representation.
- **Compliance with privacy regulations** - Google Cloud provides detailed guidance regarding how it supports compliance with HIPAA in the US, the PIPEDA in Canada, and other global privacy standards at cloud.google.com/security/compliance.
- **Data location control** - The Cloud Healthcare API treats data location as a core component of the API. You have the option to select the storage location for each dataset from a list of currently available locations which correspond to distinct geographic areas aligned with Google Cloud's regional structure. Future Google Cloud regions will allow for the distribution of storage across wider geographic areas.
- **Security** - The Cloud Healthcare API security model is based on Google's proven Identity and Access Management (IAM) system. IAM's fine-grained permissions give you complete control over access to your healthcare data. In addition, we've created open-source proxies for our powerful Apigee API Management system, which provides comprehensive threat detection and traffic management capabilities that allow you to securely expose sensitive ePHI with patient and provider applications.
- **Bulk import and export** - The Cloud Healthcare API's DICOM and FHIR modalities support bulk import and export of data, making it easier to transfer data via the Cloud Storage system.
- **De-identification** - De-identification support for DICOM is available, making it much easier to redact patient information from studies for research and other purposes. The de-identification process operates on a data store basis.
- **Auditability** - Both administrative and data access requests to the Cloud Healthcare API can be audited. Logs are available through Google Cloud's Stackdriver hybrid monitoring system.
- **High availability** - Availability for mission-critical scenarios is made possible through Google Cloud's robust and highly redundant infrastructure.

For many applications, the Cloud Healthcare API can provide a modern alternative to legacy stacks implementing DICOM, HL7v2 or FHIR STU3 standards, simplifying data integration with existing systems and enabling the application developers to focus on their differentiating features such as UX and intelligence.

Healthcare API concepts

To get the most out of the Cloud Healthcare API, there are a few key concepts you'll want to understand. The information below should give you a good sense of Cloud Healthcare API capabilities, but you can find more details in the Cloud Healthcare API documentation.

General structure of the Cloud Healthcare API

General structure of the Cloud Healthcare API

The Cloud Healthcare API exposes interfaces that enable you to perform different types of functions:

- **Administrative functions**, such as creating or listing datasets and stores that will contain your data.
- **Data access functions** that allow you to create, update, delete and search the data stored in Cloud Healthcare API, or to perform bulk import and export operations.
- **Security functions** that allow you to impose access controls on data stored in Cloud Healthcare API.
- **De-identification functions** that allow you to replace ePHI with anonymized data, or to obfuscate ePHI so that it cannot be used.
- **Metadata functions**, such as retrieval of a FHIR capabilities statement for the FHIR API.

These functions may vary slightly depending on the modality of data (FHIR, HL7 v2 or DICOM) being operated on. For example, data retrieval operations against an FHIR data store use an API that conforms to the FHIR standard, but data retrieval operations against an HL7 v2 store use operations better suited to operating on HL7v2-structured data.

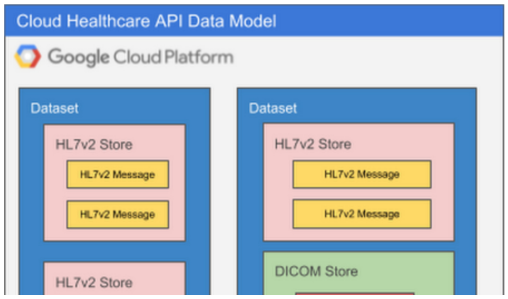
Datasets and stores

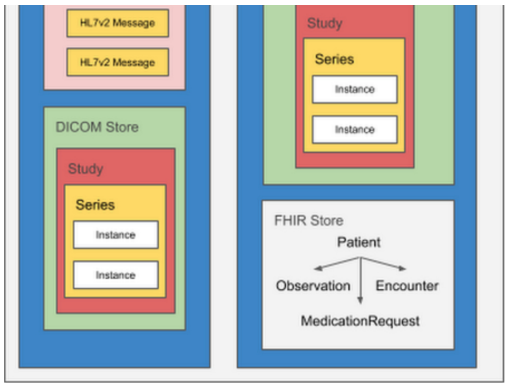
All Cloud Healthcare API usage occurs within the context of a Google Cloud project. Projects form the basis for creating, enabling, and using all Google Cloud services including managing APIs, enabling billing, adding and removing collaborators, and managing permissions for Google Cloud resources. Cloud Healthcare API can be used in one or many Google Cloud projects, as appropriate; this flexibility allows you to separate production from non-production usage, for example, or to segregate applications and resources in order to better manage access or accommodate different development lifecycles.

Within a project, data ingested through Cloud Healthcare API is stored in a **dataset**, which resides in a geographic location corresponding to a specific Google Cloud region. You use the Cloud Healthcare API's administrative functions to create a dataset in a particular location; doing so facilitates implementation of data location requirements for the countries in which your applications provide services. For example, you can choose to create a dataset in Google Cloud's "us-central1" region for US-based applications, or in an EU or UK region for applications serving those customers. This level of location control is also available in other Google Cloud products, which can be combined with Cloud Healthcare API to create a complete application architecture. A list of generally available Google Cloud products and the regions in which they are implemented can be found on Google Cloud, Cloud locations.

Because each healthcare data modality has different structural and processing characteristics, datasets are split into modality-specific **stores**. A single dataset can contain one or many stores, and those stores can all service the same modality or different modalities as application needs dictate. Using multiple stores in the same dataset might be appropriate if a given application processes different types of data, for example, or if you'd like to be able to separate data according to its source hospital, clinic, department, etc. An application can access as many datasets or stores as its requirements dictate with no performance penalty, so it's important to design your overall dataset and store architecture to meet the organization's broad goals for locality, partitioning, access control, and so on.

The diagram below illustrates two datasets in a Google Cloud project, each of which contains multiple stores.





There are many ways to structure datasets and stores. As you design systems that use the Cloud Healthcare API, you may want to take the following into consideration:

- **Security and access control:** Rules can be defined at both a dataset and store level, but you may choose to group all data for a particular application into the same dataset, and set access control rules such that only that application can access the dataset.
- **Application requirements:** An application processing different types of data may have all of its data for all modalities in a single dataset.
- **Source systems:** Often, the structure of healthcare data can vary according to the source system and modality. Separating data for different source systems into their own datasets may facilitate processing.
- **Intended use:** Data from different systems can have different intended uses, such as research, analytics, or machine learning predictions. Grouping data by intended use may facilitate ingestion into the target system.
- **Separating ePHI from de-identified data:** Cloud Healthcare API data de-identification functions read from a source dataset and write the output into a new dataset that you specify. If you are preparing data to be used by researchers, for example, this approach to de-identifying data may be a consideration in how you use datasets to segregate data.

MLLP Adapter

The minimal lower layer protocol (MLLP) is the standard used for transmitting HL7v2 messages over TCP/IP connections within a network, such as a hospital.

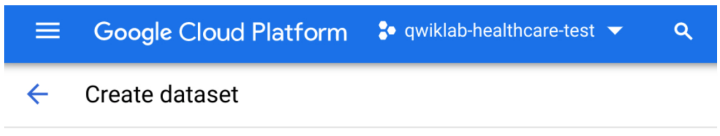
MLLP does not offer an exact mapping to the Cloud Healthcare API HL7v2 REST API], which uses HTTP. Therefore, an MLLP adapter must be used to convert messages transmitted over MLLP into a format that an HTTP/REST API can accept. To transmit messages over MLLP and then to the Cloud Healthcare API, use the Google Cloud MLLP adapter.

Task 1. Create Healthcare dataset

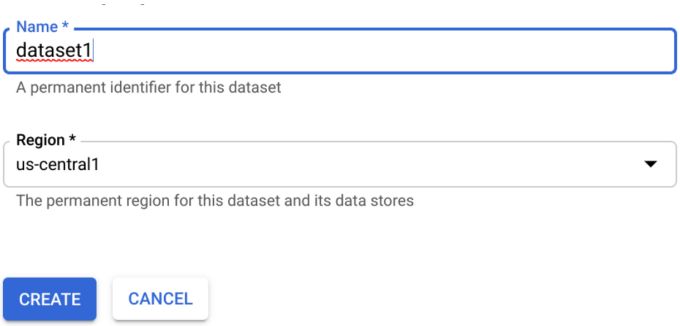
1. In the Cloud Console, from the **Navigation Menu**, scroll down to the Analytics section, then select **Healthcare**, and click **Enable**.

2. Click **Create Dataset**.

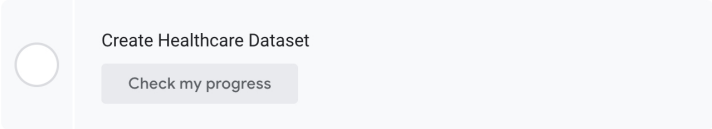
3. Name the dataset **dataset1** within region `REGION` and click **Create**.



Dataset properties

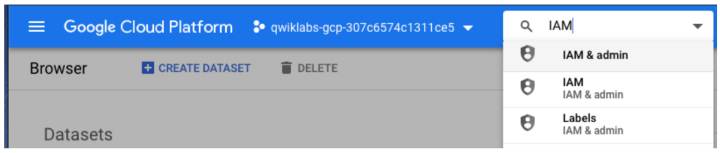


Click **Check my progress** to verify the objective.



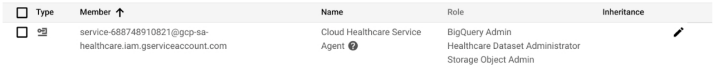
Task 2. Set up IAM permissions

1. Navigate to the **IAM & Admin** > **IAM** using the navigation panel or search bar.



2. In the **IAM** page, select the **Include Google-provided role grants** box.

3. Edit the permissions for your **Cloud Healthcare Service Agent** by locating the service agent under the IAM list and selecting the pencil icon. The service account will have the Domain @gcp-sa-healthcare.iam.gserviceaccount.com.

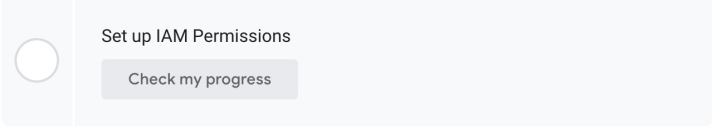


4. Click **add another role** in the dialog and then selecting the appropriate role.

- **BigQuery > BigQuery Admin**
- **Cloud Storage > Storage Object Admin**
- **Cloud Healthcare > Healthcare Dataset Administrator**
- **Pub/Sub > Pub/Sub Publisher**

5. When all of the above roles have been added, click **Save** to commit your updates.

Click **Check my progress** to verify the objective.



Task 3. Define variables needed

- In Cloud Shell, run the following to set variables needed for this lab:

```
export PROJECT_ID=`gcloud config get-value project`
export REGION=REGION
export DATASET_ID=dataset1
export FHIR_STORE_ID=fhirstore1
export DICOM_STORE_ID=dicomstore1
export HL7_STORE_ID=hl7v2store1
```

Task 4. Create a Cloud Pub/Sub topic and subscription

You need to configure a Cloud Pub/Sub topic with your HL7v2 store to receive notifications when messages are ingested.

1. Create a topic using the following command:

```
gcloud pubsub topics create projects/$PROJECT_ID/topics/hl7topic
```

This will return:

```
Created topic [projects/qwiklabs-gcp-{ID}/topics/hl7topic].
```

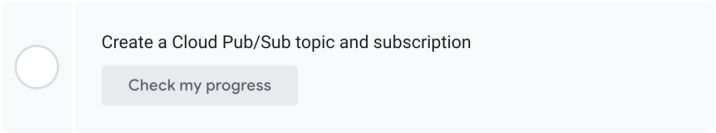
2. Subscribe to the topic using the following command:

```
gcloud pubsub subscriptions create hl7_subscription --
topic=hl7topic
```

This will return:

```
Created subscription [projects/qwiklabs-gcp-
{ID}/subscriptions/hl7_subscription].
```

Click **Check my progress** to verify the objective.



Task 5. Create data stores

Data in Cloud Healthcare API datasets and stores can be accessed and managed using a REST API that identifies each store using its project, location, dataset, store type and store name. This API implements modality-specific standards for access that are consistent with industry standards for that modality. The HL7v2 API provides operations for reading and searching HL7v2 messages based on HL7v2 message or segment criteria.

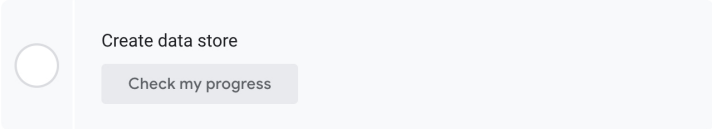
- Call the API to create a HL7v2 store:

```
gcloud healthcare hl7v2-stores create $HL7_STORE_ID --
dataset=$DATASET_ID --location=$REGION --notification-
config=pubsub-topic=projects/$PROJECT_ID/topics/hl7topic
```

The server returns the path to the newly created store.

Operations that access a modality-specific store use a request path that is comprised of two pieces: a base path, and a modality-specific request path. Administrative operations—which generally operate only on locations, datasets and stores—may only use the base path, but data modality-specific retrieval operations use both the base path (for identifying the store to be accessed) and request path (for identifying the actual data to be retrieved).

Click **Check my progress** to verify the objective.



Task 6. MLLP adapter setup and cloud HL7v2 store connection

As you saw above in the Healthcare API Concepts, you need to use the Google Cloud MLLP Adapter. You will set it up as a local Docker container and connect it to the HL7v2 store.

The MLLP adapter docker image has been built beforehand and staged in the cloud healthcare public registry. This image is used throughout the lab.

1. To run the `setup_mllp_adapter.sh` and create your MLLP adapter GKE cluster and HL7v2 store, issue the following command:

```
docker pull gcr.io/cloud-healthcare-containers/mllp-adapter:latest
```

This may take several minutes to complete.

2. When complete, there will be a banner displaying details of results:

```
Status: Downloaded newer image for gcr.io/cloud-healthcare-containers/mllp-adapter:latest
```

Task 7. Testing

The easiest way of testing the setup without exposing the server to public Internet is using port-forward command.

1. Run the following:

```
docker run \
    --network=host \
    gcr.io/cloud-healthcare-containers/mllp-adapter \
    /usr/mllp_adapter/mllp_adapter \
    --hl7_v2_project_id=$PROJECT_ID \
    --hl7_v2_location_id=$REGION \
    --hl7_v2_dataset_id=$DATASET_ID \
    --hl7_v2_store_id=$HL7_STORE_ID \
    --export_stats=false \
    --receiver_ip=127.0.0.1 \
```

```
    --receiver_ip=127.0.0.1 \
    --pubsub_project_id=$PROJECT_ID \
    --pubsub_subscription=hl7_subscription \
    --mllp_addr=127.0.0.1:2575 \
    --logtostderr
```

2. Open a second Cloud Shell tab by clicking the **+** button next to the open terminal, then install NetCat:

```
sudo apt install netcat
```

3. Type **"y"** when asked to confirm this action.

4. Still in the second shell, copy the `hl7v2-mllp-sample.txt` file to your local directory:

```
curl https://cloud.google.com/healthcare-api/docs/resources/hl7v2-mllp-sample.txt --output hl7v2-mllp-sample.txt
```

5. In the same directory where you added the file, run the following command to start sending HL7v2 messages to your HL7v2 store:

```
echo -n -e "\x0b$(cat hl7v2-mllp-sample.txt)\x1c\x0d" | nc localhost 2575
```

The message will be sent through the MLLP adapter to your HL7v2 store. If the message was successfully ingested into the HL7v2 store, the command returns the following output:

```
MSA|AA|20150503223000CILITY|FROM_APP|FROM_FACILITY|20190502181627||ACK|c1f
cdc2-497f-b701-a71efc61bf0d|P|2.5
```

6. Navigate back to the first Cloud Shell tab and verify the message was sent by viewing the output. There should be lines for dialing, accepted, sending, and message was successfully sent.

Note: You can safely ignore any **handler** errors.

7. Still on the first Cloud Shell tab, close the running Docker application with **Ctrl** + **C**.

8. List the messages in your HL7v2 store to ensure the message was added:

```
curl -X GET \
     -H "Authorization: Bearer "$(gcloud auth print-access-token) \
     -H "Content-Type: application/json; charset=utf-8" \

"https://healthcare.googleapis.com/v1/projects/$PROJECT_ID/locatior
```

It will output the following:

```
{
 "messages": [
   "projects/qwiklabs-gcp-{ID}/locations/us-central1/datasets/dataset1/hl7V2Stores/hl7v2store1/messages/PTXYFyqOkHq_yF
 ]
}
```

You've successfully sent a HL7v2 message to a Cloud Healthcare API datastore.

Deploying the MLLP adapter to Google Kubernetes Engine

In a production environment, when transmitting HL7v2 messages over MLLP from your care center, one possible configuration is to send the messages to an adapter that is deployed in Google Cloud and can forward them to the Cloud Healthcare API.

The MLLP adapter runs as a stateless application on a GKE cluster. A GKE cluster is a managed group of VM instances for running containerized applications. Stateless applications are applications which do not store data or application state to the cluster or to persistent storage. Instead, data and application state stay with the client, which makes stateless applications more scalable.

GKE uses the Deployment controller to deploy stateless applications as uniform, non-unique Pods. Deployments manage the desired state of your application: how many Pods should run your application, what version of the container image should run, what the Pods should be labelled, and so on. The desired state can be changed dynamically through updates to the Deployment's Pod specification.

At the same time that you deploy the adapter, you create a Service controller that allows you to connect the adapter to the Cloud Healthcare API using internal load balancing.

Lab review

Cloud Healthcare API provides a comprehensive facility for ingesting, storing, managing, and securely exposing healthcare data in FHIR, DICOM, and HL7 v2 formats. Using Cloud Healthcare API, you can ingest and store data from electronic health records systems (EHRs), radiological information systems (RISs), and custom healthcare applications. You can then immediately make that data available to applications for analysis, machine learning prediction and inference, and consumer access.

Cloud Healthcare API enables application access to healthcare data via widely-accepted, standards-based interfaces such as FHIR STU3 and DICOMweb. These APIs allow data ingestion into modality-specific data stores, which support data retrieval, update, search and other functions using familiar standards-based interfaces.

Further, the API integrates with other capabilities in Google Cloud through two primary mechanisms:

- *Cloud Pub/Sub*, which provides near-real-time updates when data is ingested into a Cloud Healthcare API data store, and
- *Import/export APIs*, which allow you to integrate Cloud Healthcare API into both Google Cloud Storage and Google BigQuery.

Using Cloud Pub/Sub with Google Cloud Functions enables you to invoke machine learning models on healthcare data, storing the resulting predictions back in Cloud Healthcare API data store. A similar integration with Cloud Dataflow supports transformation and cleansing of healthcare data prior to use by applications.

To support healthcare research, Cloud Healthcare API offers de-identification capabilities for FHIR and DICOM. This feature allows customers to share data with researchers working on new cutting-edge diagnostics and medicines.

Congratulations

Now you have some general knowledge of the Cloud Healthcare API and its role in managing healthcare data. You have learned how to create datasets and stores for HL7v2 data and imported HL7v2 data.

Finish your quest

This self-paced lab is part of the Cloud Healthcare API quest. A quest is a series of related labs that form a learning path. Completing this quest earns you a badge to recognize your achievement. You can make your badge or badges public and link to them in your online resume or social media account. Enroll in this quest or any quest that contains this lab and get immediate completion credit. See the Google Cloud Skills Boost catalog to see all available quests.

Take your next lab

Continue your quest with Ingesting FHIR Data with the Healthcare API or try one of these suggestions:

- De-identifying DICOM Data with the Healthcare API

End your lab

When you have completed your lab, click **End Lab**. Your account and the resources you've used are removed from the lab platform.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

- 1 star = Very dissatisfied
- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.

Manual Last Updated October 20, 2023

Lab Last Tested October 23, 2023