

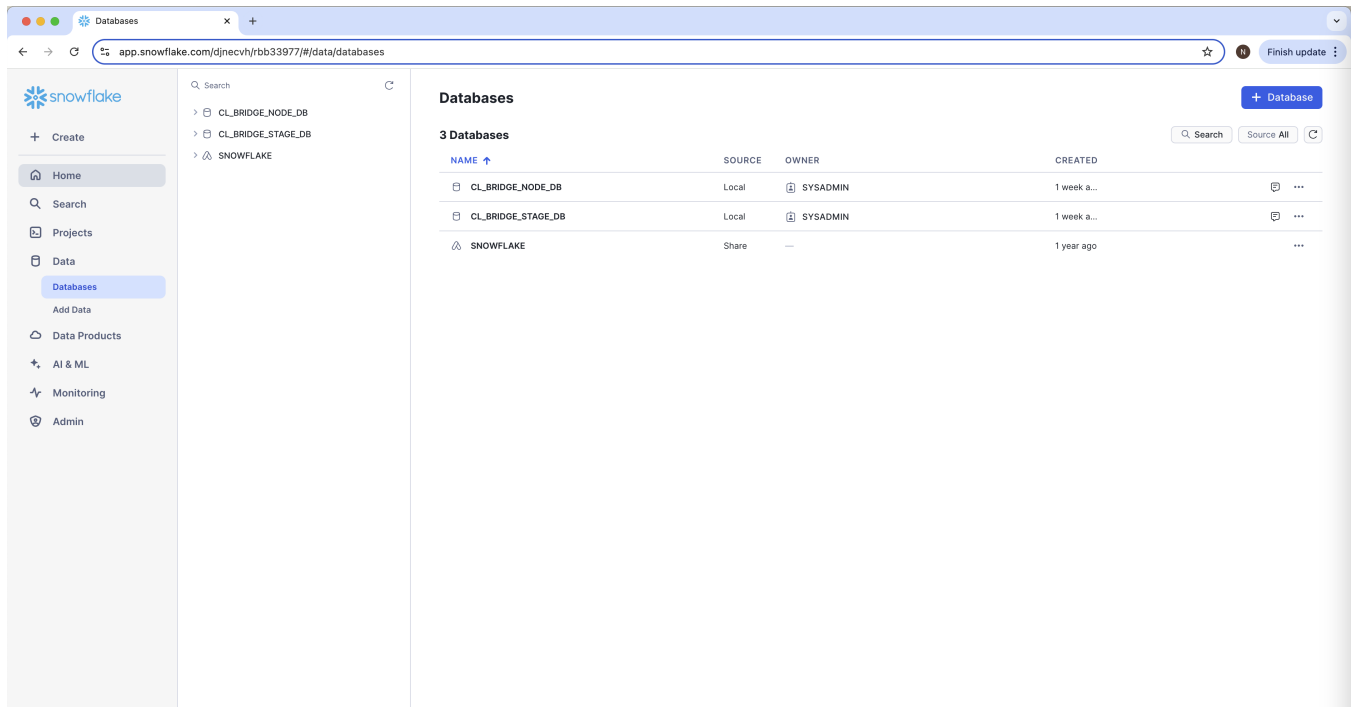
IBSNOW: Viewing the data in Snowflake

New IoT Bridge for Snowflake Available

This document covers v2.0.0 and older versions of IoT Bridge for Snowflake. It now ships as part of Chariot. To see the latest IoT Bridge for Snowflake documentation, go to [IoT Bridge for Snowflake in Chariot](#).

Data can be view in the databases created by the [Snowflake Setup Scripts](#) during your initial setup:

- Staging database created by SQL Script 01 (e.g., CL_BRIDGE_STAGE_DB)
- Node database created by SQL Script 02 (e.g., CL_BRIDGE_NODE_DB)



Terminology

As the data moves from the Ignition Edge device through MQTT Sparkplug and into Snowflake different terminology will be used with the equivalence shown below:

Ignition Edge	MQTT Sparkplug	Snowflake	Snowflake tables/views
Ignition UDT Definition	Sparkplug Template	Model	Template_Reference
Ignition UDT Instance	Sparkplug Template Instance	Model Instance	Device_Name

Staging Database

The Staging database contains only a single table, SPARKPLUG_RAW. This table contains information derived from each Sparkplug message, like the Sparkplug IDs: GroupId, EdgenodeId and DeviceId, the message timestamp, the model / model instance, the raw message payload in JSON, etc.

The table is populated by the Snowflake Snowpipe Streaming API as data is streamed in by the IoT bridge.

The screenshot shows the Snowflake interface with a SQL query executed against the SPARKPLUG_RAW table. The query filters for GROUP_ID = 'G1' and EDGE_NODE_ID = 'E1'. The results table shows two rows of data.

MSG_TOPIC	NAMESPACE	GROUP_ID	EDGE_NODE_ID	DEVICE_ID	MESSAGE_TYPE	MSG	INSERTED_AT
spBv1.0/G1/DBIRTH/E1/PLC 1	spBv1.0	G1	E1	PLC 1	DBIRTH	{ "metrics": [{ "dataType": "Template", "metaData": { "contentType": "", "name": "Node Control/Next Server", "value": "" } }] }	1730848515389
spBv1.0/G1/NBIRTH/E1	spBv1.0	G1	E1	null	NBIRTH	{ "metrics": [{ "dataType": "Boolean", "name": "Node Control/Next Server", "value": "" }] }	1730848511331

Node Database

The Node database also contains only a single table, SPARKPLUG_DEVICE_MESSAGES, but also has additional views, user defined functions, store procedures, streams, etc.

The SPARKPLUG_DEVICE_MESSAGES table contains partially processed data where each row in the table is a single message containing data for each model instance (aka. Device Name). This data is sourced from SPARKPLUG_RAW table and is inserted by the synch_device_messages() stored procedure. The synch_device_messages() stored procedure is executed on demand by the IoT Bridge for Snowflake.

The screenshot shows the Snowflake interface with a SQL query executed against the SPARKPLUG_DEVICE_MESSAGES table. The query filters for GROUP_ID = 'G1' and EDGE_NODE_ID = 'E1'. The results table shows 13 rows of data.

GROUP	EDGE	DEVICE_ID	DEVICE_NAME	TEMPLATE_REFERENCE	DEVICE_METRIC_TIMESTAMP	DDATA_MSG	MEASURES_INFO
G1	E1	PLC 1	TestMotor	Motor	1730316124378	{ "dataType": "Template", "metaData": { "State/Current State": 87.56264820205526 } }	17
G1	E1	PLC 1	Line 1	Assembly Line	1730316159611	{ "dataType": "Template", "metaData": { "State/Last Error": 100232 } }	17
G1	E1	PLC 1	Line 1	Assembly Line	1730316700173	{ "dataType": "Template", "metaData": { "State/Last Error": 100232 } }	17
G1	E1	PLC 1	Line 1	Assembly Line	1730848510872	{ "dataType": "Template", "metaData": { "State/Last Error": 100232 } }	17
G1	E1	PLC 1	Line 1	Assembly Line	1730318408732	{ "dataType": "Template", "metaData": { "State/Last Error": 100232 } }	17
G1	E1	PLC 1	Line 1	Assembly Line	1730318054483	{ "dataType": "Template", "metaData": { "State/Last Error": 100232 } }	17
G1	E1	PLC 1	Line 1	Assembly Line	1730316159611	{ "dataType": "Template", "metaData": { "State/Last Error": 100232 } }	17
G1	E1	PLC 1	Line 1	Assembly Line	1730318408732	{ "dataType": "Template", "metaData": { "State/Last Error": 100232 } }	17
G1	E1	PLC 1	Line 1	Assembly Line	1730316700173	{ "dataType": "Template", "metaData": { "State/Last Error": 100232 } }	17
G1	E1	PLC 1	Line 1	Assembly Line	1730316159611	{ "dataType": "Template", "metaData": { "State/Last Error": 100232 } }	17
G1	E1	PLC 1	Line 1	Assembly Line	1730318054483	{ "dataType": "Template", "metaData": { "State/Last Error": 100232 } }	17
G1	E1	PLC 1	TestMotor	Motor	1730316159611	{ "dataType": "Template", "metaData": { "State/Last Error": 0 } }	17
G1	E1	PLC 1	Line 1	Assembly Line	1730324917649	{ "dataType": "Template", "metaData": { "State/Last Error": 100232 } }	17

The screenshot shows the IoT Bridge for Snowflake interface. On the left, a sidebar displays a database schema tree for 'CL_BRIDGE_NODE_DB', including tables like 'SPARKPLUG_DEVICE_MESSAGES'. The main area shows a SQL query:

```

1 select
2   GROUP_ID, EDGE_NODE_ID, DEVICE_ID,
3   DEVICE_NAME,
4   TEMPLATE_REFERENCE,
5   -- ROOT_MESSAGE_TIMESTAMP,
6   DEVICE_METRIC_TIMESTAMP,
7   DDATA_MSG,
8   MEASURES_INFO,
9   -- MEASURE_TIMESTAMP,
10  -- MEASURE_TS
11  -- MESSAGE_TYPE,
12  INSERTED_AT
13 from CL_BRIDGE_NODE_DB.STAGE_DB.SPARKPLUG_DEVICE_MESSAGES
14 where GROUP_ID = 'G1' and EDGE_NODE_ID = 'E1' and TEMPLATE_REFERENCE = 'Motor' and DEVICE_NAME = 'Motor 3'

```

Below the query, the 'Results' tab shows a table with 5 rows and 9 columns:

GROUP_ID	EDGE_NODE_ID	DEVICE_ID	DEVICE_NAME	TEMPLATE_REFERENCE	DEVICE_METRIC_TIMESTAMP	DDATA_MSG	MEASURES_INFO	INSERTED_AT
G1	E1	PLC 1	Motor 3	Motor	1730848510872	{ "dataType": "Templ...	{ "RPMs": 1202}	1730848515389
G1	E1	PLC 1	Motor 3	Motor	1730848510872	{ "dataType": "Templ...	{ "Temperature": 49}	1730848515389
G1	E1	PLC 1	Motor 3	Motor	1730848510872	{ "dataType": "Templ...	{ "State/Last Error": 0}	1730848515389
G1	E1	PLC 1	Motor 3	Motor	1730848510872	{ "dataType": "Templ...	{ "State/Current State": 0}	1730848515389
G1	E1	PLC 1	Motor 3	Motor	1730848510872	{ "dataType": "Templ...	{ "Amps": 1211}	1730848515389

On the right, 'Query Details' shows a duration of 110ms and a query ID. Below the table, there are filters for 'GROUP_ID' (G1), 'EDGE_NODE_ID' (E1), and 'DEVICE_ID' (PLC 1).

The IoT Bridge for Snowflake will automatically create a new schema in its "Node" database for every Sparkplug edge node it finds. The Bridge will then create Views within this schema for each model found in the Sparkplug edge node data ingested via the bridge . There are three views per model:

Base View

This view contains the model instance tag data where each row is a single tag change per model instance. Tag change values are still encoded in JSON in the MEASURES_INFO and DDATA_MSG columns.

- Example: CL_BRIDGE_NODE_DB.NB_GROUP_EDGE.MOTOR

The screenshot shows the IoT Bridge for Snowflake interface. The left sidebar displays a tree view of databases and views, with 'MOTOR' selected. The main panel shows a SQL query:

```

select
  GROUP_ID, EDGE_NODE_ID, DEVICE_ID,
  DEVICE_NAME,
  MEASURE_TIMESTAMP,
  MEASURES_INFO,
  INSERTED_AT
from CL_BRIDGE_NODE_DB.NB_G1_E1.MOTOR
where GROUP_ID = 'G1' and EDGE_NODE_ID = 'E1' and DEVICE_NAME = 'Motor 3'

```

The results table below the query shows 5 rows of data:

GROUP_ID	EDGE_NODE_ID	DEVICE_ID	DEVICE_NAME	MEASURE_TIMESTAMP	MEASURES_INFO	INSERTED_AT
G1	E1	PLC 1	Motor 3	1730848316753	{ "State/Last Error": 0 }	1730848515389
G1	E1	PLC 1	Motor 3	1730848316753	{ "Temperature": 49 }	1730848515389
G1	E1	PLC 1	Motor 3	1730848316753	{ "State/Current State": 0 }	1730848515389
G1	E1	PLC 1	Motor 3	1730848316753	{ "Amps": 121 }	1730848515389
G1	E1	PLC 1	Motor 3	1730848316753	{ "RPMs": 1202 }	1730848515389

The right sidebar shows query details, including a duration of 79ms and a list of rows.

Intermediate View

This view contains the model instance tag data where each row is a single tag change per model instance. In this view, each model instance tag is separated out into its own column with their values. This view will show each individual model instance tag change and the tags that did not changed will be marked as null.

- Example: CL_BRIDGE_NODE_DB.NB_GROUP_EDGE.MOTOR_VW

The screenshot shows the IoT Bridge for Snowflake interface. The left sidebar displays a tree view of databases and views, with 'MOTOR_VW' selected. The main panel shows a SQL query:

```

select
  GROUP_ID, EDGE_NODE_ID, DEVICE_ID,
  DEVICE_NAME,
  MEASURE_TIMESTAMP,
  TEMPERATURE,
  RPMs,
  AMPS,
  INSERTED_AT
from CL_BRIDGE_NODE_DB.NB_G1_E1.MOTOR_VW
where GROUP_ID = 'G1' and EDGE_NODE_ID = 'E1' and DEVICE_NAME = 'Motor 3'

```

The results table below the query shows 5 rows of data:

GROUP_ID	EDGE_NODE_ID	DEVICE_ID	DEVICE_NAME	MEASURE_TIMESTAMP	TEMPERATURE	RPMs	AMPS	INSERTED_AT
G1	E1	PLC 1	Motor 3	1730848316753	null	null	null	1730848515389
G1	E1	PLC 1	Motor 3	1730848316753	null	null	null	1730848515389
G1	E1	PLC 1	Motor 3	1730848316753	null	121	null	1730848515389
G1	E1	PLC 1	Motor 3	1730848316753	49	null	null	1730848515389
G1	E1	PLC 1	Motor 3	1730848316753	null	1202	null	1730848515389

The right sidebar shows query details, including a duration of 40ms and a list of rows.

Asof View

This view is basically the same as the Intermediate view - each model instance tag is separated out into its own column. However, each row in this view has the null values from the Intermediate views removed (unless the tag change really had a null value) and the last known good value for the model instance tag will be replicated in its column. Said another way, each row will have the last known good values for all model instance tags.

- Example: CL_BRIDGE_NODE_DB.NB_GROUP_EDGE.MOTOR_ASOF_VW

The screenshot shows the Snowflake IoT Bridge interface. On the left, a navigation pane shows the database structure: CL_BRIDGE_NODE_DB > INFORMATION_SCHEMA > NB_G1_E1 > Views > MOTOR_ASOF_VW. The main area displays a SQL query:

```
1 select
2   GROUP_ID, EDGE_NODE_ID, DEVICE_ID,
3   DEVICE_NAME,
4   MEASURE_TIMESTAMP,
5   TEMPERATURE,
6   RPMS,
7   AMPS,
8   INSERTED_AT
9 from CL_BRIDGE_NODE_DB.NB_G1_E1.MOTOR_ASOF_VW
10 where GROUP_ID = 'G1' and EDGE_NODE_ID = 'E1' and DEVICE_NAME = 'Motor 3'
```

Below the query, the results are shown in a table with 10 columns: GROUP_ID, EDGE_NODE_ID, DEVICE_ID, DEVICE_NAME, MEASURE_TIMESTAMP, TEMPERATURE, RPMS, AMPS, and INSERTED_AT. The table contains 5 rows of data for the specified filter.

GROUP_ID	EDGE_NODE_ID	DEVICE_ID	DEVICE_NAME	MEASURE_TIMESTAMP	TEMPERATURE	RPMS	AMPS	INSERTED_AT
G1	E1	PLC 1	Motor 3	1730848316753	49	1202	121	1730848515389
G1	E1	PLC 1	Motor 3	1730848316753	49	null	121	1730848515389
G1	E1	PLC 1	Motor 3	1730848316753	49	null	null	1730848515389
G1	E1	PLC 1	Motor 3	1730848316753	49	null	null	1730848515389
G1	E1	PLC 1	Motor 3	1730848316753	49	null	null	1730848515389

On the right side, there are filters for GROUP_ID (G1), EDGE_NODE_ID (E1), and DEVICE_ID (PLC 1). A 'Query Details' panel shows a query duration of 134ms and 5 rows returned. An 'Ask Copilot' button is also visible.