IBSNOW: Snowflake Setup Scripts v1.0.0 through v1.0.3

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.

This document contains the Snowflake setup scripts to be copied into and run in the Snowflake SQL worksheet.

Click the 'Expand source' button on the right to copy the script source code.

SQL Script 01

Set up assets related to the staging database and associated assets. These are:

- Database
- Staging schema

SQL Script 01

```
-- ========================
-- In this script, we are setting up assets related to the staging database
-- and associated assets. These are:
--  - Database
--  - Staging schema

-- The database & schema will be owned by SYSADMIN

-- REPLACE THE SESSION VARIABLE ACCORDING TO YOUR ENVIRONMENT
-- ========================

set cl_bridge_staging_db = 'CL_BRIDGE_STAGE_DB';
set staging_schema = 'stage_db';

-- >>>>>>>>>>>>>>>>>>>>>> DATABASE >>>>>>>>>>>>>>>>>>>>>>>>

use role sysadmin;

create database if not exists identifier($cl_bridge_staging_db)
   -- DATA_RETENTION_TIME_IN_DAYS = 90
   -- MAX_DATA_EXTENSION_TIME_IN_DAYS = 90
   comment = 'used for storing messages received from CirrusLink Bridge'
;

-- >>>>>>>>>>>>>>>>>>>>>> STAGING SCHEMA >>>>>>>>>>>>>>>>>>>>>>>>

use database identifier($cl_bridge_staging_db);

create schema if not exists identifier($staging_schema)
  with managed access
  -- data_retention_time_in_days = 90
  -- max_data_extension_time_in_days = 90
  comment = 'Used for staging data direct from CirrusLink Bridge';

-- >>>>>>>>>>>>>>>>>>>>>> STAGING SCHEMA ASSETS >>>>>>>>>>>>>>>>>>>>>>>>

use schema identifier($staging_schema);

-- ========================
-- Define tables
-- ========================

create or replace table sparkplug_raw (
    msg_id varchar
    ,msg_topic varchar
    ,namespace varchar
    ,group_id varchar
    ,message_type varchar
    ,edge_node_id varchar
    ,device_id varchar
    ,msg variant
    ,inserted_at number
)
change_tracking = true
cluster by (message_type ,group_id ,edge_node_id ,device_id)
comment = 'Used for storing json messages from sparkplug bridge/gateway'
;
```

SQL Script 02

Set up assets related to the node database which would eventually contain all the device specific views and tables. These are:

- Node Database
- Staging schema

SQL Script 02

```
-- ========================
-- In this script, we are setting up assets related to the node database
-- ,which would eventually contain all the device specific views and tables.
-- At the very core, the following assets are created:
--  - Node Database
--  - Staging schema

-- The database & schema will be owned by SYSADMIN

-- REPLACE THE SESSION VARIABLE ACCORDING TO YOUR ENVIRONMENT
-- ========================

set staged_sparkplug_raw_table = 'cl_bridge_stage_db.stage_db.sparkplug_raw';
set cl_bridge_node_db = 'cl_bridge_node_db';
set staging_schema = 'stage_db';

-- >>>>>>>>>>>>>>>>>>>>> DATABASE >>>>>>>>>>>>>>>>>>>>>>>>

use role sysadmin;

create database if not exists identifier($cl_bridge_node_db)
   -- DATA_RETENTION_TIME_IN_DAYS = 90
   -- MAX_DATA_EXTENSION_TIME_IN_DAYS = 90
   comment = 'used for storing flattened messages processed from the staging database'
;

-- >>>>>>>>>>>>>>>>>>>>> STAGING SCHEMA >>>>>>>>>>>>>>>>>>>>>>>>

use database identifier($cl_bridge_node_db);

create schema if not exists identifier($staging_schema)
  with managed access
  -- data_retention_time_in_days = 90
  -- max_data_extension_time_in_days = 90
  comment = 'used for storing flattened messages processed from the staging database';

-- >>>>>>>>>>>>>>>>>>>>> STAGING SCHEMA ASSETS >>>>>>>>>>>>>>>>>>>>>>>>

use schema identifier($staging_schema);

-- ========================
-- Define tables
-- ========================

-- NOTE THE 'cl_bridge_stage_db.stage_db.sparkplug_raw' is hardcoded here; as the identifier
-- staged_sparkplug_raw_table replacement does not work.

create or replace view sparkplug_messages_vw
    change_tracking = true
    comment = 'parses out the core attributes from the message and topic.'
    as
    select
        msg_id
        ,namespace
        ,group_id
        ,message_type
        ,edge_node_id
        ,device_id
        ,parse_json(msg) as message
        ,message:seq::int as message_sequence
        ,message:timestamp::number as message_timestamp
        ,inserted_at
    from cl_bridge_stage_db.stage_db.sparkplug_raw
    ;

-- --  >>>>>>>>>>>>>>>>>>>>>

create or replace view nbirth_vw
```

```
    change_tracking = true
    comment = 'filtered to nbirth messages. This is a mirror'
    as
    select
       group_id ,edge_node_id
    from sparkplug_messages_vw
    where message_type = 'NBIRTH'

    ;

create or replace view node_machine_registry_vw
    comment = 'Used to retreive the latest template definitions for a given group and edge_node'
    as
    with base as (
        select
            group_id ,edge_node_id
            ,max_by(message ,message_timestamp) as message
            ,max(message_timestamp) as latest_message_timestamp
        from sparkplug_messages_vw
        where message_type = 'NBIRTH'
        group by group_id ,edge_node_id
    )
    select
        group_id ,edge_node_id
        ,f.value as template_definition
        ,template_definition:name::varchar as machine
        ,template_definition:reference::varchar as reference
        ,template_definition:version::varchar as version
        ,template_definition:timestamp::int as timestamp
    from base as b
        ,lateral flatten (input => b.message:metrics) f
    where template_definition:dataType::varchar = 'Template'
    ;

-- --  >>>>>>>>>>>>>>>>>>>>>>

create or replace view node_birth_death_vw
    comment = 'shows the latest node birth & death messages for each device'
    as
    select
        b.* exclude(namespace)
        ,message_type as nbirth_or_ndeath_raw
        ,iff((message_type = 'NBIRTH') ,f.value:value ,null)::number as nbirth_bdSeq_raw
        ,iff((message_type = 'NDEATH') ,f.value:value ,null)::number as ndeath_bdSeq_raw
        ,inserted_at as nbirth_ndeath_inserted_at_raw
    from sparkplug_messages_vw as b
        ,lateral flatten (input => b.message:metrics) as f
    where message_type in ('NBIRTH' ,'NDEATH')
     and f.value:name::varchar = 'bdSeq'
     ;

create or replace view device_records_vw
    change_tracking = true
    as
    select
        b.* exclude(namespace)
        ,null as nbirth_or_ndeath_raw
        ,null as nbirth_bdSeq_raw
        ,null as ndeath_bdSeq_raw
        ,null as nbirth_ndeath_inserted_at_raw
    from sparkplug_messages_vw as b
    where message_type in ('DBIRTH' ,'DDATA')
    ;

create or replace stream device_records_stream
    on view device_records_vw

    show_initial_rows = true
    comment = 'used for monitoring latest device messages'
    ;
```

```
create or replace view sparkplug_msgs_nodebirth_contextualized_vw
    as
    with device_node_unioned as (
        select *
        from node_birth_death_vw
        union all
        select * exclude(METADATA$ROW_ID ,METADATA$ACTION ,METADATA$ISUPDATE)
        from device_records_stream
    )
    select
        -- group_id ,message_type ,edge_node_id ,device_id
        -- ,message ,message_sequence ,inserted_at
        * exclude(nbirth_or_ndeath_raw ,nbirth_bdSeq_raw ,ndeath_bdSeq_raw ,nbirth_ndeath_inserted_at_raw)
        ,nvl(nbirth_or_ndeath_raw
                ,lag(nbirth_or_ndeath_raw) ignore nulls over (order by inserted_at ,message_sequence)
            ) as nbirth_or_ndeath

        ,nvl(nbirth_bdSeq_raw
                ,lag(nbirth_bdSeq_raw) ignore nulls over (order by inserted_at ,message_sequence)
            ) as nbirth_bdSeq

        ,nvl(ndeath_bdSeq_raw
                ,lag(ndeath_bdSeq_raw) ignore nulls over (order by inserted_at ,message_sequence)
            ) as ndeath_bdSeq

        ,nvl(nbirth_ndeath_inserted_at_raw
                ,lag(nbirth_ndeath_inserted_at_raw) ignore nulls over (order by inserted_at ,message_sequence)
            ) as nbirth_ndeath_inserted_at

        ,case true
            when (nbirth_or_ndeath = 'NBIRTH') then false
            when ( (nbirth_or_ndeath = 'NDEATH') and (nbirth_bdSeq != ndeath_bdSeq) ) then false
            when ( (nbirth_or_ndeath = 'NDEATH') and (nbirth_bdSeq = ndeath_bdSeq) ) then true
            else true
        end as is_last_known_good_reading

        ,case lower(message_type)
                when lower('NBIRTH') then 1
                when lower('DBIRTH') then 2
                when lower('DDATA') then 3
                when lower('DDEATH') then 4
                when lower('NDEATH') then 5
                else 99
            end as message_type_order

        ,(nbirth_or_ndeath = 'NBIRTH') as is_node_alive

    from device_node_unioned
    ;

create or replace view sparkplug_messages_flattened_vw
    as
    with base as (
        select
            -- sparkplugb message level
            msg_id ,group_id, edge_node_id ,device_id ,message_type
            ,message_sequence ,inserted_at
            ,nbirth_or_ndeath ,nbirth_bdseq ,ndeath_bdseq
            ,nbirth_ndeath_inserted_at ,is_last_known_good_reading
            ,message_type_order ,is_node_alive
            ,message_timestamp as root_message_timestamp

            -- attributes related to device data (ddata / dbirth)
            ,f.value:name::varchar as device_name
            ,f.value:value:reference::varchar as template_reference
            ,f.value:value:version::varchar as template_version
            ,f.value:timestamp::number as device_metric_timestamp
            ,f.value as ddata_msg

            -- attributes related to device level metrics
            ,concat(msg_id ,'^' ,f.index ,'::',d.index) as device_measure_uuid
```

```
            ,d.value:name::varchar as measure_name
            ,d.value:value as measure_value
            ,d.value:timestamp::number as measure_timestamp

        from sparkplug_msgs_nodebirth_contextualized_vw as b
            ,lateral flatten(input => b.message:metrics) as f
            ,lateral flatten(input => f.value:value:metrics) as d
        where message_type in ('DBIRTH' ,'DDATA')
            and template_reference is not null
    )
    select
        group_id, edge_node_id ,device_id ,message_type
        ,message_sequence ,inserted_at
        ,nbirth_or_ndeath ,nbirth_bdseq ,ndeath_bdseq
        ,nbirth_ndeath_inserted_at ,is_last_known_good_reading
        ,message_type_order ,is_node_alive ,root_message_timestamp

        ,device_name ,template_reference ,template_version ,device_metric_timestamp ,ddata_msg
        ,null as is_historical

        ,device_measure_uuid
        ,object_agg(distinct measure_name ,measure_value) as measures_info
        ,measure_timestamp

        ,to_timestamp(measure_timestamp/1000) as measure_ts
        ,to_date(measure_ts) as measure_date
        ,hour(measure_ts) as measure_hour
    from base
    group by group_id, edge_node_id ,device_id ,message_type
            ,message_sequence ,inserted_at
            ,nbirth_or_ndeath ,nbirth_bdseq ,ndeath_bdseq
            ,nbirth_ndeath_inserted_at ,is_last_known_good_reading
            ,message_type_order ,is_node_alive ,root_message_timestamp

            ,device_name ,template_reference ,template_version ,device_metric_timestamp ,ddata_msg
            ,is_historical ,device_measure_uuid
            ,measure_timestamp

    ;

create or replace transient table sparkplug_device_messages (
        group_id varchar
        ,edge_node_id varchar
        ,device_id varchar
        ,message_type varchar
        ,message_sequence number

        ,inserted_at number
        ,nbirth_or_ndeath varchar
        ,nbirth_bdseq number
        ,ndeath_bdseq number
        ,nbirth_ndeath_inserted_at number
        ,is_last_known_good_reading boolean
        ,message_type_order number
        ,is_node_alive boolean

        ,root_message_timestamp number
        ,device_name varchar
        ,template_reference varchar
        ,template_version varchar
        ,device_metric_timestamp number
        ,ddata_msg variant
        ,is_historical boolean

        ,device_measure_uuid varchar
        ,measures_info variant
        ,measure_timestamp number

        ,measure_ts timestamp
        ,measure_date date
        ,measure_hour number
```

```
    )
    cluster by ( group_id ,edge_node_id ,device_id
        ,template_reference ,template_version ,device_name
        ,measure_date ,measure_hour)
    comment = 'materialized version of the sparkplug_messages_flattened_vw for easier downstream pipelines.'
    ;


-- --  >>>>>>>>>>>>>>>>>>>>>
-- ================
--  NODE BIRTH related assets
-- ================

create or replace stream nbirth_stream
    on view nbirth_vw

    show_initial_rows = true
    comment = 'stream to monitor for nbirth messages, so that assets are created automatically'
    ;
```

SQL Script 03

Creates stored procedures called by the IoT Bridge

SQL Script 03

```
set cl_bridge_node_db = 'cl_bridge_node_db';
set staging_schema = 'stage_db';

use role sysadmin;
use database identifier($cl_bridge_node_db);
use schema identifier($staging_schema);

CREATE OR REPLACE PROCEDURE synch_device_messages()
    RETURNS VARIANT NOT NULL
    LANGUAGE JAVASCRIPT
    COMMENT = 'Synch latest device updates and stores in table'
    AS
    $$

    // --- MAIN -------------------------------------
    var failure_err_msg = [];
    var return_result_as_json = {};
    var sucess_count = 0;
    var failure_count = 0;

    const qry = `
    insert into sparkplug_device_messages
    select *
    from sparkplug_messages_flattened_vw
    ;`

    res = [];
    try {
        var rs = snowflake.execute({ sqlText: qry });
        sucess_count = sucess_count + 1;

    } catch (err) {
        failure_count = failure_count + 1;
        failure_err_msg.push(` {
        sqlstatement : '${qry}',
            error_code : '${err.code}',
            error_state : '${err.state}',
            error_message : '${err.message}',
            stack_trace : '${err.stackTraceTxt}'
            } `);
    }
```

```
    return_result_as_json['asset_creation'] = res;
    return_result_as_json['Success'] = sucess_count;
    return_result_as_json['Failures'] = failure_count;
    return_result_as_json['Failure_error'] = failure_err_msg;
    return return_result_as_json;
    $$;

CREATE OR REPLACE FUNCTION GENERATE_TEMPLATE_ASSET_BASE_NAME
    (PARAM_TEMPLATE_NAME varchar ,PARAM_TEMPLATE_VERSION varchar)
  RETURNS VARCHAR
  LANGUAGE JAVASCRIPT
  COMMENT = 'Used for generating device template name.'
  AS $$
    function normalize_name_str(p_str) {
        return p_str.replace(/[\W_]+/g,"_").trim().toLowerCase();
    }

    function get_device_view_base_name(p_machine ,p_version) {
        const v = (p_version != null) ? p_version : "";
        return normalize_name_str(`${p_machine}${v}`);
    }


  return get_device_view_base_name(PARAM_TEMPLATE_NAME ,PARAM_TEMPLATE_VERSION);
  $$
  ;


CREATE OR REPLACE FUNCTION GENERATE_DEVICE_BASE_VIEW_DDL
    (   PARAM_GROUP_ID VARCHAR ,PARAM_EDGE_NODE_ID VARCHAR ,PARAM_TEMPLATE_REFERENCE VARCHAR
        ,PARAM_SOURCE_DB varchar
        ,PARAM_SOURCE_SCHEMA varchar
        ,PARAM_TARGET_SCHEMA VARCHAR
        ,PARAM_TEMPLATE_ASSET_BASE_NAME varchar
        ,PARAM_TEMPLATE_DEFN variant
        ,PARAM_FORCE_RECREATE boolean
        )
  RETURNS VARCHAR
  LANGUAGE JAVASCRIPT
  COMMENT = 'Used for generating generic view ddl for device template.'
  AS $$

    var stmt_condition = `create view if not exists`;
    if (PARAM_FORCE_RECREATE == true)
        stmt_condition = `create or replace view`;

    const sql_stmt = `
        ${stmt_condition} ${PARAM_SOURCE_DB}.${PARAM_TARGET_SCHEMA}.${PARAM_TEMPLATE_ASSET_BASE_NAME}
        as
        select
            group_id ,edge_node_id ,device_id ,message_type ,message_sequence ,root_message_timestamp
            ,inserted_at ,nbirth_or_ndeath ,nbirth_bdseq  ,ndeath_bdseq
            ,nbirth_ndeath_inserted_at  ,is_last_known_good_reading
            ,message_type_order ,is_node_alive

            ,device_name ,template_version ,device_metric_timestamp ,ddata_msg ,is_historical

            ,device_measure_uuid ,measures_info ,measure_timestamp

            ,measure_ts ,measure_date ,measure_hour

        from ${PARAM_SOURCE_SCHEMA}.sparkplug_device_messages
        where group_id = '${PARAM_GROUP_ID}'
            and edge_node_id = '${PARAM_EDGE_NODE_ID}'
            and template_reference = '${PARAM_TEMPLATE_REFERENCE}'
        ;
    `;

    return sql_stmt;
  $$
  ;
```

```
CREATE OR REPLACE FUNCTION GENERATE_DEVICE_VIEW_DDL
    (   PARAM_GROUP_ID VARCHAR ,PARAM_EDGE_NODE_ID VARCHAR ,PARAM_TEMPLATE_REFERENCE VARCHAR
        ,PARAM_SOURCE_DB varchar
        ,PARAM_TARGET_SCHEMA VARCHAR
        ,PARAM_TEMPLATE_ASSET_BASE_NAME varchar
        ,PARAM_TEMPLATE_DEFN variant
        ,PARAM_FORCE_RECREATE boolean
        )
  RETURNS VARIANT
  LANGUAGE JAVASCRIPT
  COMMENT = 'Used for generating generic view ddl for device template.'
  AS $$

    function normalize_name_str(p_str) {
        return p_str.replace(/[\W_]+/g,"_").trim().toLowerCase();
    }

    function build_column_ddl_defn(p_template_defn ,p_suffix) {
        var cols = [];
        const data_type_map = {
            "Int32":"::integer"
            ,"Int64":"::integer"
            ,"Float":"::double"
            ,"Template":"::variant"
            ,"Boolean":"::boolean"
            ,"String":"::varchar"
            };

        const m_entries = p_template_defn['value']['metrics']
        for (const [m_key, m_value] of Object.entries(m_entries)) {

            const measure_name = m_value['name'];
            const dtype = m_value['dataType'];

            const mname_cleansed = normalize_name_str(measure_name) + p_suffix;
            //     # default string cast, if the datatype is not mapped
            const dtype_converted = data_type_map[dtype] || "::varchar";

            const col_defn = `measures_info:"${measure_name}"${dtype_converted} as ${mname_cleansed} `;
            cols.push(col_defn);
        }
        /* in some cases client have defined UDT with no tags set. there seems
        to be a valid use case if they are only using the UDT to transmit UDT parameters */
        const cols_joined = (cols.length > 0) ? ',' + cols.join(',') : '';
        return cols_joined
    }

    const vw_name = `${PARAM_TEMPLATE_ASSET_BASE_NAME}_vw`;
    const cols_joined = build_column_ddl_defn(PARAM_TEMPLATE_DEFN ,'')
    const cols_joined_alternate = build_column_ddl_defn(PARAM_TEMPLATE_DEFN ,'_')

    const sql_stmt = `
        create or replace view ${PARAM_SOURCE_DB}.${PARAM_TARGET_SCHEMA}.${vw_name}
        as
        select
            * exclude(ddata_msg  ,measures_info ,template_version)
            ${cols_joined}
        from ${PARAM_SOURCE_DB}.${PARAM_TARGET_SCHEMA}.${PARAM_TEMPLATE_ASSET_BASE_NAME}
        ;
    `;

    /*metric name can endup being reserved keywords (ex: trigger), this
    can result in an error during view creation. to overcome this
    we suffix the column with a '_' and return the create ddl as
    an alternate sql statement. it is expected the caller will use this
    alternate sql statement if the first/default/primary sql statement fails
    */
    const sql_stmt_alternate = `
```

```
        create or replace view ${PARAM_SOURCE_DB}.${PARAM_TARGET_SCHEMA}.${vw_name}
        as
        select
            * exclude(ddata_msg  ,measures_info ,template_version)
            ${cols_joined_alternate}
        from ${PARAM_SOURCE_DB}.${PARAM_TARGET_SCHEMA}.${PARAM_TEMPLATE_ASSET_BASE_NAME}
        ;
    `;

    return [sql_stmt ,sql_stmt_alternate];
$$;

CREATE OR REPLACE FUNCTION GENERATE_DEVICE_ASOF_VIEW_DDL
    (   PARAM_GROUP_ID VARCHAR ,PARAM_EDGE_NODE_ID VARCHAR ,PARAM_TEMPLATE_REFERENCE VARCHAR
        ,PARAM_SOURCE_DB varchar
        ,PARAM_TARGET_SCHEMA VARCHAR
        ,PARAM_TEMPLATE_ASSET_BASE_NAME varchar
        ,PARAM_TEMPLATE_DEFN variant
        ,PARAM_FORCE_RECREATE boolean
        )
  RETURNS VARIANT
  LANGUAGE JAVASCRIPT
  COMMENT = 'Used for generating device asof view ddl.'
  AS $$

    function normalize_name_str(p_str) {
        return p_str.replace(/[\W_]+/g,"_").trim().toLowerCase();
    }

    function build_column_ddl_defn(p_template_defn ,p_suffix) {
        var cols = []

        const data_type_map = {
                "Int32":"::integer"
                ,"Int64":"::integer"
                ,"Float":"::double"
                ,"Template":"::variant"
                ,"Boolean":"::boolean"
                ,"String":"::varchar"
                }

        const m_entries = p_template_defn['value']['metrics']
        for (const [m_key, m_value] of Object.entries(m_entries)) {

            const measure_name = m_value['name'];
            const dtype = m_value['dataType'];

            const mname_cleansed = normalize_name_str(measure_name) + p_suffix;
            //     # default string cast, if the datatype is not mapped
            const dtype_converted = data_type_map[dtype] || "::varchar";

            const col_defn = `nvl(${mname_cleansed}
                    ,lag(${mname_cleansed}) ignore nulls over (
                        partition by device_id ,device_name
                        order by message_type_order ,measure_timestamp ,message_sequence)
                ) AS ${mname_cleansed}
            `;

            cols.push(col_defn);
        }

        /* in some cases client have defined UDT with no tags set. there seems
        to be a valid use case if they are only using the UDT to transmit UDT parameters */
        const cols_joined = (cols.length > 0) ? ',' + cols.join(',') : '';
        return cols_joined
    }

    const vw_name = `${PARAM_TEMPLATE_ASSET_BASE_NAME}_vw`;
    const recordasof_vw_name = `${PARAM_TEMPLATE_ASSET_BASE_NAME}_asof_vw`;
    const cols_joined = build_column_ddl_defn(PARAM_TEMPLATE_DEFN,'');
    const cols_joined_alternate = build_column_ddl_defn(PARAM_TEMPLATE_DEFN,'_');
```

```
    const sql_stmt = `
        create or replace secure view ${PARAM_SOURCE_DB}.${PARAM_TARGET_SCHEMA}.${recordasof_vw_name}
        as
        select
                group_id ,edge_node_id ,device_id ,device_name ,message_sequence
                ,root_message_timestamp ,inserted_at

                ,message_type ,message_type_order

                ,nbirth_or_ndeath ,nbirth_bdseq  ,ndeath_bdseq
                ,nbirth_ndeath_inserted_at  ,is_last_known_good_reading
                ,is_node_alive
                ,is_historical

                ,device_measure_uuid
                ,measure_timestamp
                ,measure_ts ,measure_date ,measure_hour

                ${cols_joined}
            from ${PARAM_SOURCE_DB}.${PARAM_TARGET_SCHEMA}.${vw_name}
            order by measure_timestamp ,message_type_order ,message_sequence
            ;
    `;

    /*metric name can endup being reserved keywords (ex: trigger), this
    can result in an error during view creation. to overcome this
    we suffix the column with a '_' and return the create ddl as
    an alternate sql statement. it is expected the caller will use this
    alternate sql statement if the first/default/primary sql statement fails
    */
    const sql_stmt_alternate = `
        create or replace secure view ${PARAM_SOURCE_DB}.${PARAM_TARGET_SCHEMA}.${recordasof_vw_name}
        as
        select
                group_id ,edge_node_id ,device_id ,device_name ,message_sequence
                ,root_message_timestamp ,inserted_at

                ,message_type ,message_type_order

                ,nbirth_or_ndeath ,nbirth_bdseq  ,ndeath_bdseq
                ,nbirth_ndeath_inserted_at  ,is_last_known_good_reading
                ,is_node_alive
                ,is_historical

                ,device_measure_uuid
                ,measure_timestamp
                ,measure_ts ,measure_date ,measure_hour

                ${cols_joined_alternate}
            from ${PARAM_SOURCE_DB}.${PARAM_TARGET_SCHEMA}.${vw_name}
            order by measure_timestamp ,message_type_order ,message_sequence
            ;
    `;

    return [sql_stmt ,sql_stmt_alternate];
$$;
```

SQL Script 04

Creates user defined functions called by the stored procedures created in SQL Script 03

SQL Script 04

```
set cl_bridge_node_db = 'cl_bridge_node_db';
set staging_schema = 'stage_db';

use role sysadmin;
```

```
use database identifier($cl_bridge_node_db);
use schema identifier($staging_schema);

CREATE OR REPLACE FUNCTION NORMALIZE_ASSET_NAME(P_STR VARCHAR)
  RETURNS VARCHAR
  LANGUAGE JAVASCRIPT
  COMMENT = 'Used for creating asset names without spaces/special characters.'
  AS $$
  return P_STR.replace(/[\W_]+/g,"_").trim().toLowerCase();
  $$
  ;

CREATE OR REPLACE FUNCTION EDGENODE_SCHEMA_NAME(PARAM_SCHEMA_PREFIX VARCHAR ,PARAM_GROUP_ID VARCHAR ,
PARAM_EDGE_NODE_ID VARCHAR)
  RETURNS VARCHAR
  LANGUAGE JAVASCRIPT
  COMMENT = 'Used for creating asset names for edgenode schema.'
  AS $$

    function normalize_name_str(p_str) {
        return p_str.replace(/[\W_]+/g,"_").trim().toLowerCase();
    }
    const schema_name = `${PARAM_SCHEMA_PREFIX}_${PARAM_GROUP_ID}_${PARAM_EDGE_NODE_ID}`;
    return normalize_name_str(schema_name);
  $$
  ;

CREATE OR REPLACE FUNCTION GENERATE_EDGENODE_SCHEMA_DDL
    (PARAM_SCHEMA_PREFIX VARCHAR ,PARAM_GROUP_ID VARCHAR
        ,PARAM_EDGE_NODE_ID VARCHAR ,PARAM_FORCE_RECREATE boolean)
  RETURNS VARCHAR
  LANGUAGE JAVASCRIPT
  COMMENT = 'Used for generating edgenode schema ddl.'
  AS $$

    function normalize_name_str(p_str) {
        return p_str.replace(/[\W_]+/g,"_").trim().toLowerCase();
    }

    // Returns the normalized schema name for the edgenode
    function get_edgenode_schema_name(p_schema_prefix ,p_group_id ,p_edge_node_id) {
        const schema_name = `${p_schema_prefix}_${p_group_id}_${p_edge_node_id}`;
        return normalize_name_str(schema_name);
    }

    const schema_name = get_edgenode_schema_name(PARAM_SCHEMA_PREFIX  ,PARAM_GROUP_ID  ,PARAM_EDGE_NODE_ID);
    var sql_stmt = `create schema if not exists ${schema_name}; `
    if(PARAM_FORCE_RECREATE == true) {
        sql_stmt = `create or replace schema ${schema_name}; `
    }
    return sql_stmt;
  $$
  ;


CREATE OR REPLACE PROCEDURE create_edge_node_schema(PARAM_SCHEMA_PREFIX VARCHAR ,PARAM_GROUP_ID VARCHAR ,
PARAM_EDGE_NODE_ID VARCHAR ,PARAM_FORCE_RECREATE boolean)
RETURNS VARIANT NOT NULL
LANGUAGE JAVASCRIPT
COMMENT = 'Creates edge node specific schema, supposed to be invoked as part of NBIRTH messsage'
AS
$$

    function get_sql_stmt(p_schema_prefix ,p_group_id ,p_edge_node_id ,p_force_recreate) {
        const sql_stmt = `select
            '${p_schema_prefix}' as schema_prefix
            ,iff(${p_force_recreate} = 1 ,true ,false) as force_recreate
            ,edgenode_schema_name(schema_prefix ,group_id ,edge_node_id) as edgenode_schema_name

            ,generate_edgenode_schema_ddl(schema_prefix, group_id ,edge_node_id ,force_recreate) as
edgenode_schema_ddl
```

```
        ,machine
        ,generate_template_asset_base_name(machine ,version) as machine_table_base_name
        ,template_definition

        ,generate_device_base_view_ddl
            (group_id ,edge_node_id  ,machine
                ,current_database() ,current_schema() ,edgenode_schema_name
                ,machine_table_base_name ,template_definition
                ,force_recreate) as device_base_view_ddl

        ,generate_device_view_ddl
            (group_id ,edge_node_id  ,machine
                ,current_database() ,edgenode_schema_name
                ,machine_table_base_name ,template_definition
                ,force_recreate) as device_view_ddl

        ,generate_device_asof_view_ddl
            (group_id ,edge_node_id  ,machine
                ,current_database() ,edgenode_schema_name
                ,machine_table_base_name ,template_definition
                ,force_recreate) as device_asof_view_ddl

        from node_machine_registry_vw
        where group_id = '${p_group_id}'
            and edge_node_id = '${p_edge_node_id}'
        ;`

    return sql_stmt;
}

// --- MAIN ----------------------------------
var failure_err_msg = [];
var return_result_as_json = {};
var sucess_count = 0;
var failure_count = 0;
var alternate_views = [];
var view_not_created = [];

const qry = get_sql_stmt(PARAM_SCHEMA_PREFIX ,PARAM_GROUP_ID ,PARAM_EDGE_NODE_ID ,PARAM_FORCE_RECREATE);
var sql_stmt = qry;
var schema_created = false;
res = [];
try {
    var rs = snowflake.execute({ sqlText: qry });
    while (rs.next()) {
        machine  = rs.getColumnValue('MACHINE');
        edgenode_schema_name = rs.getColumnValue('EDGENODE_SCHEMA_NAME');
        edgenode_schema_ddl = rs.getColumnValue('EDGENODE_SCHEMA_DDL');
        device_base_view_ddl = rs.getColumnValue('DEVICE_BASE_VIEW_DDL');
        device_view_ddl = rs.getColumnValue('DEVICE_VIEW_DDL');
        device_asof_view_ddl = rs.getColumnValue('DEVICE_ASOF_VIEW_DDL');

        if(schema_created == false) {
            sql_stmt = edgenode_schema_ddl;
            snowflake.execute({ sqlText: edgenode_schema_ddl });
            //blind setting is not good,
            //TODO check if the schema is indeed created
            schema_created = true;
        }

        try {
            sql_stmt = device_base_view_ddl;
            snowflake.execute({ sqlText: device_base_view_ddl });
        } catch (err) {
            /*This is safety to handle scenarion where the
            base view is not able to be created. in these cases
            for now we just capture this and ignore the rest of the proces
            for this machine*/
            failure_count = failure_count + 1;
            failure_err_msg.push(` {
                sqlstatement : '${sql_stmt}',
```

```
                error_code : '${err.code}',
                error_state : '${err.state}',
                error_message : '${err.message}',
                stack_trace : '${err.stackTraceTxt}'
                } `);

            continue;
        }

        try {
         /*
         try creating the view using the default, if the
         view creation fails; we assume that this is probably
         due to reserved keyword (ex: trigger) as being defined
         as metric name. in such cases this view creation will result
         in a failure. the alternate sql will be executed for these
         views
         */
            sql_stmt = device_view_ddl[0];
            snowflake.execute({ sqlText: sql_stmt });
        } catch (err) {
            /*alternate view creation which has the fix
            */
            alternate_views.push(machine)
            alternate_view_ddl = device_view_ddl[1];
            snowflake.execute({ sqlText: alternate_view_ddl });
        }

        try {
         /*
         try creating the view using the default, if the
         view creation fails; we assume that this is probably
         due to reserved keyword (ex: trigger) as being defined
         as metric name. in such cases this view creation will result
         in a failure. the alternate sql will be executed for these
         views
         */
            sql_stmt = device_asof_view_ddl[0];
            snowflake.execute({ sqlText: sql_stmt });
        } catch (err) {
            /*alternate view creation which has the fix
            */
            alternate_view_ddl = device_asof_view_ddl[1];
            snowflake.execute({ sqlText: alternate_view_ddl });
        }

        sucess_count = sucess_count + 1;
        res.push(edgenode_schema_name + '.' + machine);
    }

} catch (err) {
    failure_count = failure_count + 1;
    failure_err_msg.push(` {
        sqlstatement : '${sql_stmt}',
        error_code : '${err.code}',
        error_state : '${err.state}',
        error_message : '${err.message}',
        stack_trace : '${err.stackTraceTxt}'
        } `);
}

return_result_as_json['asset_creation'] = res;
 return_result_as_json['Success'] = sucess_count;
 return_result_as_json['Failures'] = failure_count;
 return_result_as_json['Failure_error'] = failure_err_msg;
 return_result_as_json['aternate_views'] = alternate_views;

return return_result_as_json;
$$;
```

SQL Script 05

Creates stored procedures called by the IoT Bridge

SQL Script 05

```
set cl_bridge_node_db = 'cl_bridge_node_db';
set staging_schema = 'stage_db';

use role sysadmin;
use database identifier($cl_bridge_node_db);
use schema identifier($staging_schema);


CREATE OR REPLACE PROCEDURE create_all_edge_node_schemas(PARAM_SCHEMA_PREFIX VARCHAR ,PARAM_FORCE_RECREATE
boolean)
RETURNS VARIANT NOT NULL
LANGUAGE JAVASCRIPT
COMMENT = 'Creates edge node specific schemas, supposed to be invoked as part of NBIRTH messsage'
AS
$$

// --- MAIN ---------------------------------------
var failure_err_msg = [];
var return_result_as_json = {};
var sucess_count = 0;
var failure_count = 0;

const qry = `
select distinct group_id ,edge_node_id
    ,current_schema() as current_schema
from node_machine_registry_vw
where edge_node_id is not null
;`
//node_machine_registry_vw
//nbirth_stream

var current_schema = 'stage_db';
res = [];
try {
    var rs = snowflake.execute({ sqlText: qry });
    while (rs.next()) {
        group_id = rs.getColumnValue('GROUP_ID');
        edge_node_id = rs.getColumnValue('EDGE_NODE_ID');
        current_schema = rs.getColumnValue('CURRENT_SCHEMA');

        var schema_out = {}

        schema_out['execution'] = snowflake.execute({
                    sqlText: `call create_edge_node_schema('${PARAM_SCHEMA_PREFIX}' ,'${group_id}'
,'${edge_node_id}' ,${PARAM_FORCE_RECREATE});`
                });

        res.push(schema_out);
     }
    sucess_count = sucess_count + 1;

} catch (err) {
    failure_count = failure_count + 1;
    failure_err_msg.push(` {
    sqlstatement : '${qry}',
        error_code : '${err.code}',
        error_state : '${err.state}',
        error_message : '${err.message}',
        stack_trace : '${err.stackTraceTxt}'
        } `);
}

return_result_as_json['asset_creation'] = res;
return_result_as_json['Success'] = sucess_count;
return_result_as_json['Failures'] = failure_count;
return_result_as_json['Failure_error'] = failure_err_msg;
return return_result_as_json;
$$;
```

SQL Script 06

Creates dynamic tables for BIRTH tracking

SQL Script 06

```
set cl_bridge_node_db = 'cl_bridge_node_db';
set staging_schema = 'stage_db';

set reader_role_warehouse = 'compute_wh';

use role sysadmin;
use database identifier($cl_bridge_node_db);
use schema identifier($staging_schema);


create or replace dynamic table D_ACTIVE_NBIRTH
    lag = '1 day'
    warehouse = compute_wh
    as
    -- Tabularize the active nbirth events for easier error handling during
    -- investigations
    with base as (
        select
            group_id ,edge_node_id ,device_id
                ,nvl(nbirth_bdseq_raw ,ndeath_bdseq_raw ) as bdseq
                ,message_type
                ,to_timestamp(message_timestamp/1000) as MEASURE_TS
        from NODE_BIRTH_DEATH_VW

    ), nbirth_ndeath_matched as (
        select distinct nb.bdseq
        from base as nb
            join base as nd
                on nb.bdseq = nd.bdseq
        where nb.message_type = 'NBIRTH'
            and nd.message_type = 'NDEATH'
    )
    select b.* exclude(measure_ts ,message_type)
        ,min(measure_ts) as nbirth
    from base as b
    where b.bdseq not in (select bdseq from nbirth_ndeath_matched)
    group by all
    ;

create or replace dynamic table D_DEVICE_HEARTBEATS
    lag = '1 day'
    warehouse = compute_wh
    as
    -- tabularize the dbirth/data messages for each devices. Meant to be used
    -- for active investigation on message receipt from devices and error tracking
    with base as (
        select group_id ,edge_node_id ,device_id ,DEVICE_NAME ,message_type ,MEASURE_TS
            ,row_number()
                over ( partition by group_id ,edge_node_id ,device_id ,DEVICE_NAME ,message_type
                        order by MEASURE_TS desc
                ) as row_num

        from SPARKPLUG_DEVICE_MESSAGES

    ), rows_filtered as (
        select *
        from base
        where row_num <= 2
    ), object_constructed as (
        select group_id ,edge_node_id ,device_id ,DEVICE_NAME ,message_type
            ,first_value(MEASURE_TS)
                ignore nulls
                over ( partition by group_id ,edge_node_id ,device_id ,DEVICE_NAME ,message_type
                        order by MEASURE_TS desc
```

```
                        --ROWS BETWEEN 1 PRECEDING  AND 1 FOLLOWING
                        )
                        as latest_message_received_at

            ,NTH_VALUE( MEASURE_TS , 2 )
                FROM FIRST
                IGNORE NULLS
                over ( partition by group_id ,edge_node_id ,device_id ,DEVICE_NAME ,message_type
                        order by MEASURE_TS desc
                        --ROWS BETWEEN 1 PRECEDING  AND 1 FOLLOWING
                        ) as prev_latest_message_received_at

            ,timestampdiff('second' ,prev_latest_message_received_at ,latest_message_received_at) as
interval_time

            ,last_value(MEASURE_TS)
                ignore nulls
                over ( partition by group_id ,edge_node_id ,device_id ,message_type
                        order by MEASURE_TS desc
                        --ROWS BETWEEN 1 PRECEDING  AND 1 FOLLOWING
                        )
                        as oldest_message_received_at

            ,object_construct(
                'latest_message_received_at' ,latest_message_received_at
                ,'prev_latest_message_received_at' ,prev_latest_message_received_at
                ,'interval_seconds',interval_time
                ,'oldest_message_received_at',oldest_message_received_at
            ) as obj
        from rows_filtered

    ), unioned as (
        select group_id ,edge_node_id ,device_id ,DEVICE_NAME
            ,null as dbirth
            ,obj as ddata
        from object_constructed
        where message_type = 'DDATA'
        union
        select group_id ,edge_node_id ,device_id ,DEVICE_NAME
            ,obj as dbirth
            ,null as ddata
        from object_constructed
        where message_type = 'DBIRTH'

    )
    select group_id ,edge_node_id ,device_id ,DEVICE_NAME
        ,first_value(ddata)
                ignore nulls
                over (partition by group_id ,edge_node_id ,device_id ,DEVICE_NAME
                    order by ddata)
            as ddata
        ,first_value(dbirth)
                ignore nulls
                over (partition by group_id ,edge_node_id ,device_id ,DEVICE_NAME
                    order by dbirth)
            as dbirth
    from unioned
    ;
```

SQL Script 07

Setup roles specifically requiring help of privileged roles like SYSADMIN. These are:

- Create a custom role
- Assign the custom role to create task and execute task
- Create warehouse specifically used for ingestion
- Grants

SQL Script 07

```
-- =========================
-- In this script, we are setting up roles specifically requiring help of
-- privlilleged roles like SYSADMIN. These are:
--  - Create a custom role
--  - Assign the custom role to create task and execute task
--  - Create warehouse specifically used for ingestion
--  - Grants

-- REPLACE THE SESSION VARIABLE ACCORDING TO YOUR ENVIRONMENT
-- =========================

set processor_role = 'cl_bridge_process_rl';
set cl_bridge_ingestion_warehouse = 'cl_bridge_ingest_wh';
set staging_db = 'cl_bridge_stage_db';
set staging_db_schema = 'cl_bridge_stage_db.stage_db';

set node_db = 'cl_bridge_node_db';
set node_db_staging_schema = 'cl_bridge_node_db.stage_db';

-- >>>>>>>>>>>>>>>>>>>>>>>>>>> ROLE CREATION >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
    use role securityadmin;
    create role if not exists identifier($processor_role)
        comment = 'role used by cirruslink bridge to ingest and process mqtt/sparkplug data';

    grant role identifier($processor_role)
        to role sysadmin;

-- >>>>>>>>>>>>>>>>>>>>>>>>>>> WAREHOUSE >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
    use role sysadmin;

    create warehouse if not exists identifier($cl_bridge_ingestion_warehouse) with
        warehouse_type = standard
        warehouse_size = xsmall
        --max_cluster_count = 5
        initially_suspended = true
        comment = 'used by cirruslink bridge to ingest'
        ;


    grant usage on warehouse identifier($cl_bridge_ingestion_warehouse)
        to role identifier($processor_role);
    grant operate on warehouse identifier($cl_bridge_ingestion_warehouse)
        to role identifier($processor_role);

-- >>>>>>>>>>>>>>>>>>>>>>>>>>> GRANTS STAGE DB >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
    grant usage on database identifier($staging_db)
        to role identifier($processor_role);

    grant usage on schema identifier($staging_db_schema)
        to role identifier($processor_role);

    grant select on all tables in schema identifier($staging_db_schema)
        to role identifier($processor_role);

    grant insert on all tables in schema identifier($staging_db_schema)
        to role identifier($processor_role);

    grant select on future tables in schema identifier($staging_db_schema)
        to role identifier($processor_role);

    grant insert on future tables in schema identifier($staging_db_schema)
        to role identifier($processor_role);


-- >>>>>>>>>>>>>>>>>>>>>>>>>>> GRANTS NODE DB >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    grant usage on database identifier($node_db)
        to role identifier($processor_role);
```

```
    grant usage on schema identifier($node_db_staging_schema)
        to role identifier($processor_role);

    grant select on all tables in schema identifier($node_db_staging_schema)
        to role identifier($processor_role);

    grant insert on all tables in schema identifier($node_db_staging_schema)
        to role identifier($processor_role);

    grant select on all views in schema identifier($node_db_staging_schema)
        to role identifier($processor_role);

    grant usage on all functions in schema identifier($node_db_staging_schema)
        to role identifier($processor_role);

    grant usage on all procedures in schema identifier($node_db_staging_schema)
        to role identifier($processor_role);

    -- need for:
    --  - creating edge node specific tables and views
    grant create schema on database identifier($node_db)
        to role identifier($processor_role);
```

SQL Script 08

Setup roles specifically requiring help of privileged roles like SYSADMIN, SECURITYADMIN, ACCOUNTADMIN. These are:

- Create a custom role
- Assign the custom role to create task and execute task
- Grants

SQL Script 08

```
-- ========================
-- In this script, we are setting up roles specifically requiring help of
-- privlilleged roles like SYSADMIN, SECURITYADMIN, ACCOUNTADMIN. These are:
--  - Create a custom role
--  - Assign the custom role to create task and execute task
--  - Grants

-- REPLACE THE SESSION VARIABLE ACCORDING TO YOUR ENVIRONMENT
-- ========================

set reader_role = 'cl_bridge_reader_rl';

set reader_role_warehouse = 'compute_wh';

set staging_db = 'cl_bridge_stage_db';
set staging_db_schema = 'cl_bridge_stage_db.stage_db';

set node_db = 'cl_bridge_node_db';
set node_db_staging_schema = 'cl_bridge_node_db.stage_db';



-- >>>>>>>>>>>>>>>>>>>>>>>>>>>> ROLE CREATION >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
    use role securityadmin;

    create role if not exists identifier($reader_role)
        comment = 'role used by user/process to query and operate on the views and tables managed by cirruslink
bridge';

    grant role identifier($reader_role)
        to role sysadmin;

-- >>>>>>>>>>>>>>>>>>>>>>>>>>>> GRANTS WAREHOUSE >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
    use role accountadmin;
```

```
    grant usage on warehouse identifier($reader_role_warehouse)
        to role sysadmin with grant option;
    grant modify on warehouse identifier($reader_role_warehouse)
        to role sysadmin with grant option;
    grant operate on warehouse identifier($reader_role_warehouse)
        to role sysadmin with grant option;

    use role sysadmin;

    grant usage on warehouse identifier($reader_role_warehouse)
        to role identifier($reader_role);
    grant modify on warehouse identifier($reader_role_warehouse)
        to role identifier($reader_role);
    grant operate on warehouse identifier($reader_role_warehouse)
        to role identifier($reader_role);

-- >>>>>>>>>>>>>>>>>>>>>>>>>>> GRANTS TASK EXECUTION >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    -- Creating and executing tasks require exculated privileges that can be done
    -- only by the accountadmin. Hence we have to switch roles
    -- use role accountadmin;
    -- grant execute managed task on account to role identifier($reader_role);
    -- grant execute task on account to role identifier($reader_role);


-- >>>>>>>>>>>>>>>>>>>>>>>>>>> GRANTS STAGE DB >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
    grant usage on database identifier($staging_db)
        to role identifier($reader_role);

    grant usage on schema identifier($staging_db_schema)
        to role identifier($reader_role);

    grant select on all tables in schema identifier($staging_db_schema)
        to role identifier($reader_role);
    grant select on future tables in schema identifier($staging_db_schema)
        to role identifier($reader_role);

-- >>>>>>>>>>>>>>>>>>>>>>>>>>> GRANTS NODE DB >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    grant usage on database identifier($node_db)
        to role identifier($reader_role);

    grant usage on schema identifier($node_db_staging_schema)
        to role identifier($reader_role);

    grant select on all tables in schema identifier($node_db_staging_schema)
        to role identifier($reader_role);

    grant select on all views in schema identifier($node_db_staging_schema)
        to role identifier($reader_role);

    grant usage on all functions in schema identifier($node_db_staging_schema)
        to role identifier($reader_role);

    grant usage on all procedures in schema identifier($node_db_staging_schema)
        to role identifier($reader_role);

    -- need for:
    --  - new generated specific tables and views
    use role securityadmin;
    grant usage on future schemas in database identifier($node_db)
        to role identifier($reader_role);
```