Meshlet Configurations for Snowflake

To enhance your DataSync integration for **Snowflake**, you can configure the Snowflake Bulk Load Meshlet to the configurations below.

⚠️ **WARNING:**

- If you will be setting up the DataSync Agent to also connect to the same database, you will want to configure the Agent to use the same column cases. By default, the Snowflake Bulk Load Meshlet uses **lowercase** for the column names. Also, the **QUOTED_IDENTIFIERS_IGNORE_CASE** setting in Snowflake should be kept with its default value of **false**.
- The meshlet uses the **TIMEZONE** session parameter to connect to Snowflake in order to save all timestamps in UTC . By default, **glide_date_time** field is mapped to the **TIMESTAMP_LTZ(9)** column type. This allows you to query using a local timezone as needed.

What's on this page?

To check out the general meshlet configurations, see General Meshlet Configurations for DataSync.

List of Meshlet Configurations for Snowflake

Directive	Default Value	Description

maxFileSize		**Required Configuration.** This configuration specifies the max size for temporary files created as the meshlet pushes records to Snowflake. **10000** will be used if input is over **10000** to prevent possible performance and memory issues. A suggested value is **5000**.
		`perspectium:`
		`  filesubscriber:`
		`  maxFileSize: 5000`

maxStringSize	500	Specifies the maximum string size for the entire collection. The maximum value you can use is **1000**, if input is greater than that value, the input will default to **1000**.
		`perspectium:`
		`  filesubscriber:`
		`  maxStringSize: 500`

customFileName	$table-$randomid	Names file with format of table - random id. File names MUST be unique.
		`perspectium:`
		`  filesubscriber:`
		`  customFileName: $table-$randomid`

fileDirectory	/files	Directory where the locally created files get made. (In respects to where application is running)
		`perspectium:`
		`  filesubscriber:`
		`  fileDirectory: /files`

postInterval	2	Minutes to check dead periods. Check every x minutes to compare if the in memory collection is the same as the last x minutes. If so, write records to file and push to Snowflake
		
		`perspectium:`
		`  snowflake:`
		`  postInterval: 2`

deleteFiles	true	Indicates whether you want to keep or delete locally created CSV files. Will not have a noticeable performance hit.
		
		`perspectium:`
		`  filesubscriber:`
		`  deleteFiles: true`

fileCleanerInterval	1	How often (in minutes) the file cleaner job will run to clean up local files created by the meshlet. This job is similar to the ServiceNow Data Cleaner Scheduled Job. For example, value of **4** will run the job every four minutes. The default value is **1.**
		
		`perspectium:`
		`  filesubscriber:`
		`  fileCleanerInterval: 4`

fileMaxAge	1	Any csv file in the filesDirectory older than fileMaxAge in minutes, will be automatically deleted. The default value is **1.**
		
		`perspectium:`
		`  filesubscriber:`
		`  fileMaxAge: 1`

legacyDbViewId	false	Derivative to use the legacy version for database view tables of concatenating GUID values into a sys_id field. If false, meshlet will use the pre-constructed encoded sys_id created by ServiceNow.
		
		`perspectium:`
		`  filesubscriber:`
		`  legacyDbViewId: false`
		
		NOTE: If using the Share only selected fields option on your bulk share and you don't want the encoded sys_id to be used, you will want to set all three of the legacyDb properties to be true:
		
		`perspectium:`
		`  filesubscriber:`
		`  legacyDbViewId: true`
		`  legacyDbViewAlphaOrder: true`
		`  legacyDbViewSkipNull: true`

legacyDbViewAlphaOrder	false	**Supported in Helium 6.1.7 Release and Newer**
		
		By default, the Snowflake meshlet will concatenate GUID values for database view tables into a sys_id field in the order it reads other *_sys_id fields (such as the **fc_sys_id** and **jc_sys_id** fields in the **flow_context_chunk** database view table) which is not guaranteed to be in alphabetical order.
		
		With this configuration set to **true**, the *_sys_id fields will be read and concatenated in alphabetical order i.e. **fc_sys_id + jc_sys_id** for **flow_context_chunk.**
		
		This feature requires **legacyDbViewId** be set to **true.** If **legacyDbViewId** is not configured to **true**, this configuration will not be enabled.
		
		`perspectium:`
		`  filesubscriber:`
		`  legacyDbViewAlphaOrder: false`
		
		If this configuration is not specified, the meshlet will concatenate GUID values in the order it reads other *_sys_id fields.

legacyDbViewSkipNull	false	**Supported in Helium 6.1.7 Release and Newer**
		
		By default, the Snowflake meshlet will concatenate all GUID values for database view tables into a sys_id field including GUID values that are **null** (i.e. have no value), matching how ServiceNow appends sys_id values. For example if the **fc_sys_id** field has a value of **12345** and **jc_sys_id** has a value of **null,** the sys_id field will be **12345null** (in ServiceNow this would show as __ENC__ABCEDEER=-NULL as ServiceNow does encoding on *_sys_id values).
		
		With this configuration set to **true**, *_sys_id fields with values of **null** will be skipped and not added to the concatenated value. In the above example, the sys_id field would be **12345**.
		
		This feature requires **legacyDbViewId** be set to **true.** If **legacyDbViewId** is not configured to **true**, this configuration will not be enabled.
		
		`perspectium:`
		`  filesubscriber:`
		`  legacyDbViewSkipNull: false`
		
		If this configuration is not specified, the meshlet will concatenate all GUID values including **null** values.

useReceipts	true	Enable data guarantee.
		
		`perspectium:`
		`  message:`
		`  receipts:`
		`  enable: true`

receiptOutboundQueue		Queue receipts are returned to, typically default subscribed to on a ServiceNow instance.
		
		`perspectium:`
		`  message:`
		`  receiptOutboundQueue: psp.out.servicenow.dev1234`

useConnectionProperties	false	When enabled, the user, password, and role properties on the yml are no longer being used to create the database connection.
		
		`perspectium:`
		`  snowflake:`
		`  useConnectionProperties: true`
		
		This will allow you to add these or other parameters to your connectionUrl to leverage other forms of authentication. For example:
		
		`connectionUrl: jdbc:snowflake://hga12424.us-east-1.snowflakecomputing.com/?USER=xxxx&PASSWORD=xxxxx&ROLE=xxxx`

updateTimeOnInsert	true	In conjunction with Add custom fields in Snowflake, if you want the IO update column to be NULL on record inserts, set the value to **false**.
		
		`perspectium:`
		`  snowflake:`
		`  updateTimeOnInsert: true`
		
		By default, updateTimeOnInsert is true, and IO update columns are populated on record inserts.

useLocalSchema	false	Enabling this directive will force the snowflake meshlet to read schemas via files exclusively from the local file storage and disable all calls to ServiceNow. Any new tables shared will require table schemas to be added to the local file storage. See Load Schema Files on how to get started.
		
		`perspectium:`
		`  auth:`
		`  useLocalSchema: true`

loadSchemasOnStart	true	**Supported in Helium 6.1.11 Release and Newer**
		
		In conjunction with the **useLocalSchema** configuration above, this configuration controls whether the schemas stored in the local file storage are loaded when the meshlet is started to immediately create the tables in the Snowflake database.
		
		By default, this value is **true** meaning that when the meshlet starts up, it will load all the schemas stored in local file storage and then use them to create the tables in Snowflake when the meshlet is starting up. This will ensure faster performance for when records are received by the meshlet to insert/update into the database table since it doesn't have to create tables at that time.
		
		Set this value to **false** if you want the meshlet to create the table only when it receives a record for the table.
		
		`perspectium:`
		`  auth:`
		`  loadSchemasOnStart: true`

softDelete	false	This feature allows you to add columns to a database table that gets updated when the meshlet receives a message indicating a record is to be deleted. The meshlet will update said columns appropriately and will not delete the record in Snowflake. For more information, see Soft Deletes.
		
		`perspectium:`
		`  snowflake:`
		`  softDelete: true`

quotedColumns	false	Enabling this directive will add quotations around column names in the database.
		
		`perspectium:`
		`  snowflake:`
		`  quotedColumns: true`

translateBooleans	false	To translate boolean values such as **true** and **false** to **1** (true) and **0** (false). For the case where you've modified the databaseConfig.json so that boolean fields are created as NUMBER fields in Snowflake, this configuration will translate values in boolean fields of the incoming ServiceNow records from **true/false** to **1/0** so they can be saved into Snowflake.
		
		By default, this configuration is **false** and boolean values are **not translated**. To enable this configuration, set it **true**:
		
		`perspectium:`
		`  mapping:`
		`  translateBooleans: true`

truncateColumns	true	**Supported in Helium 6.1.4 Release and Newer**
		
		By default, the Snowflake meshlet will create fields without specifying a length (the one exception is timestamps) so that these fields are created with the maximum size i.e. VARCHAR(16777216) to avoid issues with the content being too long for the column.
		
		If columns are created with a size where content being inserted is longer than the column's maximum length, by default the meshlet will truncate the text to fit into the column using Snowflake's TRUNCATECOLUMNS option. To change it so that the meshlet does not truncate the column (and instead will throw an error), change the configuration to **false**:
		
		`perspectium:`
		`  snowflake:`
		`  truncateColumns: false`
		
		If this configuration is not specified, the meshlet will treat this configuration as true and truncate content to fit into a column's length.

extendColumnsMaxLength	true	**Supported in Helium 6.1.5 Release and Newer**
		
		With this configuration, if the meshlet sees content that is greater than the current column's length, it will extend the column to the max size supported for Snowflake i.e. VARCHAR(16777216) for string (VARCHAR) fields.
		
		Using this configuration along with the above **truncateColumns** will allow columns to be extended to their maximum size and then only truncate when the content is longer than the maximum size for the column so that there is no loss of records.
		
		For example, with both the **extendColumnsMaxLength** and **truncateColumns** configurations set to true, you will see:
		
		1) The Snowflake meshlet receives a record of 5,000 characters for a table column that is defined as VARCHAR(4000)
		
		2) The column is extended to VARCHAR(16777216)
		
		3) The record is saved successfully
		
		4) A record with 18,000,000 characters is received and is truncated to 16,777,216 characters to fit into this column and save the record into Snowflake
		
		If you prefer that columns not be extended, change the configuration to **false**. Note that by default all columns are created with their lengths as maximum size as mentioned in the **truncateColumns** configuration. So this is only applicable if you've configured it or created tables with smaller sizes.
		
		`perspectium:`
		`  snowflake:`
		`  extendColumnsMaxLength: false`

Go to top of page

Debugging Logs

To enable debugging logs for the Snowflake Meshlet, you can configure your connection url in the **application.yml** with one the following:

```
jdbc:snowflake://xxxxxx.snowflakecomputing.com/?TRACING=CONFIG
```

```
jdbc:snowflake://xxxxxx.snowflakecomputing.com/?TRACING=FINEST
```

```
jdbc:snowflake://xxxxxx.snowflakecomputing.com/?TRACING=ALL
```

Where **ALL** will provide the most detailed level of additional logging. See tracing=<string> for more information on the different logging levels available.

A tmp log file will be created. See How to generate log file on Snowflake connectors.

Go to top of page