Make a copy of the document if any changes are required for customer deliverables.

Please DO NOT edit this document.

Copy Instructions:

1. Click **File** -> **Make a Copy.**
2. Replace ‘Copy of ‘ from the **Name** text field with ‘<CUSTOMER> - ‘.
3. Replace ‘<SOLUTION_NAME> ’ with the Provider’s solution.
4. Select the desired folder to save a copy.
5. Select the **Copy comments and suggestions** box. This will also select the **Share it with the same people** box above it.

<CUSTOMER>
<SOLUTION_NAME>
Detailed Design

Version 1
<AUTHOR>
<DATE>

Revision Summary

Date	Revision History	Comments
<REV_DATE>	1.0	Initial Version

snowflake

Table of Contents

Prerequisites

- The provider must accept the Snowflake Marketplace Provider Terms of Service.
- The consumer must accept the Snowflake Marketplace Client Terms of Service.
- The provider must determine which cloud regions the app will be available.
- The provider must create an account that serves as the main account for the ACF.
- The provider must create an account in each cloud region their native app will be available in. This account is used to collect events from consumer apps in each region. Events from this account are routed to the native app/ACF account via private data listings.
    - Each account must be enrolled in the Listing API Private Preview
    - Once this account is created, the provider will set it as the Event account for the cloud region, by executing the following in the Snowflake Organization account, as `ORGADMIN`
        - `CALL SYSTEM$SET_EVENT_SHARING_ACCOUNT_FOR_REGION('<REGION>', 'PUBLIC', '<ACCOUNT_NAME>');`
            - <REGION> can be found by executing `SELECT CURRENT_REGION();`
            - <ACCOUNT_NAME> can be found by executing `SELECT CURRENT_ACCOUNT_NAME();`
        - `CALL SYSTEM$SYSTEM$ENABLE_GLOBAL_DATA_SHARING_FOR_ACCOUNT('<ACCOUNT_NAME>');`
            - <REGION> can be found by executing `SELECT CURRENT_REGION();`
            - <ACCOUNT_NAME> can be found by executing `SELECT CURRENT_ACCOUNT_NAME();`
- The user that will execute the scripts in each account must have either the `ACCOUNTADMIN` role granted or a role that can create roles and manage grants on the account to other roles.

Provider vs. Consumer

The Native Apps Framework term “provider” refers to <CUSTOMER>, as the app’s owner. The Native Apps Framework term “consumer” refers to <CUSTOMER> clients that install the app.

Disclaimer

Any screenshots included in this guide are examples. Please refer to the text in the steps below when installing this app.

Snowflake Native Apps Overview

Snowflake Native Apps lets providers build data apps that leverage Snowflake functionality. Providers can share data content and related app logic with consumers. After creating a Native App, it can be deployed by installing it in a consumer account.

Some of the functionality and advantages of a Native App include:

- The ability to share data content and include stored procedures, user-defined functions, and external functions within a Native App.
- The ability to create versions and patches for a Native App.
- A streamlined testing environment that supports the ability to create and install a Native App from a single account.
- A robust developer workflow. While data and related database objects remain within Snowflake, supporting app files and resources can be managed locally and used with a preferred source control system.

For more information, view the Native Apps Developer Guide: https://docs.snowflake.com/en/developer-guide/native-apps/tutorials/getting-started-tutorial#id6.

Application Control Framework

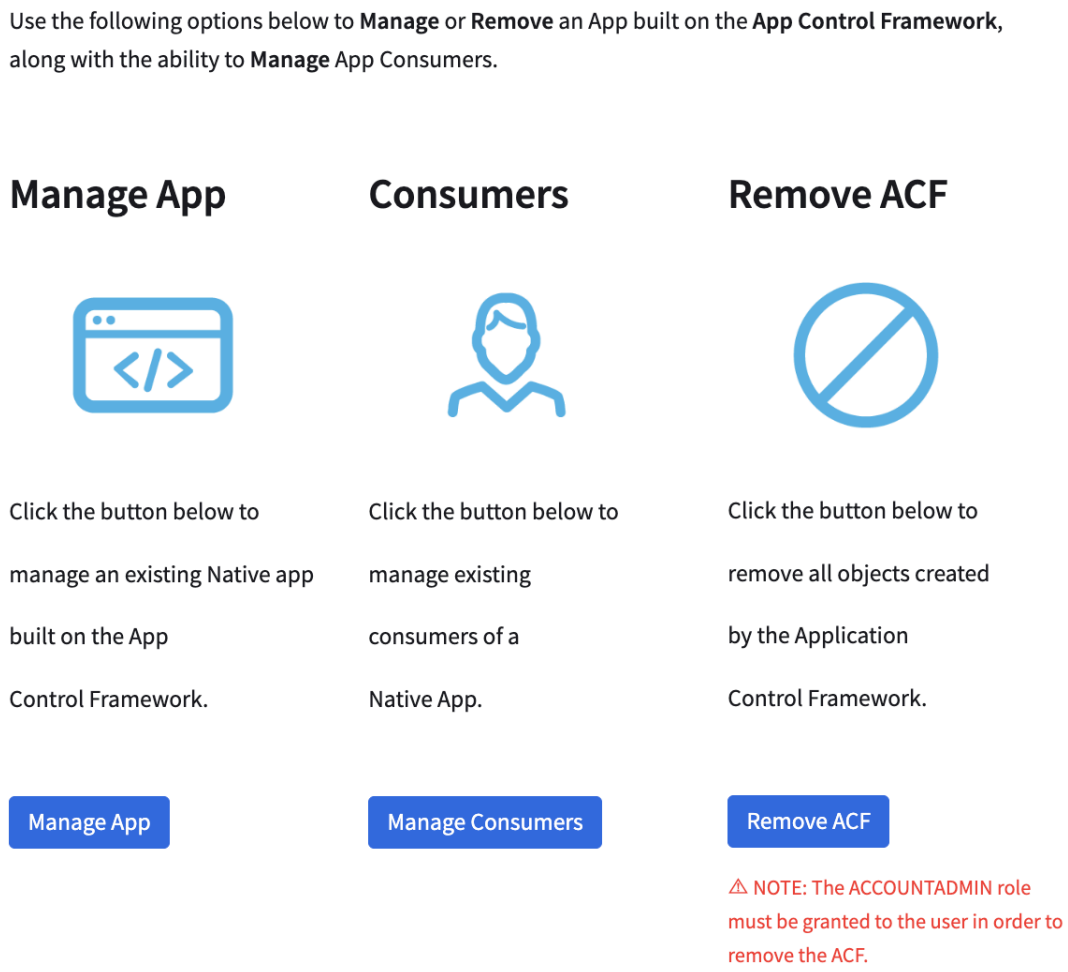
Snowflake's Application Control Framework (ACF) is built using Snowflake Native Apps, and allows a provider to integrate their existing app logic (already on Snowflake), with minimal/no modification, into a Snowflake Native App.

The ACF has pre-built controls which allows the provider to monitor and control app usage, for each consumer. The provider can also create their own custom controls and rules that manage access to the app.

In addition, the provider can control which stored procedure(s) the consumer can access. The stored procedure(s) will remain hidden from the consumer, but accessible by the app.

App Control Manager

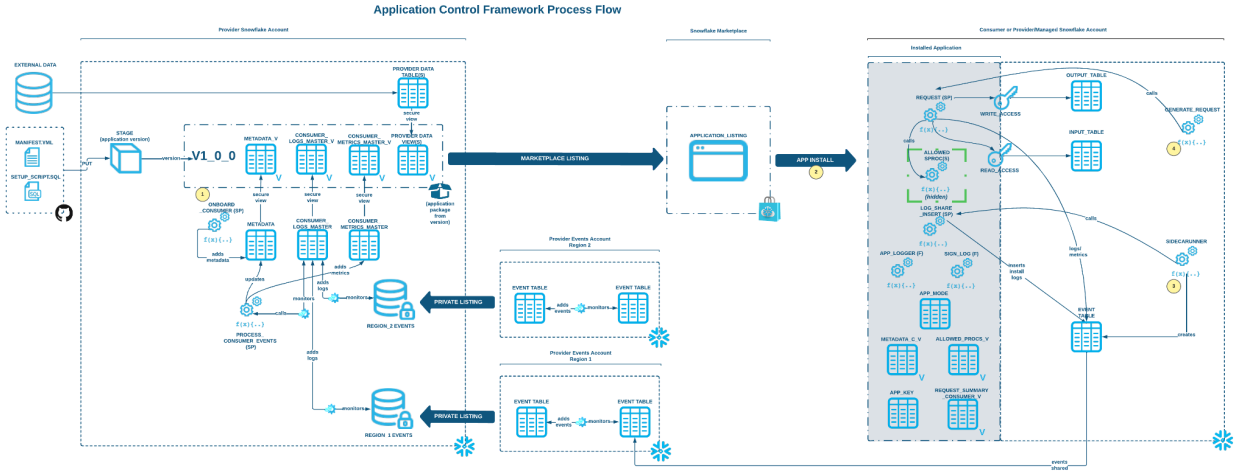
Once the Application Control Framework scripts have been executed, the framework, along with a Streamlit app, called the App Control Manager, are available in the provider's account. The App Control Manager allows the provider to easily build and manage an app built on the ACF, manage consumers, and remove the ACF if/when desired.



Detailed Design

The Application Control Framework consists of a set of scripts that the provider and the consumer deploy in their respective Snowflake accounts that will create the necessary app objects. Once these scripts are deployed, the provider can onboard/enable consumers and the consumer can use the Native App. Refer to the Deployment Guide and Consumer Onboarding Guide documents for more details.

Design Diagram



Key Framework Features

The following section describes key features of the Application Control Framework.

Multiple Native App Modes

The framework comes with the ability to build a native app with custom functionality, depending on the version of the app. For example, the consumer can evaluate the “free” version of the app from the Snowflake Marketplace, without interaction from the provider. Once the consumer is interested in the one or more paid versions of the app, they can be granted access to the desired version.

By default, the ACF supports three app modes:

- **FREE**: a free version of the app that is publicly available in the Snowflake Marketplace. This version offers limited functionality, meant to entice the consumer to convert to a paid version of the app. Each consumer of this app version has the same entitlements/limits (i.e. five requests).

- **PAID**: a paid version of the app that is publicly available in the Snowflake Marketplace. This version offers more or complete app functionality. Each consumer of this app has the same entitlements/limits (if any) enforced (i.e. process 1MM records every 30 days).
- **ENTERPRISE**: a version of the app where unique entitlements/limits can be set for each consumer. The entitlements/limits are managed via the ACF's App Control Manager. This is ideal for providers that want to create custom deals with consumers where the default entitlements/limits of the other app versions are not ideal for the consumer. Enterprise versions of the app should be listed privately and only made available to a single consumer.

Pre-built and Custom Controls

The framework comes with pre-built controls that manage each consumer's access to the app, records they can process within a given period, etc. In addition, if the app provides multiple functionality, providers can specify which functionality each consumer has access to. The provider can also add custom controls that are unique to their app.

Access via Custom Rules

The framework allows the provider to create custom rules, based on either pre-built and/or custom controls. This feature enables the provider to have even more control over access to the app, beyond the out-of-the-box protections.

App Key Generation

Each app installation generates a unique app key. This app key is used to tie requests, logs, and metrics to a particular install; which may be useful when troubleshooting any potential issues. In addition, the app key is also used to determine whether to process incoming logs from the consumer's share, in the event access to the log share was lost, then restored. See Log Share Monitoring for more details.

Event Collection

The consumer must create an events table and allow the events to be shared with the provider. The two types of events native apps built via the ACF collect are **logs** and **metrics**.

Once granted, the app's APP_LOGGER stored procedure adds events to this table. Events are collected using a VARIANT field, which accepts any valid JSON payload. This allows the provider to customize the payload to collect any data deemed necessary.

NOTE: Native apps built using the ACF **require** the consumer to enable event sharing to the provider.

Consumption Tracking

Comments have been added to key framework objects to track Snowflake credit consumption of apps built using the ACF. **Please do not modify comments**. Any modifications could result in the inability to properly track consumption.

Pre-built Controls

The Application Control Framework comes with pre-built controls out of the box. These controls are maintained in the METADATA table (described later in this document). Custom controls can always be added, as needed.

Control	Description
app_code	The abbreviated name of the app. This code is used to tie all objects created for managing this app together.
allowed_procs	The comma-separated list of app stored procedures the provider allows the consumer to access (ideal for providers that offer multiple types of functionality in their app)
allowed_funcs	The comma-separated list of app functions the provider allows the consumer to access (ideal for providers that offer multiple types of functionality in their app)
record_cost	The per-record cost for using this solution. This is used when custom_billing is set to 'Y'
custom_billing	Y/N flag that determines whether custom billing is enabled for the consumer.
limit	The number of either records or requests the consumer is allowed
limit_type	The type of limit (either `RECORD_LIMIT` or `REQUESTS_LIMIT`)
limit_interval	The interval at which the limit is enforced (i.e. 1 day)
limit_enforced	Flag indicating whether to enforce request limits for the consumer (ideal for providers that create test consumer accounts and do not want the limit to interfere with testing)

custom_rules	Flag indicating which custom rules (if any) should be enforced when making a request. **NOTE**: Currently only one custom rule can be enforced per consumer
managed	Flag indicating whether the consumer is 'managed' via a provider proxy Snowflake account (for those consumers that are not on Snowflake)
auto_enable	Flag indicating whether to auto-enable the consumer once their log share is available to the provider **NOTE**: this is Y by default to facilitate auto-enabling the consumer. In the event the consumer drops the log share, this is set to N and not switched back until the provider explicitly does so. This is to prevent the consumer from constantly dropping/re-adding the provider to the share.
comments	Field to store applicable comments, i.e. why the consumer has been disabled
enabled	Flag indicating whether the consumer has been enabled to use the app
app_key	Unique key assigned to each install the consumer makes
install_count	Number of times the app has been installed by the consumer
first_install_timestamp	Timestamp of first install by the consumer
last_install_timestamp	Timestamp of last install by the consumer
input_records	The number of input records the consumer has submitted
input_records_this_interval	The number of input records submitted during the allotted period. This gets reset to 0 at the allotted period
total_requests	The number of requests made by the consumer
requests_processed_this_interval	The number of consumer requests processed during the allotted period. This gets reset to 0 at the allotted period (i.e. daily)
last_request_timestamp	Timestamp of last request made by the consumer (SYSDATE())
total_records_processed	The number of consumer records processed since installation
records_processed_this_interval	The number of consumer records processed during the allotted period. This gets reset to 0 at the allotted period.
total_matches	The total number of matched records (if applicable) the consumer has received since installing the app

matches_this_interval	The number of matched records (if applicable) the consumer has received during the allotted period. This gets reset to 0 at the allotted period
limit_reset_timestamp	Timestamp of when the counters will be reset

ACF Deployment

See the **Application Control Framework - Deployment Guide** document for details on configuring the events account(s) and deploying the ACF.

Application Setup and Listing

See the **Application Control Framework - Native App Deployment Guide** document for details on app setup and listing.

Consumer Onboarding

See the **Application Control Framework - Native App Deployment Guide** document for details on the consumer onboarding process.

Design Limitations

Event Latency

- There is a latency between when events are generated and when they arrive in the provider's ACF account. This framework is designed to provide as little delay as possible in getting the events to the ACF account.
- When the consumer initially installs a non-free version of the native app, the consumer will experience a delay of a few minutes in being able to use the native app.

Design Considerations

Match Definition

- The ACF has predefined controls called **total_matches** and **matches_this_interval**, but are not defined. The provider must define these, if applicable. If required, there will need to be slight modifications to the ACF to define and track matches. Please consult Snowflake for more information.

Objects Created

Events Account(s)

The following objects are created when the provider executes the scripts in the `event_acct` directory of the ACF code repository. These objects are created in each event account (one per region) where the scripts are executed See the Deployment Guide for more details.

Database: EVENTS

Description
The database that stores the consumer events from the specified region.

Schema: EVENTS

Description:
This schema stores the consumer events from the specified region

Table: EVENTS

Description
Table containing the consumer events from the specified region.

Definition
See https://docs.snowflake.com/en/developer-guide/logging-tracing/event-table-columns for event table definition details.

Stored Procedure: STREAM_EVENTS

Description:
For each app code, this stored procedure creates a stream and task to stream events from the account's event table to the app's table shared to the ACF account.

Parameters:
- **app_codes** (ARRAY) - Array of app codes.

Stored Procedure: REMOVE_APP_EVENTS

Description:
This stored procedure removes all event-related objects for each app code submitted.

Parameters:
- **app_codes** (ARRAY) - Array of app codes.

Stored Procedure: REMOVE_ALL_EVENTS

Description:
This stored procedure removes all event-related objects from the account.

Parameters:
- N/A

Database: <APP_CODE>_EVENTS_FROM_<CURRENT_REGION

Description
Database that contains events streamed from the account's event table. A separate database, schema, and table are created because event tables cannot be shared.

Schema: EVENTS

Description:
This schema stores the consumer events streamed from this account's event table.

Table: EVENTS

Description
Table containing the consumer events streamed from this account's event table.

Definition
See https://docs.snowflake.com/en/developer-guide/logging-tracing/event-table-columns for event table definition details.

Stream: EVENTS_STREAM

Description
Stream created to stream events from the account's event table to the table shared to the ACF account.

- **Append Only**: True
- **Data Retention Time in Days**: 1

Task: EVENTS_TASK_i

Description
Task that adds events from the EVENTS_STREAM stream to the table shared to the ACF account.

- **Warehouse**: N/A (serverless)

- **Schedule**: 1 minute
- **Action**: Inserts events into the <APP_CODE>_EVENTS_TO_<ACF_ACCOUNT_LOCATOR>.EVENTS.EVENTS table.

NOTE: *i* = the number of tasks created. There are 20 tasks (i = 1 - 20), started 3 seconds apart. This results in one of the tasks checking the EVENTS_STREAM stream every 3 seconds. This can be altered as needed.

Share: <APP_CODE>_EVENTS_FROM_<CURRENT_REGION>_SHARE

Description
Share created to share streamed events from this account to the main ACF account. A separate database, schema, and table are created because event tables cannot be shared.

Contents:
Event Table: <APP_CODE>_EVENTS_FROM_<CURRENT_REGION>.EVENTS.EVENTS

Listing: <APP_CODE>_EVENTS_FROM_<CURRENT_REGION>

Description
Listing created to privately share streamed events from this account to the main ACF account.

Contents:
Share: <APP_CODE>_EVENTS_FROM_<CURRENT_REGION>_SHARE

Main (ACF) Account

The following objects are created when the provider executes the scripts in the `main_acct` directory of the ACF code repository. See the Deployment Guide for more details.

Role: P_<APP_CODE>_ACF_ADMIN
The Application Control Framework utilizes an "administrative" role that creates/manages the app objects.

Warehouse: P_<APP_CODE>_ACF_WH

Description
The warehouse used to create the framework objects.

Database: <APP_CODE>_EVENTS_FROM_<REGION>

Description
Database created from the private listing for events from each region the app is available in. The events from each region are streamed to the EVENTS_MASTER table, then processed for consumer installs, requests, interval resets, etc.

Schema: EVENTS

Description:
This schema stores the consumer events from the specified region

Table: EVENTS

Description
Table containing the consumer events from the specified region.

Definition
See https://docs.snowflake.com/en/developer-guide/logging-tracing/event-table-columns for event table definition details.

Database: P_<APP_CODE>_ACF_DB

Description
Database that contains the app control framework objects.

Schema: EVENTS

Description
Schema containing the events table.

Table: EVENTS_MASTER

Description
Table containing consumer event messages.

Definition

Column	Data Type	Description	Null?
MSG	VARIANT	Log message from relevant app events.	N

Table: CONTROL_EVENTS

Description
Table containing messages for various ACF events (i.e. onboarding/updating consumers).

Definition

Column	Data Type	Description	Null?
MSG	VARIANT	Log message from relevant ACF events.	N

Stream: <EVENTS_DB>_EVENTS_STREAM

Description
Stream created to stream events from the region to the EVENT_MASTER table.

- **Append Only**: True
- **Data Retention Time in Days**: 1

Task: <EVENTS_DB>_EVENTS_TASK_i

Description
Task that processes events from the <EVENTS_DB>_EVENTS_STREAM stream.

- **Warehouse**: N/A (serverless)
- **Schedule**: 1 minute
- **Action**: Inserts events into the EVENTS_MASTER table.

NOTE: *i* = the number of tasks created. There are 2 tasks (i = 1 - 2), started 30 seconds apart. This results in one of the tasks checking the <EVENTS_DB>_EVENTS_STREAM every 30 seconds. This can be altered as needed.

Stream: EVENTS_MASTER_STREAM

Description
Stream created to process new events in the EVENT_MASTER table.

- **Append Only**: True
- **Data Retention Time in Days**: 1

Task: PROCESS_CONSUMER_EVENTS_TASK_i

Description
Task that processes events from the EVENTS_MASTER_STREAM stream.

- **Warehouse**: N/A (serverless)
- **Schedule**: 1 minute
- **Action**: Calls the PROCESS_CONSUMER_EVENTS stored procedure.

NOTE: *i* = the number of tasks created. There are 2 tasks (i = 1 - 2), started 30 seconds apart. This results in one of the tasks checking the EVENTS_MASTER_STREAM every 30 seconds. This can be altered as needed.

Stored Procedure: STREAM_TO_EVENT_MASTER

Description:
For each event account database, this stored procedure creates a stream and task to stream events from the event account's event table to the EVENTS_MASTER_TABLE.

Stored Procedure: PROCESS_CONSUMER_EVENTS

Description:
Once new events are added to the EVENTS_MASTER table, this stored procedure is called to check for consumer installs, requests, when to reset their interval counts, etc.

Parameters:
- N/A

Schema: METRICS

Description
Schema containing metrics table and metrics summary view.

View: REQUEST_SUMMARY_MASTER_V

Description

View presenting all metrics entries from the METRICS table, in tabular format.

Definition

Column	Data Type	Description	Null?
ACCOUNT	VARCHAR	The consumer's Snowflake Account Locator	N
CONSUMER_NAME	VARCHAR	The consumer's company name	N
ENTRY_TYPE	VARCHAR	The type of event (i.e. log or metric)	N
REQUEST_ID	VARCHAR	Request ID	N
PROC_NAME	VARCHAR	The allowed procedure called	N
PROC_PARAMETERS	VARCHAR	The allowed procedure's parameters	Y
INPUT_TABLE_NAME	VARCHAR	Input table name	N
INPUT_RECORD_COUNT	NUMBER(38,0)	Input table record count	Y
RESULTS_TABLE_NAME	VARCHAR	Results table name	Y
RESULTS_RECORD_COUNT	NUMBER(38,0)	Results table record count	N
RESULTS_RECORD_COUNT_DISTINCT	NUMBER(38,0)	Results table record count(distinct)	N
STATUS	VARCHAR	Request status	N
COMMENTS	VARCHAR	Comments	Y
SUBMITTED_TS	TIMESTAMP_LTZ(9)	Submitted timestamp	N
COMPLETED_TS	TIMESTAMP_LTZ(9)	Completed timestamp	Y

Schema: METADATA

Description

Schema containing metadata-related tables.

Table: METADATA_DICTIONARY

Description

Table containing the definitions, key attributes (such as default value), for each pre-built and custom control (metadata key).

Definition

Column	Data Type	Description	Null?
CONTROL_NAME	VARCHAR	The name of the control (metadata key)	N
CONTROL_TYPE	VARCHAR	One of the four types of control: • preventive • detective • deterrent • corrective	N
CONDITION	VARCHAR	The relationship between the control and its default value (i.e. = or <=)	N
DEFAULT_VALUE	VARCHAR	The control's default value	Y
CONSUMER_CONTROL	BOOLEAN	Flag indicating whether the control is set for each consumer	N
SET_VIA_ONBOARD	BOOLEAN	Flag indicating whether the control can be modifiable during the consumer onboarding process. Custom and certain pre-built controls are allowed to be overwritten.	N
CONSUMER_VISIBLE	BOOLEAN	Flag indicating whether or not the control is visible by the consumer via the app's METADATA_C_V view. These are controls the provider would like to store/track, but hide from the consumer	N
DESCRIPTION	VARCHAR	The control's description	N

Table: RULES_DICTIONARY

Description

Table containing all custom rules, in JSON format, that the provider creates to further control access to the app.

Definition

Column	Data Type	Description	Null?
RULE_NAME	VARCHAR	The name of the rule	N

RULE_TYPE	VARCHAR	The type of rule (currently only CUSTOM)	N
RULE	VARCHAR	The JSON string containing the rule's conditions	N
METADATA_USED	VARCHAR	A comma-separated list of metadata keys used by the rule.	N
DESCRIPTION	VARCHAR	The rule's description	N

Table: METADATA

Description
Table containing all metadata, in key/pair format.

Definition

Column	Data Type	Description	Null?
ACCOUNT_LOCATOR	VARCHAR	The consumer's Snowflake Account Locator	N
CONSUMER_NAME	VARCHAR	The consumer's company name	N
KEY	VARCHAR	The metadata key name (i.e. 'enabled')	N
VALUE	TIMESTAMP_LTZ(9)	The metadata value (i.e. 'Y')	N

Schema: CONSUMER_MGMT

Description
Schema created to store the consumer-related stored procedures.

Stored Procedure: ONBOARD_CONSUMER

Description:
This stored procedure adds consumer values, including record limit and interval are added to the METADATA table. This procedure is executed once per consumer to be onboarded.

NOTE:

- Consumers of the **FREE** and **PAID** app versions get onboarded when they share events with the provider.
- Consumers of the **ENTERPRISE** app version get onboarded prior to making the private listing available to the consumer.

Parameters:

- **account_locator** (VARCHAR) - The consumer's Snowflake Account Locator

- **consumer_name** (VARCHAR) - The consumer's company name
- **controls** (VARCHAR) - JSON payload, passed as a VARCHAR, that contains the control values to set. Any values passed will overwrite defaults. If no controls are passed, then default values, as defined in the `METADATA_DICTIONARY` table, will be used.

Stored Procedure: REMOVE_CONSUMER

Description:
This procedure removes a consumer's app-related objects and metadata values from the METADATA table. Consumer logs are kept, for record keeping purposes.

Parameters:
- **account_locator** (VARCHAR) - The consumer's Snowflake Account Locator
- **consumer_name** (VARCHAR) - The consumer's company name

Schema: UTIL

Description
Schema containing various stored procedures utilized to create and manage the app.

Stored Procedure: GRANTS_TO_DATA_OWNER

Description:
This stored procedure grants each data owner role the appropriate privileges needed for the data owner role to grant source data that it owns to the app package. This stored procedure is to be used in the event the **P_<APP_CODE>_ACF_ADMIN** role does not own the source data referenced in the app package. This stored procedure is executed once for each app package.

Once executed, the data owner role then executes the **APP_PKG_SOURCE_VIEWS** stored procedure to grant access to the source tables/views.

NOTE: the list of objects must be of the same type (i.e. `TABLE` or `FUNCTION`)

Parameters:
- **pkg_list** (ARRAY) - An array of app package(s).
- **role** (VARCHAR) - The data owner role

Stored Procedure: APP_PKG_SOURCE_VIEWS

Description:
This stored procedure is used by the source data owner to grant the app package(s) privileges to the source data and creates a view from each table/views in the source table list in the specified app package(s). In addition, this stored procedure can also revoke access to any

tables/views from the app package(s). This stored procedure is executed once for each app package.

Parameters:

- **table_list** (ARRAY) - An array of source tables/views.
- **pkg_list** (ARRAY) - An array of app package(s)
- **action** (VARCHAR) - The action to perform. The only accepted options are GRANT or REVOKE.

Stored Procedure: REMOVE_APP

Description:
This procedure removes the ACF and all app packages from their Snowflake Account.

Parameters:

- N/A

Schema: ACF_STREAMLIT

Description
Schema created to store the App Control Manager Streamlit UI artifacts.

Stage: ACF_STREAMLIT

Description
Stage created to store the App Control Manager Streamlit UI artifacts.

Contents:

- **Python files** - code that creates and manages the App Control Manager actions.
- **Images** - image files used in the App Control Manager UI
- **Templates** - template files used to create the **manifest.yml**, **setup_script.sql**, **readme.md**, and **load_sidecar.py** files

Streamlit: P_<APP_CODE>_APP_CONTROL_MANAGER

Description
Streamlit in Snowflake object that creates a UI for the Application Control Manager.

Table: COMMANDS

Description
Table that stores commands that the P_<APP_CODE>_APP_CONTROL_MANAGER cannot execute within the UI. This table has a stream called COMMANDS_STREAM that stores new

commands. These commands are executed via the EXECUTE_CMD stored procedure, which is called via one of the PROCESS_COMMANDS_(i) tasks.

Stored Procedure: EXECUTE_CMD

Description:
This procedure executes new commands from the COMMANDS_STREAM stream.

Parameters:
- N/A

Stream: COMMANDS_STREAM

Description
Stream created to identify new commands from the COMMANDS table.

- **Append Only**: True
- **Data Retention Time in Days**: 1

Task: PROCESS_COMMANDS_(i)

Description
Task that processes each new log entry captured in the COMMANDS_STREAM stream.

- **Warehouse**: P_<APP_CODE>_APP_WH
- **Schedule**: 1 minute
- **Action**: Calls the EXECUTE_CMD stored procedure.

NOTE: *i* = the number of tasks created. There are 12 tasks (i = 01 - 12), started 5 seconds apart. This results in one of the tasks checking the COMMAND_STREAM every 5 seconds.

Dev Environment: P_<APP_CODE>_SOURCE_DB_DEV

Description
The “Dev Environment” is the database that includes the source data, functions, and/or procedures that will be included in the provider’s Native App.

Schema: DATA

Description
Schema containing the provider’s source data (if created by the P_<APP_CODE>_ACF_ADMIN role).

Schema: APP

Description
The schema that contains the app_key table and the run_id sequence required for testing.

Table: APP_KEY

Description
The table that stores a dummy app key for testing purposes.

Definition

Column	Data Type	Description	Null?
APP_KEY	VARCHAR	A test app key	N

Table: APP_MODE

Description
The table that stores fields related to which mode the app is in. When in `FREE` mode, this table is used to regulate the terms of usage (i.e. how many records can be processed). Can be helpful for testing functionality, depending on the mode of the app.

Column	Data Type	Description	Null?
KEY	VARCHAR	The metadata key name (i.e. app_mode)	N
VALUE	TIMESTAMP_LTZ(9)	The metadata value (i.e. 'Y')	N

The following keys are inserted into this table:

- **app_mode** - The app's current mode (FREE, PAID, or ENTERPRISE)

Table: LIMIT_TRACKER

Description
The table that locally stores the counts, used to enforce the limits.

Column	Data Type	Description	Null?
KEY	VARCHAR	The metadata key name (i.e. app_mode)	N
VALUE	TIMESTAMP_LTZ(9)	The metadata value (i.e. 'Y')	N

The following keys are inserted into this table:

- **total_requests** - The number of requests made throughout the life of consumer usage of the native app.
- **requests_processed_this_interval** - The number of requests made within the defined LIMIT_INTERVAL (this is defined in the METADATA table)
- **input_records** - The total number of input records submitted
- **input_records_this_interval** - The number of input records submitted this interval (i.e. 30 days)
- **total_records_processed** - The number of records processed throughout the life of consumer usage of the native app.
- **records_processed_this_interval** - The number of records processed within the defined LIMIT_INTERVAL (this is defined in the METADATA table).
- **total_matches** - The number of matches throughout the life of consumer usage of the native app (if applicable).
- **matches_this_interval** - The number of matches within the defined LIMIT_INTERVAL (this is defined in the METADATA table).
- **last_request_timestamp** - The timestamp of the last request

Table: RUN_TRACKER

Description

The table that stores historical runs.

Column	Data Type	Description	Null?
TIMESTAMP	TIMESTAMP_NTZ(9)	The timestamp of the request	N
REQUEST_ID	VARCHAR	The run's request ID	N
REQUEST_TYPE	VARCHAR	The type of request	N
INPUT_TABLE	VARCHAR	The input table	Y
OUTPUT_TABLE	VARCHAR	The output table	Y

Schema: METADATA

Description

Schema containing the metadata-related secure views required for testing.

View: METADATA_V

Description

A view created from the METADATA table containing all metadata required for testing.

Definition

Column	Data Type	Description	Null?
ACCOUNT_LOCATOR	VARCHAR	The consumer's Snowflake Account Locator	N
CONSUMER_NAME	VARCHAR	The consumer's company name	N
KEY	VARCHAR	The metadata key name (i.e. 'enabled')	N
VALUE	TIMESTAMP_LTZ(9)	The metadata value (i.e. 'Y')	N

Schema (Versioned): UTIL_APP

Description
Schema containing utility-type objects, such as the consumer's metadata view, metric views, etc required for testing

View: METADATA_C_V

Description
A view created from the METADATA table containing the test consumer metadata required for testing.

Definition

Column	Data Type	Description	Null?
ACCOUNT_LOCATOR	VARCHAR	The consumer's Snowflake Account Locator	N
CONSUMER_NAME	VARCHAR	The consumer's company name	N
KEY	VARCHAR	The metadata key name (i.e. 'enabled')	N
VALUE	TIMESTAMP_LTZ(9)	The metadata value (i.e. 'Y')	N

Table: REQUEST_ID_TEMP

Description
A table created when calling the REQUEST stored procedure to store the request id required for testing.

Definition

Column	Data Type	Description	Null?
REQUEST_ID	VARCHAR	The request ID generated when calling the app's REQUEST stored procedure	N

Stored Procedure: APP_LOGGER

Description:
This procedure adds event messages to the local events table. This stored procedure is used for testing.

Parameters:

- **account_locator** (VARCHAR) - The consumer's Snowflake Account Locator
- **consumer_name** (VARCHAR) - The consumer's name
- **app_key** (VARCHAR) - The consumer's app key for the installation
- **app_mode** (VARCHAR) - The app's mode (i.e. FREE, PAID, or ENTERPRISE)
- **entry_type** (VARCHAR) - The type of entry (i.e. LOG or METRIC)
- **event_type** (VARCHAR) - The type of event being logged (i.e. INSTALL or REQUEST)
- **event_attributes** (VARCHAR) - Any applicable attributes associated with the event type (can be NULL). **NOTE**: if the message is a string, it should be enclosed in double quotes (i.e. `'"this is a test event"'`).
- **timestamp** (TIMESTAMP_NTZ) - The UTC timestamp for the event
- **status** (VARCHAR) - The status of the event (i.e. PROCESSING, COMPLETE, or ERROR)
- **message** (VARCHAR) - The message to log. **NOTE**: if the message is a string, it should be enclosed in double quotes (i.e. `'"this is my message"'`).

Schema: RESULTS_APP

Description
The schema that stores the results table from calling the provider's stored procedure(s).

Schema: FUNCS_APP

Description
The schema that stores the app's functions to test that are either used by the provider's stored procedures or to be made accessible to the consumer.

Schema (Versioned): PROCS_APP

Description
The schema that contains the stored procedures to test.

Application Package: P_<APP_CODE>_APP_PKG_(PACKAGE_NAME)

Description
The application package that contains the “proxy” objects (secure views) that allow the app access to source data, logs, metrics, and metadata. Secure views are created from each source data, logs, metrics, and metadata table.

In addition, the manifest.yml and setup_script.sql files stored on the app version’s stage are tied to the app package.

Schema: VERSIONS

Description
The schema that contains stages the store manifest.yml and setup_script.sql files.

Stage: (APPLICATION VERSION)

Description
Each stage is a “version” of the app.

Contents:

- **manifest.yml** - contains permissions, streamlit UI settings, etc. for the native app.
- **setup_script.sql** - contains the commands to create the app objects, including the app logic’s stored procedure(s), in the consumer’s account
- **readme_<app_version>-** the readme file for each app version.
- **Streamlit files**

Updates to a version can be “patched”. Please refer to the Deployment Guide for more information.

Schema: EVENTS

Description
The schema that contains a view of the EVENTS_MASTER table.

View (Secure): EVENTS_MASTER_V

Description
A secure view created from the EVENTS_MASTER table containing consumer event messages.

Definition

Column	Data Type	Description	Null?

MSG	VARIANT	Log message from relevant app events.	N

Schema: DATA

Description
Schema containing secure view(s) of the provider's source data.

Schema: METADATA

Description
Schema containing the metadata-related secure views.

View (Secure): METADATA_DICTIONARY_V

Description
A secure view created from the METADATA_DICTIONARY table containing consumer metadata, included in the app package.

Definition

Column	Data Type	Description	Null?
CONTROL_NAME	VARCHAR	The name of the control (metadata key)	N
CONTROL_TYPE	VARCHAR	One of the four types of control: • preventive • detective • deterrent • corrective	N
CONDITION	VARCHAR	The relationship between the control and its default value (i.e. = or <=)	N
DEFAULT_VALUE	VARCHAR	The control's default value	Y
CONSUMER_CONTROL	BOOLEAN	Flag indicating whether the control is set for each consumer	N
SET_VIA_ONBOARD	BOOLEAN	Flag indicating whether the control can be modifiable during the consumer onboarding process. Custom and certain pre-built controls are allowed to be overwritten.	N
CONSUMER_VISIBLE	BOOLEAN	Flag indicating whether or not the control is visible by the consumer via the app's METADATA_C_V	N

		view. These are controls the provider would like to store/track, but hide from the consumer	
DESCRIPTION	VARCHAR	The control's description	N

View (Secure): RULES_DICTIONARY_V

Description

A secure view created from the RULES_DICTIONARY table containing consumer metadata, included in the app package.

Definition

Column	Data Type	Description	Null?
RULE_NAME	VARCHAR	The name of the rule	N
RULE_TYPE	VARCHAR	The type of rule (currently only CUSTOM)	N
RULE	VARCHAR	The JSON string containing the rule's conditions	N
METADATA_USED	VARCHAR	A comma-separated list of metadata keys used by the rule.	N
DESCRIPTION	VARCHAR	The rule's description	N

View (Secure): METADATA_V

Description

A secure view created from the METADATA table containing consumer metadata, included in the app package.

Definition

Column	Data Type	Description	Null?
ACCOUNT_LOCATOR	VARCHAR	The consumer's Snowflake Account Locator	N
CONSUMER_NAME	VARCHAR	The consumer's company name	N
KEY	VARCHAR	The metadata key name (i.e. 'enabled')	N
VALUE	TIMESTAMP_LTZ(9)	The metadata value (i.e. 'Y')	N

Consumer Account

The following objects are created in the consumer's account when the app is installed.

Application: (APPLICATION OBJECT)

Description
The app object that contains the native app objects installed in the consumer's account.

Application Role: APP_ROLE

The app role is granted privileges to objects that should be accessible to the consumer using the app.

There are objects that the app creates that are used by the app, but not accessible to the consumer. The app role will not be granted privileges to these objects.

Schema (Non-versioned): APP

Description
The non-versioned schema that contains the app_key table and the run_id sequence. This schema is non-versioned to avoid updating these objects in the event of an app upgrade.

Table: APP_KEY

Description
The table created during installation, that stores the app key for the consumer's installation. The app key is used to confirm requests are coming from a valid installation.

Definition

Column	Data Type	Description	Null?
APP_KEY	VARCHAR	The consumer's app key for the installation	N

Table: APP_MODE

Description
The table that stores fields related to which mode the app is in. When in `FREE` mode, this table is used to regulate the terms of usage (i.e. how many records can be processed).

Column	Data Type	Description	Null?
KEY	VARCHAR	The metadata key name (i.e.	N

		app_mode)	
VALUE	TIMESTAMP_LTZ(9)	The metadata value (i.e. 'Y')	N

The following keys are inserted into this table:

- **app_mode** - The app's current mode (FREE, PAID, or ENTERPRISE)

Table: LIMIT_TRACKER

Description
The table that stores the limits to be enforced for non-free versions of the app.

Column	Data Type	Description	Null?
KEY	VARCHAR	The metadata key name (i.e. app_mode)	N
VALUE	VARCHAR	The metadata value (i.e. 'Y')	N

The following keys are inserted into this table:

- **total_requests** - The number of requests made throughout the life of consumer usage of the native app.
- **requests_processed_this_interval** - The number of requests made within the defined LIMIT_INTERVAL (this is defined in the METADATA table)
- **input_records** - The total number of input records submitted
- **input_records_this_interval** - The number of input records submitted this interval (i.e. 30 days)
- **total_records_processed** - The number of records processed throughout the life of consumer usage of the native app.
- **records_processed_this_interval** - The number of records processed within the defined LIMIT_INTERVAL (this is defined in the METADATA table).
- **total_matches** - The number of matches throughout the life of consumer usage of the native app (if applicable).
- **matches_this_interval** - The number of matches within the defined LIMIT_INTERVAL (this is defined in the METADATA table).
- **last_request_timestamp** - The timestamp of the last request

NOTE: This is hidden from the consumer.

Table: RUN_TRACKER

Description
The table that stores historical runs.

Column	Data Type	Description	Null?
TIMESTAMP	TIMESTAMP_NTZ(9)	The timestamp of the request	N
REQUEST_ID	VARCHAR	The run's request ID	N
REQUEST_TYPE	VARCHAR	The type of request	N
INPUT_TABLE	VARCHAR	The input table	Y
OUTPUT_TABLE	VARCHAR	The output table	Y

NOTE: This is hidden from the consumer.

Task: COUNTER_RESET_TASK

Description
Task that resets the consumer's counter values, after a specified time has elapsed. This is created when the consumer executes the CONFIGURE_TRACKER stored procedure. This task is only created for PAID and ENTERPRISE versions of the app.

- **Warehouse**: N/A (serverless)
- **Schedule**: '15 minute'
- **Action**: Updates the counters in the LIMIT_TRACKER table. The task also logs a message to reset the counters. Once received in the ACF account, the consumer's counts are reset in the METADATA table.

Schema (Versioned): UTIL_APP

Description
Schema containing utility-type objects, such as the consumer's metadata view, metric views, etc.

Table: ALL_PROCS

Description
The table that contains a list of all of the app logic stored procedures created by the app. These stored procedures will be hidden from the consumer, but accessible via the app's REQUEST stored procedure (described later in this document).

Definition

Column	Data Type	Description	Null?
PROC_NAME	VARCHAR	The name of the app logic stored	N

		procedure	
PROC_SIGNATURE	VARCHAR	The stored procedure's signature	N
REQUIRE_INPUT_TABLE	VARCHAR	Flag indicating whether the stored procedure requires an input table	N

NOTE: This is hidden from the consumer.

View (Secure): ALLOWED_PROCS_V

Description

The secure view created from the ALL_PROCS table that reveals the app logic stored procedure(s) the consumer has access to. The consumer uses this to view the parameters passed to the accessible procedures. The consumer's access to the app logic stored procedures is controlled by the METADATA table.

Definition

Column	Data Type	Description	Null?
PROC_NAME	VARCHAR	The name of the app logic stored procedure	N
PROC_SIGNATURE	VARCHAR	The stored procedure's signature	N
REQUIRE_INPUT_TABLE	VARCHAR	Flag indicating whether the stored procedure requires an input table	N

Table: REQUEST_ID_TEMP

Description

A table created when calling the REQUEST stored procedure to store the request id.

Definition

Column	Data Type	Description	Null?
REQUEST_ID	VARCHAR	The request ID generated when calling the app's REQUEST stored procedure	N

View (Secure): METADATA_C_V

Description

A secure view created from the METADATA_V view in the app package, only containing the consumer's metadata.

Definition

Column	Data Type	Description	Null?
ACCOUNT_LOCATOR	VARCHAR	The consumer's Snowflake Account Locator	N
CONSUMER_NAME	VARCHAR	The consumer's company name	N
KEY	VARCHAR	The metadata key name (i.e. 'enabled')	N
VALUE	TIMESTAMP_LTZ(9)	The metadata value (i.e. 'Y')	N

View: REQUEST_SUMMARY_C_V

Description

View presenting the consumer's metrics entries from the METRICS table, in tabular format.

Definition

Column	Data Type	Description	Null?
ACCOUNT	VARCHAR	The consumer's Snowflake Account Locator	N
CONSUMER_NAME	VARCHAR	The consumer's company name	N
ENTRY_TYPE	VARCHAR	The type of event (i.e. log or metric)	N
REQUEST_ID	VARCHAR	Request ID	N
PROC_NAME	VARCHAR	The allowed procedure called	N
PROC_PARAMETERS	VARCHAR	The allowed procedure's parameters	Y
INPUT_TABLE_NAME	VARCHAR	Input table name	N
INPUT_RECORD_COUNT	NUMBER(38,0)	Input table record count	Y
RESULTS_TABLE_NAME	VARCHAR	Results table name	Y
RESULTS_RECORD_COUNT	NUMBER(38,0)	Results table record count	N
RESULTS_RECORD_COUNT_DISTINCT	NUMBER(38,0)	Results table record count(distinct)	N
STATUS	VARCHAR	Request status	N
COMMENTS	VARCHAR	Comments	Y
SUBMITTED_TS	TIMESTAMP_LTZ(9)	Submitted timestamp	N

COMPLETED_TS	TIMESTAMP_LTZ(9)	Completed timestamp	Y

Stored Procedure: CONFIGURE_TRACKER

Description:
This procedure creates a serverless task that reset the consumer's interval counts according to the interval set. This is only available in PAID and ENTERPRISE versions of the app.

Parameters:
- **N/A**

Stored Procedure: REGISTER_SINGLE_CALLBACK

Description:
This procedure registers an object to be used by the native app (i.e. table or view).

Parameters:
- **ref_name** (STRING) - The reference name
- **operation** (STRING) - The operation to be performed.
- **ref_or_alias** (STRING) - Reference or alias

Stored Procedure: APP_LOGGER

Description:
This procedure adds event messages to the consumer's events table.

Parameters:
- **account_locator** (VARCHAR) - The consumer's Snowflake Account Locator
- **consumer_name** (VARCHAR) - The consumer's name
- **app_key** (VARCHAR) - The consumer's app key for the installation
- **app_mode** (VARCHAR) - The app's mode (i.e. FREE, PAID, or ENTERPRISE)
- **entry_type** (VARCHAR) - The type of entry (i.e. LOG or METRIC)
- **event_type** (VARCHAR) - The type of event being logged (i.e. INSTALL or REQUEST)
- **event_attributes** (VARCHAR) - Any applicable attributes associated with the event type (can be NULL). **NOTE**: if the message is a string, it should be enclosed in double quotes (i.e. `'"this is a test event"'`).
- **timestamp** (TIMESTAMP_NTZ) - The UTC timestamp for the event
- **status** (VARCHAR) - The status of the event (i.e. PROCESSING, COMPLETE, or ERROR)
- **message** (VARCHAR) - The message to log. **NOTE**: if the message is a string, it should be enclosed in double quotes (i.e. `'"this is my message"'`).

NOTE: This is hidden from the consumer.

Stored Procedure: CUSTOM_EVENT_BILLING

Description:
This procedure registers a custom billing event that charges the consumer a specified amount for an event (i.e. $0.05 for each record processed). Custom billing events are useful in the ENTERPRISE app version, where consumers may be charged different rates for billing events.

Parameters: See
https://docs.snowflake.com/en/sql-reference/functions/system_create_billing_event

NOTE: This is hidden from the consumer.

Stored Procedure: LOG_FORM

Description:
This procedure adds contact information, from the app's Streamlit UI, to the events table. This is useful for collecting information from consumers interested in upgrading their app.

Parameters:

- **first_name** (STRING) - The consumer's first name
- **last_name** (STRING) - The consumer's last name
- **title** (STRING) - The consumer's title
- **business_email** (STRING) - The consumer's business email address
- **industry** (STRING) - The consumer's professional industry
- **contact_reason** (STRING) - The reason the consumer is contacting the provider (i.e. app upgrade)
- **contact_reason_text** (STRING) - More details the consumer provides regarding the contact reason.

NOTE: This is hidden from the consumer.

Stored Procedure: LOAD_INSTALL_SQL

Description:
This procedure loads the consumer setup commands into the `SQL.SETUP` table.

Parameters:

- **app_name** (VARCHAR) - The name of the app, as installed in the consumer's account.
- **app_user** (VARCHAR) - The current user

Schema (Non-versioned): SETUP

Description

The schema that stores the table that contains SQL commands to be executed by the SidecarRunner stored procedure.

Table: SQL

Description
A table that contains SQL commands to be executed by the SidecarRunner stored procedure.

Definition

Column	Data Type	Description	Null?
SQL	VARCHAR	The SQL commands to be executed by the SidecarRunner stored procedure.	N

Schema (Non-versioned): RESULTS_APP

Description
The non-versioned schema that stores the results table from calling the allowed stored procedure(s) via the app's REQUEST stored procedure. This schema is non-versioned to avoid losing results tabled in this schema, in the event of an app upgrade.

Table: (RESULTS TABLE)

Description
The resulting table from calling the allowed stored procedure(s) via the app's REQUEST stored procedure.

Schema (Versioned): FUNCS_APP

Description
The schema that stores the app's functions that are used by the consumer and/or the app's stored procedures.

Schema (Versioned): PROCS_APP

Description
The schema that contains the app's stored procedures.

Stored Procedure: LOG_SHARE_INSERT

Description:

This procedure is shared with the app that inserts the initial app install logs into the logs table that is shared from the consumer to the provider.

NOTE: this stored procedure should only be executed after installation or upgrade/downgrade.

Parameters:

- **provider_account_locator** (VARCHAR) - the provider's Snowflake Account Locator
- **app_code** (VARCHAR) - The abbreviated/shorthand name of the provider's app

Stored Procedure: REQUEST

Description:
This procedure is a “helper” stored procedure that allows the consumer to make a request to use an allowed app logic stored procedure. This procedure validates that the consumer can use the app, can access the specified stored procedure, and collects pre-set metrics about the request (i.e. input table record count, result record counts, etc.).

Parameters:

- **app_code** (VARCHAR) - The abbreviated/shorthand name of the provider’s app
- **parameters** (VARCHAR) - The object containing input/output table names, the requested proc, and parameters to pass to the stored procedure.

Consumer Objects

The following objects are created to streamline app usage, but are not required to use the app.

NOTE: References to `<APP_CODE>` refer to the shorthand/abbreviated name for the app (i.e. SDE for an app called Sample Data Enrichment).

Database: SIDECAR

Description
Database created to store the SidecarRunner stored procedure.

Schema: Runner

Description:
This schema stores the SidecarRunner stored procedure.

Stored Procedure: SidecarRunner

Description:
This procedure executes the commands loaded in the `SQL.SETUP` table.

Parameters:

- **app_name** (VARCHAR) - The name of the app, as installed in the consumer's account.

The following objects are created when the consumer executes the SidecarRunner procedure.

Role: C_<APP_CODE>_APP_ADMIN

Description
The admin role is created to create the log share to the provider, along with the stored procedures that help streamline app usage.

Warehouse: C_<APP_CODE>_APP_WH

Description
The warehouse used to create the objects required to interact with the provider's app.

Database: C_<APP_CODE>_HELPER_DB

Description
Database that contains the consumer's source data, shared data, and objects required to interact with the app.

Schema: SOURCE

Description
The schema that contains tables/views created from the source data tables.

Table/View: (SOURCE TABLES/VIEWS)

Description
The tables/views created from the source data tables.

Schema: RESULTS

Description
The schema that stores the results tables from each request, if applicable.

NOTE: This version of the Snowflake Native Apps framework does not support creating objects outside of the app.

Table: (RESULT TABLES)

Description
The results tables from each request.

Schema: PRIVATE

Description
Schema containing the helper stored procedures utilized to streamline the log share and request processes.

Stored Procedure: DETECT_EVENT_TABLE

Description:
This procedure detects if the consumer's account has an event table and creates one if there isn't one present.

Parameters:

- N/A

Database: EVENTS

Description
Database that contains the consumer's event table (if the account does not already have one). Events in this table are shared with the provider to enable app usage (non-free app versions).

Schema: EVENTS

Description:
This schema stores the consumer events.

Table: EVENTS

Description
Table containing the consumer events.

Definition
See https://docs.snowflake.com/en/developer-guide/logging-tracing/event-table-columns for event table definition details.