

/LiveRamp

LiveRamp Embedded Identity in Snowflake v1

Table of Contents

LiveRamp Embedded Identity in Snowflake	3
LiveRamp Native Apps	3
Prerequisites	5
Enabling LiveRamp Embedded Identity in Snowflake	5
Authentication	5
Product Support	5
Set Up a Native App	6
Overall Steps	6
Accept the Snowflake Marketplace Terms and Conditions	6
Install a LiveRamp Native App	7
Open a Native App	12
Set the Variables	14
Enable Authentication	14
Grant Shared Database Permissions	15
Activate Logging and Metrics	15
Perform RampID Transcoding	18
Overall Steps	18
Completion Checklist for Native App Setup	18
Prepare the Tables for Transcoding	19
Table Naming Guidelines	20
Metadata Table Columns and Descriptions	20
Input Table Columns and Descriptions	20
Perform Transcoding	21
Perform Identity Resolution	24
Overall Steps	24
Completion Checklist for Native App Setup	25
Prepare the Tables for Identity Resolution	25
Metadata Table Columns and Descriptions	27
Input Table Columns and Descriptions	28
Input Table Columns for PII Resolution	28
Input Table Columns for Email-Only Resolution	30
Input Table Columns for Device Resolution	31
Input Table Columns for CID Resolution	31
Perform the Identity Resolution Operation	32
View the Output Table	33
View the PII Resolution Output Table	33
View the Email-Only Resolution Output Table	34
View the Device Identifier Resolution Output Table	34
View the CID Resolution Output Table	34
Privacy Filter	34

LiveRamp Embedded Identity in Snowflake

LiveRamp's Embedded Identity is available through native apps in Snowflake's [Marketplace](#).



CAUTION

This content is for customers utilizing the previous versions of LiveRamp's native apps in Snowflake (LiveRamp Identity Resolution and LiveRamp Transcoding). For customers utilizing the LiveRamp Identity native app in Snowflake (starting in October 2023), see [this documentation](#).

There are two available operations that you can perform using the LiveRamp native apps:

- **Identity Resolution**, which allows you to resolve an identifier (a device ID or PII, such as name, address, email, or phone) to LiveRamp's person-based, pseudonymous identifier, RampID. You can also resolve a person-based RampID to a household-based RampID. For more information, see "[Perform Identity Resolution \[24\]](#)".
- **RampID Transcoding**, which transcodes a RampID in one domain to a RampID in another domain. Transcoding converts or translates the pseudonymous person-based identifier (the RampID) for use by another party. For more information, see "[Perform RampID Transcoding \[18\]](#)".

When you install either of these native apps, it creates a set of tables and a schema, and writes a set of procedures into the Snowflake worksheet. To perform an operation, you provide an input table that contains the relevant identifiers and a metadata table that contains information on the operation to be performed.



NOTE

LiveRamp's Embedded Identity in Snowflake is currently in beta. To find out more about participating in this program, contact snowflake@liveramp.com.

LiveRamp Native Apps

Transcoding and identity resolution capabilities are available within Snowflake through a LiveRamp native app, which creates a share to your account, opening up a view to query the reference data set from within your own Snowflake environment. The LiveRamp native apps install from tiles in the Snowflake Marketplace.

For instructions on setting up a native app, see "[Set Up a Native App \[6\]](#)".

A native application has two sides:

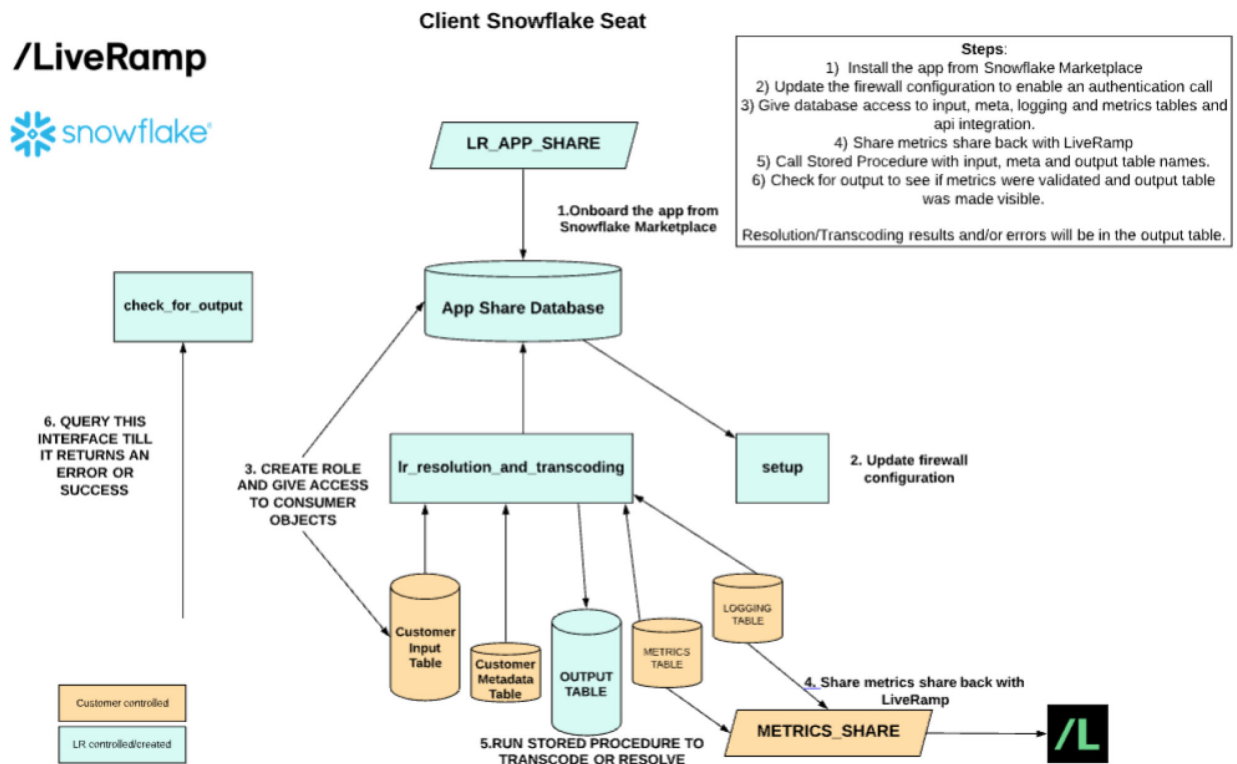
- Provider side (LiveRamp).
- Consumer side (LiveRamp partner).

You perform the operation that the native application enables. Upon initialization, the native app runs an installer script in your account to create an appropriate role and pass the needed stored procedures. The data can then be queried from your account while the reference data set remains in LiveRamp's account. This is accomplished through the secure share capability, enabling view access to the app database.

The LiveRamp native application's architecture is shown in the figure below. This application can perform various operations based on the parameters you specify with metadata: transcoding or identity resolution, for example.

As an example, consider a transcoding operation that converts a RampID in one domain to a RampID in another domain shown in the figure below. This operation requires the permission of the two parties, and once permission is granted, LiveRamp makes available the LR_APP_SHARE to the partner performing the transcode. That share appears in the partner's Snowflake Shares list.

Figure 1. Native application architecture.



**NOTE**

After you've updated the firewall configuration to enable authentication, the authentication call is routed to LiveRamp's GCP instance with the client ID and secret. All data stays in Snowflake.

The sections that follow describe the prerequisites for performing operations, how to enable the operation, and how to get support if you require it.

Prerequisites

The following prerequisites are needed to access LiveRamp in Snowflake:

- Administrator or role access to a Snowflake account. For information, see [“Enabling non-ACCOUNTADMIN Roles to Perform Data Sharing Tasks”](#) in the Snowflake help.
- Permission to create Snowflake roles.

Enabling LiveRamp Embedded Identity in Snowflake

To enable LiveRamp Embedded Identity in Snowflake, the following tasks must be performed:

1. You execute an agreement with LiveRamp to access the service, including the permission required between the parties for transcoding operations, if appropriate (permissions can also be revoked at any time by emailing transcodingpermission@liveramp.com).
2. LiveRamp reviews the use case, including any additional Data Ethics reviews if required.
3. If you don't already have credentials for LiveRamp's Identity API, LiveRamp sends you a client ID and a secret for authentication.
4. LiveRamp sends you the Snowflake account ID/locator for the app to be installed.

Authentication

The LiveRamp Identity Service in Snowflake relies on the same authentication service as LiveRamp's AbiliTec and RampID APIs (Identity APIs). If you have credentials to those APIs, you can use your previously assigned credentials.

Authenticating with LiveRamp's native app service requires a call directly from the client's Snowflake seat to LiveRamp's core services. For information, see the ["Enable Authentication \[14\]"](#) section in ["Set Up a Native App \[6\]"](#).

Client credentials are used to obtain an access token by passing the client ID and client secret values. For information, see ["About Identity Authentication."](#)

Product Support

For support issues, email liverampsnowflakesupport@liveramp.com.

Set Up a Native App in Snowflake

Abstract

Transcoding and identity resolution capabilities are available within Snowflake through native apps, which create a share to your account, opening up a view to query the reference data set from within your own Snowflake environment.

Transcoding and identity resolution capabilities are available within Snowflake through native apps, which create a share to your account, opening up a view to query the reference data set from within your own Snowflake environment.

Once you've completed these steps, you're ready to perform the desired operation. See the articles listed below for the appropriate instructions:

- ["Perform RampID Transcoding \[18\]"](#)
- ["Perform Identity Resolution \[24\]"](#)

Overall Steps

To set up a native app:

1. [Accept the Marketplace terms and conditions \[6\]](#) in the Snowflake UI.



NOTE

This must be performed by a user with an "orgadmin" role.

2. [Install the appropriate LiveRamp native app \[7\]](#).
3. [Set the necessary variables \[14\]](#) that will be used in subsequent steps.
4. [Update the firewall configuration to enable authentication \[14\]](#).
5. [Create and grant permissions to a role \[15\]](#) that gives the App Share database access to your tables (including logging and metrics tables).
6. [Create logging and metrics tables \[15\]](#), add them to a share, and share that share back to LiveRamp. For more information, see the Snowflake help topic "[Working with Shares](#)."

See the sections below for detailed information on performing these steps.

Accept the Snowflake Marketplace Terms and Conditions

To install the LiveRamp native app, an organizational administrator (ORGADMIN) or greater role needs to accept the Snowflake Data Marketplace terms and conditions:

1. Log in to your Snowflake account with the ORGADMIN role.

**NOTE**

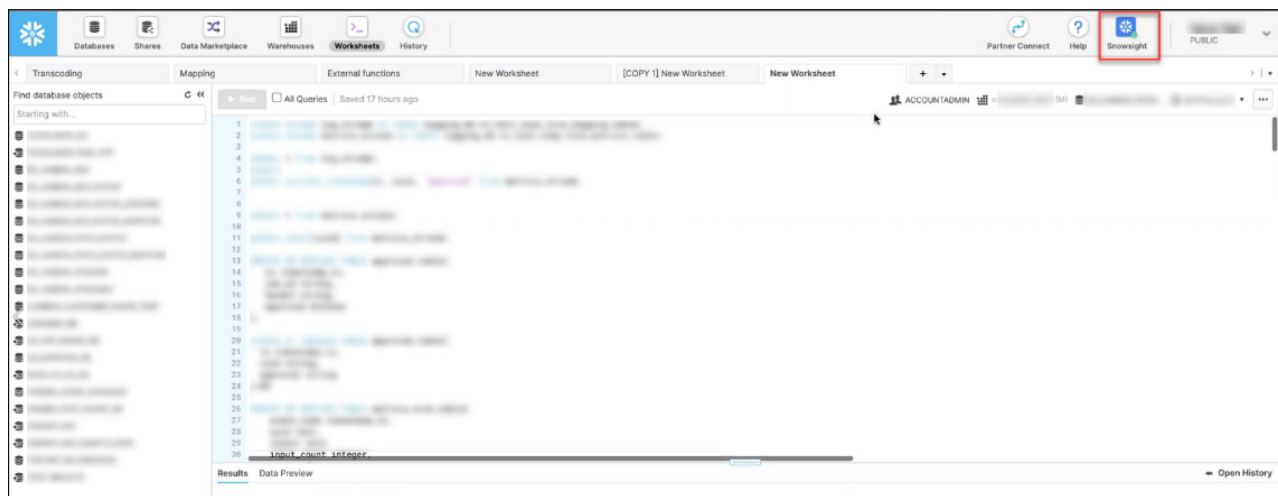
- Alternatively, once logged in click Switch Role next to the login name and select ORGADMIN.
- The ORGADMIN role must have provided their first name, last name, and email address in their account properties. A USERADMIN or another role with OWNERSHIP properties can add these items to the account profile, if required.

2. Click Organizations, and then select Snowflake Data Marketplace Billing.
3. Click Review terms & conditions.
4. In the Review Terms and Conditions dialog box, click the link to review the Terms and Services.
5. If you agree to the terms and conditions, click Accept Terms & Conditions.

Install a LiveRamp Native App

After you've accepted the Marketplace terms and conditions, you install a LiveRamp native app using either the LiveRamp Transcoding tile or the LiveRamp Identity Resolution tile in the Snowflake Marketplace. The instructions that follow detail the process, using the Identity Resolution native app as an example.

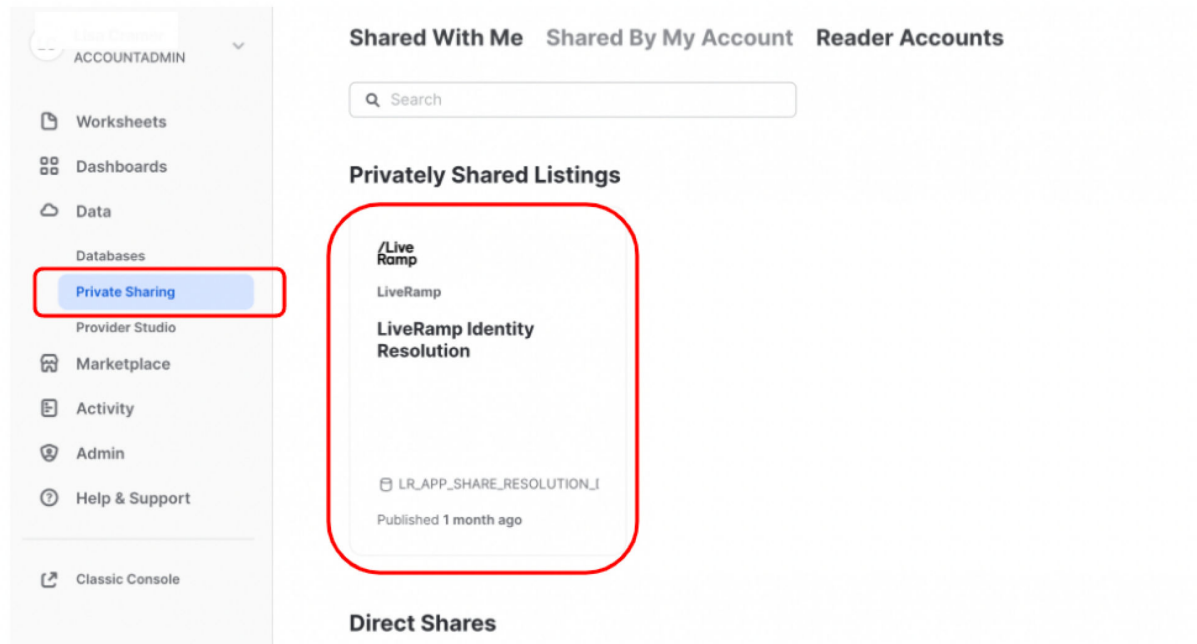
This section details the installation from the Marketplace within the new interface. To switch from the legacy interface to the Snowsight interface, click the Snowsight icon to the left of your account name in the icon bar.

**NOTE**

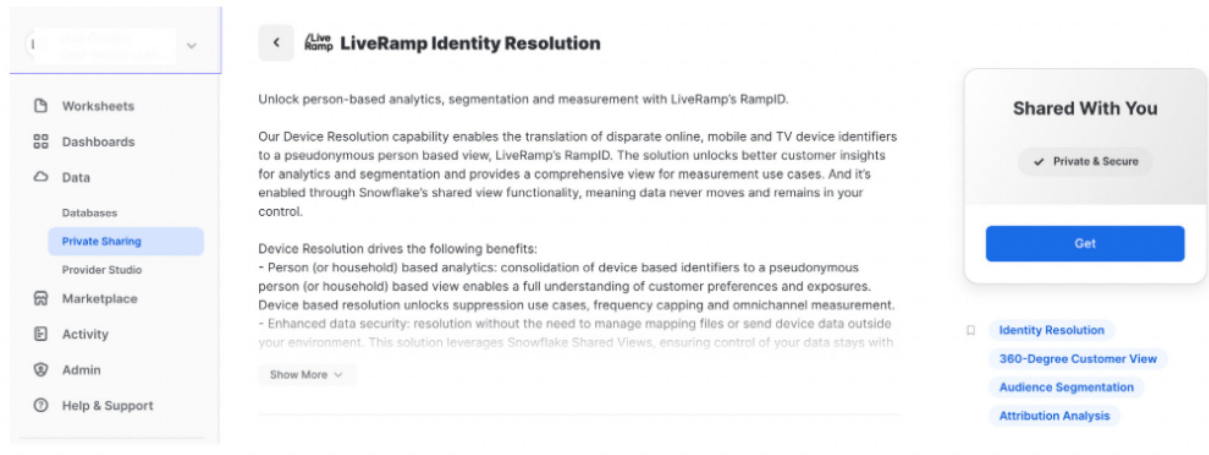
The Snowflake help topic "Installing a Snowflake Native Application from the UI" in Snowflake's native app documentation describes installation in more detail.

To install a native app:

1. In Snowsight, click Private Sharing in the Data section of the left-hand panel.



2. In the Privately Shared Listings section, click the appropriate LiveRamp tile. The listing page appears and describes the native app including several usage examples.



3. Click Get to view the Create Database dialog box shown below.

/Live Ramp

LiveRamp - LiveRamp Identity Resolution

X

Shared Privately with You

Baked-in Governance & Security

Optimized Performance

Fully Managed

LiveRamp Identity Resoluti...

Unlock person-based analytics, segmentation and measurement with LiveRamp's RampID. Our Device Resolution capability enables the translation of disparate online, mobile and TV device identifiers to a...

Warehouse used for installation

Select Warehouse

Options

V

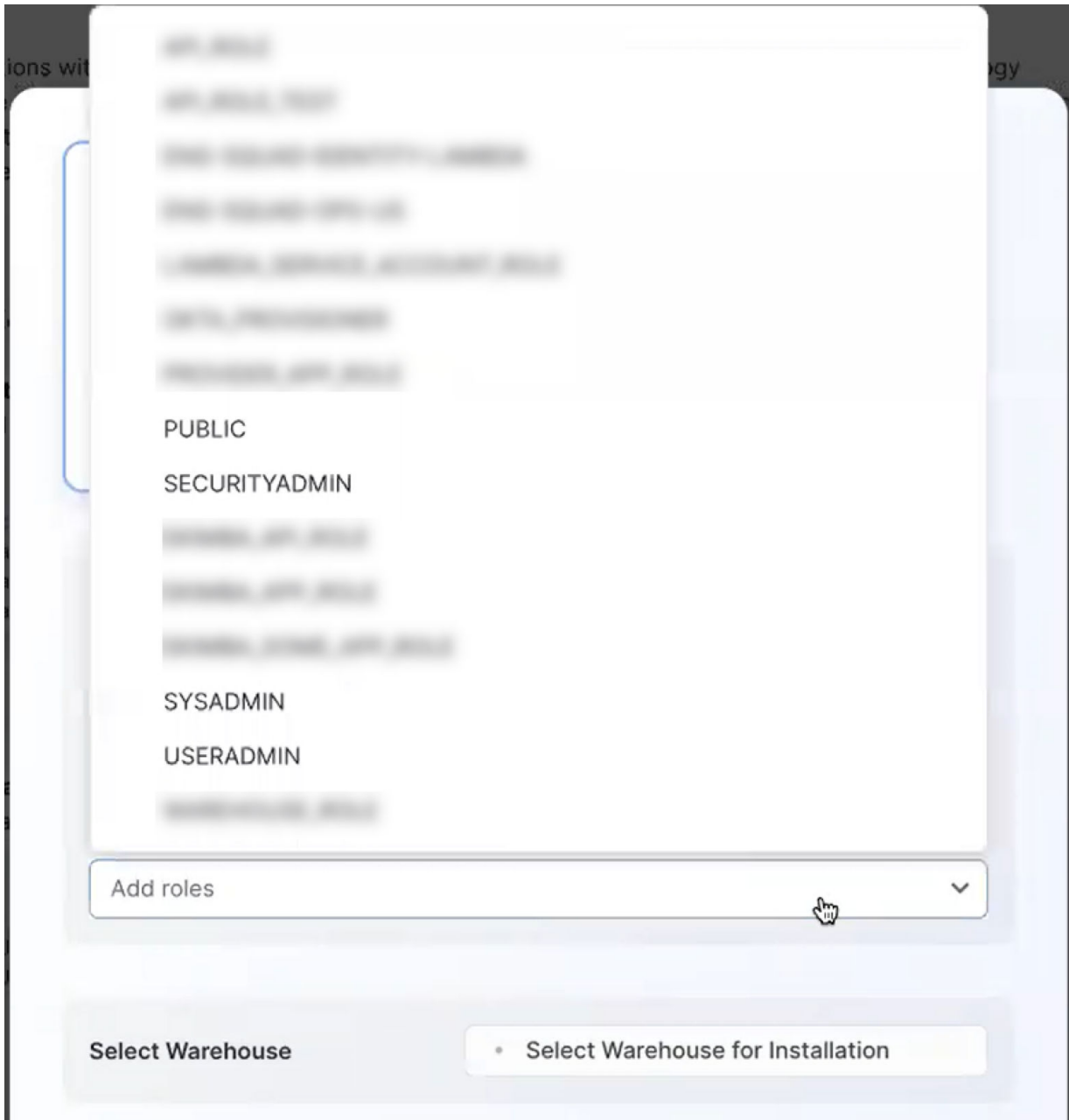
4. In Create Database, enter a name for your application.



NOTE

You will need to use this name as your native app database name later in the process.

- From the Add Roles drop-down menu, select the role(s) you want to be able to perform the identity resolution operations.



TIP

Roles you might want to consider for inclusion are: SYSADMIN, USERADMIN, or maybe even PUBLIC.

6. Click Select Warehouse and select the warehouse you will use to install the application.

Warehouse used for installation

- LAMBDA_XS

Options ^

Database name

LR_APP_SHARE_RESOLUTION

A new database will be created. Consumes no storage.

Which roles, in addition to (ACCOUNTADMIN), can access this database?

ACCOUNTADMIN v

Get

- Click Get.
Snowflake runs the installer script of the app to create a database with the name you specified.
When the installation is complete the Installation Complete dialog box will appear.



LiveRamp Identity Resolution is
now ready to use in your account.

Manage

Done

8. Click Manage to return to the LiveRamp Identity Resolution Marketplace listing page.

After the database is created successfully, the status on that page changes to Installed, and the button to open the app changes to Open as shown on the LiveRamp Transcoding Marketplace home page.

Open a Native App

To open a LiveRamp native app:

1. Click Open on the listing page.

< /Live Ramp **LiveRamp Identity Resolution**

Unlock person-based analytics, segmentation and measurement with LiveRamp's RampID.

Our Device Resolution capability enables the translation of disparate online, mobile and TV device identifiers to a pseudonymous person based view, LiveRamp's RampID. The solution unlocks better customer insights for analytics and segmentation and provides a comprehensive view for measurement use cases. And it's enabled through Snowflake's shared view functionality, meaning data never moves and remains in your control.

Device Resolution drives the following benefits:

- Person (or household) based analytics: consolidation of device based identifiers to a pseudonymous person (or household) based view enables a full understanding of customer preferences and exposures. Device based resolution unlocks suppression use cases, frequency capping and omnichannel measurement.
- Enhanced data security: resolution without the need to manage mapping files or send device data outside

Shared With You

✓ Private & Secure

Open ↗

LR_APP_SHARE_RESOLU...



NOTE

Alternatively in the Snowflake UI click on Private Sharing in the Data section of the left hand panel to return to the listing page, then click Open on the LiveRamp Native tile.

You are taken to the Databases page in the Snowflake UI.

2. Open the LiveRamp Identity Resolution app from the Marketplace listing.
3. Select the database you just created for the app.

The worksheet with the various operations as queries saved as stored procedures is displayed.

```

1  // Create input and metadata table
2  /*
3  Create input and metadata table
4  */
5  --Switch to the database and schema that has your input table
6  USE DATABASE <consumer_database_name>;
7  USE schema <consumer_schema_name>;
8
9  --This is a sample schema, you can bring your own input table
10 --but it should have these three columns(column names can vary
11 --but make sure to reflect it in the metadata table
12 CREATE OR REPLACE TABLE <consumer_database_name>.<consumer_schema_name>.<INPUT_TABLE> (
13   device_id text
14 );
15
16 --Metadata table schema
17 CREATE OR REPLACE TABLE <consumer_database_name>.<consumer_schema_name>.<META_TABLE> (
18   client_id text,
19   client_secret text,
20   ...

```

In this interface, you can run a query or procedure by clicking on the procedure in the worksheet and then clicking the Run button shown boxed in red above.

See Snowflake's [warehouse documentation](#) for more information on the creation and use of warehouses. A warehouse is a specification of computer resources used for operations and follows Snowflake's [sizing and pricing rules](#) for pricing and availability. A Snowflake warehouse will auto-suspend after a certain period of inactivity.

The Input table contains the RampIDs that need to be transcoded or the identifiers that need to be resolved. You also prepare the metadata table which specifies the parameters of the operation, including the type of operation (transcode or resolution).

The results of the operation appear in a read-only output table. The output table contains the converted or transcoded identifiers and lives in your Snowflake instance. Your customer data (in the form of the input table) and the metadata table never leaves your Snowflake instance.

Set the Variables

Set some variables that will be used in the subsequent steps (Identity Resolution is shown as an example):

1. Locate the Defining parameters for the Native App section at the top of the script.

```
set native_app_db_name = 'LIVERAMP_IDENTITY_RESOLUTION';
set native_app_schema_name = 'lr_app_schema';

set role_name = 'PERMISSIONS_ROLE';

set customer_db_name = 'RESOLUTION_DEMO';
set customer_schema_name = concat($customer_db_name, '.', 'PUBLIC');
set customer_input_table_name=concat($customer_schema_name, '.', 'RESOLUTION_INPUT_TABLE');
set customer_meta_table_name=concat($customer_schema_name, '.', 'RESOLUTION_META_TABLE');
set customer_metrics_table_name=concat($customer_schema_name, '.', 'METRICS_TABLE');
set customer_logging_table_name=concat($customer_schema_name, '.', 'LOGGING_TABLE');
set share_name='LIVERAMP_LOG_METRICS_SHARE';
set lr_account='poa18931';
```

2. Make any necessary changes to the variables:



NOTE

Do not change `native_app_schema_name`. Leave this value as `lr_app_schema`.

- `LIVERAMP_IDENTITY_RESOLUTION`: The name of the database native app is loaded to.
- `PERMISSIONS_ROLE`: The name of the role to hand permissions to the native app. The permissions to database objects outside the native app are given to this role. This role is then granted to the native app database.
- `RESOLUTION_DEMO`: The name of your database.
- `PUBLIC`: The name of the schema that holds the tables for identity resolution.
- `RESOLUTION_INPUT_TABLE`: The name of the input table to use for the operation.
- `RESOLUTION_META_TABLE`: The name of the metadata table to use for the operation.
- `METRICS_TABLE`: The name of the metrics table to use for the operation.
- `LOGGING_TABLE`: The name of the logging table to use for the operation.
- `SHARE_NAME`: The name of the share with logging and metrics to be sent to LiveRamp.
- `POA18931`: The LiveRamp account number. Unless your LiveRamp representative indicates otherwise, please leave this as is.

3. Click Run.

Enable Authentication

The LiveRamp [Identity APIs](#) (ID-API) provide identity resolution technology for offline data (personally identifiable information or “PII,” such as email address, name, postal address, etc.). The LiveRamp native app relies on ID-API for authentication and authorization.

To enable authentication:

1. Update the firewall configuration to enable the native app to send requests for authentication:

```
alter database identifier ($native_app_database_name) set firewall_configuration=('https://')
```

2. Click Run.

Grant Shared Database Permissions

To access the required tables (the input and metadata tables), the appropriate permissions must be granted to the native app. The following procedures create the role for access to these tables.



NOTE

The formats of these tables are described in the sections on preparing the tables in "[Perform RampID Transcoding \[18\]](#)" and in "[Perform Identity Resolution \[24\]](#)".

To create and grant usage to the appropriate roles, execute the following:

```
create role if not exists identifier($role_name);
grant usage on database identifier($customer_db_name) to role identifier($role_name);
grant usage on schema identifier($customer_schema_name) to role identifier($role_name);
grant select on table identifier($customer_input_table_name) to role identifier($role_name);
grant select on table identifier($customer_meta_table_name) to role identifier($role_name);
grant select, insert, update on identifier($customer_metrics_table_name) to role identifier($role_name);
grant select, insert, update on identifier($customer_logging_table_name) to role identifier($role_name);
grant role identifier($role_name) to database identifier($native_app_db_name);
```

Activate Logging and Metrics

The native app can log activity to a log table, and aggregate event data into a set of metrics. If you are running multiple operations, logging and metrics can help you better understand your performance.

The log and metrics data are stored in the `logging_table` and `metrics_table` that you create in the `Create Logging and Metrics tables` section in the worksheet shown below.

To activate logging and metrics functionality:

1. Access the LiveRamp native app, click in the `Create Logging and Metrics Table` section shown below, and then click Run.

```
create table if not exists identifier($customer_metrics_table_name)(
  event_time timestamp_tz,
  output_table_name text,
  uuid text,
  tenant text,
  metrics variant,
  job_type varchar(15),
  signature text
```

```
);
```

```
create table if not exists identifier($customer_logging_table_name)(
  ts timestamp_tz,
  uuid string,
  msg string
);
```

2. Verify that the two commands shown below have set `CHANGE_TRACKING = TRUE`. If not, change the value(s) to `TRUE`,

```
-- Alter Shared Table CHANGE_TRACKING to TRUE (Needed for LiveRamp account to create table)
alter table identifier($customer_metrics_table_name) set CHANGE_TRACKING = TRUE;
alter table identifier($customer_logging_table_name) set CHANGE_TRACKING = TRUE;
/*
END
*/
```

3. Click the line with the procedure and then click Run to enforce the change.
4. Run the commands in the Create logging and metrics share section (shown below) and share back the new share with logging and metrics table with LiveRamp.



NOTE

The logging and metrics table needs to be shared back to LiveRamp. LiveRamp validates the metrics and makes the output table visible.

```
// Create logging and metrics share
/*
This step shares back logging and metrics share back to LiveRamp to enable processing at
*/
*/
create share if not exists identifier($share_name);grant usage on database identifier($
grant usage on schema identifier($customer_schema_name) to share identifier($share_name
grant select on identifier($customer_metrics_table_name) to share identifier($share_name
grant select on identifier($customer_logging_table_name) to share identifier($share_name
alter share identifier($share_name) add accounts = $lr_account;
/*
END SECTION
*/
```



NOTE

If the logging share is not shared back with LiveRamp, the output table will not be visible in your Snowflake instance.

The value of the LiveRamp account will be provided to you by your LiveRamp account manager.

Usage data are collected in the metrics table, which is part of your database.

Each record contains:

- A timestamp
- The UUID (transaction ID)
- The tenant identifier
- The number of records processed or not, an error count
- The job type
- An encrypted field that contains all of this information combined

Perform RampID Transcoding in Snowflake

Abstract

LiveRamp's Transcoding application in Snowflake allows for the translation of a RampID from one partner encoding to another using either *maintained* or *derived* RampIDs. This allows you to match persistent pseudonymous identifiers to one another and enables use of the data without sharing the sensitive underlying identifiers.

LiveRamp's Transcoding application in Snowflake allows for the translation of a RampID from one partner encoding to another using either *maintained* or *derived* RampIDs. This allows you to match persistent pseudonymous identifiers to one another and enables use of the data without sharing the sensitive underlying identifiers.



NOTE

For more information about RampID, see "[RampID Methodology](#)".

Specifically, RampID transcoding enables:

- Person-based analytics
- Increased match rates in data collaboration
- Measurement enablement across device types

These capabilities are available within Snowflake through a native app, which creates a share to your account, opening up a view to query the reference data set from within your own Snowflake environment. See "[LiveRamp Embedded Identity in Snowflake \[3\]](#)" for more information.

Overall Steps

After you've set up the transcoding native app in Snowflake (see "[Set Up a Native App \[6\]](#)" for instructions), perform the following steps to perform RampID transcoding:

1. Verify that you've performed all the native app setup tasks in the completion checklist.
2. Prepare the input and metadata tables for transcoding.
3. Perform transcoding.

See the sections below for information on performing these tasks.

Completion Checklist for Native App Setup

Before performing a transcoding operation for the first time, verify that you've completed the following tasks to set up a transcoding native app in Snowflake:

**NOTE**

For instructions on completing these tasks, see "[Set Up a Native App \[6\]](#)".

- ☐ Accepted the terms and conditions of the application.
- ☐ Installed the appropriate LiveRamp Snowflake native application.
- ☐ Updated the firewall configuration for authentication.
- ☐ Created and granted permissions to a role that gives the App Share database access to the appropriate tables.
- ☐ Created logging and metrics tables, added them to a share, and shared that share back to LiveRamp.

After these tasks have been completed, you are ready to prepare the tables and perform the operation. See the sections below for more information.

Prepare the Tables for Transcoding

Transcoding with the LiveRamp native app requires the preparation and deployment of two tables:

- A metadata table, indicated in the code as `<{{transcoding_meta_table}}>`.

**NOTE**

As long as the column names in the input table stay the same, the original metadata table can be reused for multiple operations. You only need to create a new metadata table if you change the column names in the input table.

- An input table, indicated in the code as `<{{transcoding_input_table}}>`.

**NOTE**

An input table needs to be prepared for each transcoding operation.

To create the input and metadata tables:

1. Click in the Create input and metadata table procedure.
2. Click **Run**.
3. Make any necessary changes to the variables:
 - `<SNOWFLAKE_CONSUMER_INPUT_DATABASE>`
 - `<SNOWFLAKE_CONSUMER_SCHEMA>`
 - `<transcoding_input_table>`
 - `<transcoding_meta_table>`

You can create these tables inside Snowflake or import the tables into your database using Snowflake's standard methods. The `<...>` variables may be substituted with your own values. Be sure

to reference the names correctly in the metadata table, which has as its default name **<transcoding_meta_table>**, and make sure that the column names also match up correctly.

When creating tables, keep the following guidelines in mind (in addition to the guidelines listed in the sections below):

- Every column name must be unique in a table.
- Try not to use additional columns in the tables required for the transcode operation. Having extra columns slows down processing.
- The transcode operation can process records containing blank fields.

Table Naming Guidelines

When naming tables, follow these guidelines:

- Table names must use ASCII characters and not contain either spaces or special characters such as !@#\$.%
- You can use underscores "_" within the name, but not as the initial character.
- Consider using the following elements in your table names: type of data or description, a date or timestamp, and an identity designation. For example, the table name `Identity_TwoButton-SuitsCampaign_impressions_2022-06-01` contains all three element types.

Metadata Table Columns and Descriptions

The metadata table passes the required credentials, specifies the type of operation, and specifies the column names in the input table to reference for the original RampIDs, the domain to transcode to, and the identifier type.

As long as the column names in the input table stay the same, the original metadata table can be reused for multiple operations. You only need to create a new metadata table if you change the column names in the input table.

Metadata column names must match those shown in the table below. The column names are not case sensitive, and should not be enclosed in single or double quotation marks.

Table 1. Metadata table columns for transcoding.

Column	Description
CLIENT_ID	Enter either an existing CLIENT_ID or a new one provided in implementation.
CLIENT_SECRET	Password/secret for the CLIENT_ID.
EXECUTION_MODE	Transcoding.
EXECUTION_TYPE	Transcoding.
TARGET_COLUMN	Enter the column name of the input table which contains the RampIDs to be transcoded.
TARGET_DOMAIN_COLUMN	Enter the column name of the input table which contains the target domain for the encoding the RampIDs should be translated to.
TARGET_TYPE_COLUMN	Enter the column name of the input table which contains the target identifier type.

Input Table Columns and Descriptions

An input table needs to be prepared for each transcoding operation.

The column names for the input table can be whatever you want to use, as long as the names match the values specified in the metadata table.

Table 2. Input table columns for transcoding.

Column	Sample	Description
rampid	XYT999RkQ3MEY1RUYtNUIy- Mi00QjJGLUFDNjgtQjQ3QUEwMTNEM- TAlCgMjVBMkNEMTktRD	RampID (maintained or derived) for translation.
Target Do- main Col- umn	T001	Target domain: <ul style="list-style-type: none"> • Enter a partner's domain when translating from your native encoding to that partner's domain. • Enter your domain when translating from a partner's encoding to your native encoding.
ID Type	RampID	Target type. Currently only "RampID" is supported.

The output table is created by the operation that you run. For an example, see the "[Perform Transcoding \[21\]](#)" section below.

**NOTE**

You can transcode both maintained RampIDs and derived RampIDs in your table. For more on RampID types and versions, see "[RampID](#)".

Perform Transcoding

You perform a transcoding operation by running the transcoding procedure and then checking that the output has succeeded. You then open the output table to check the results.

To transcode the RampIDs in the input table:

1. Select the input table in the left-hand Database Objects navigation pane.
2. Click on the **magnifying glass icon** tab shown in the figure below.

Figure 2. Results of the transcode operation.

The screenshot displays the 'LiveRamp Transcoding: Secure Identity Translation Technology - Examples' interface. It features a table with the following data:

RAMPID	TARGET_DOMAIN	TARGET_TYPE
1	XYMP01F3CKsO9j8oNguOijtze4XVsVOCM8qs_edwP998z4tys	T030
2	XYMP01dfqMPAz6Hp7LjSWOVaPL_QI1y8rIGj_BIBxCLyLdcdi	T030
3	XIMP01QuADsbU1NBbOTXJiUx7AD4Mk-K1SqP8BaAJcE43Bn4sFFU5iPIVcVhs3UmJ8kASR	T030
4	XYMP01NB8rph1fUw8eREFJp_ExuPIEBevpe22QcSmXjnEvK4	T030
5	XYMP01a13mMCnUBE4ZTYHFNAHa0fwf9ICFB9gvxh87CQ_aeTo	T030
6	XYMP019oFTMPenrkREEOMM6nJQRuCswwXKntbPE82PehiWGC	2107
7	XYMP01xcX2GcCFGWtVtZPchgXD6s6aFqMcdxaO7aAScTr71Q	2107
8	XYMP01f6-Nt157mSVvu1nL3VtDRBBQ9pLIEYrsPKRcvE4qaiA	2107
9	XYMP01H9EP8p74yGirOoOJbRVIabGns_3Eg7kN9QKU0g6Etto	2107
10	XYMP01amLDo9JYZJU3Dpq8gKW3QPTeHQ118VRGazF3pGI0d8	T030
11	XYMP01XhS1WAdpBDohjoDjDir-NE8e7UqBfkrYgJdEEZef50	T030
12	XYMP01DSLe3rL02ubXJZCvWbt7I2lo1SgRkwwuqEH-r9airNA	T030
13	XIMP01DNvg9bvswq1SzMbtFm8w5lpmwQCq8p8P7UrmVlqKFESZuLF0Y64i3GptzcA1A15	T030
14	XYMP01jwbsczGMqnxhiadNGkrfUJ87eUcokRy5ME2BTID-W8U	2107
15	XYMP01BaH28IfxbHrdJXhT8q9MJcn1FDMHM2wQOEbNi0sKI	2107
16	XYMP011KGeomQPJ266ZE5Mi48ZMPy_w8Wf8LhBLWS9ZZmgZI	2107
17	XYMP019R9TVZXK1ZE6icLUJbHuW4qm_nuuEINkaMGotVExxE	2107

Below the table, the SQL query editor shows the following code:

```

call <native_app_database_name>.lr_app_schema.lr_resolution_and_transcoding(
  '<consumer_database_name>.<consumer_schema_name>.<INPUT_TABLE>',
  '<consumer_database_name>.<consumer_schema_name>.<META_TABLE>',
  '<OUTPUT_TABLE>',
  '<consumer_database_name>.<consumer_schema_name>.<LOGGING_TABLE>',
  '<consumer_database_name>.<consumer_schema_name>.<METRICS_TABLE>'
);

```

Run the transcoding operation specific for your database and schema to convert your input table ID values using the parameters in the metadata table for conversion to new RampIDs in the output table.

3. Locate the `lr_resolution_and_transcoding` procedure shown below and run that SQL.

```

call <native_app_database_name>.lr_app_schema.lr_resolution_and_transcoding
(
  '<consumer_database_name>.<consumer_schema_name>.<INPUT_TABLE>',
  '<consumer_database_name>.<consumer_schema_name>.<META_TABLE>',
  '<OUTPUT_TABLE>',
  '<consumer_database_name>.<consumer_schema_name>.<LOGGING_TABLE>',
  '<consumer_database_name>.<consumer_schema_name>.<METRICS_TABLE>'
);

```

4. Ensure that you have correctly entered the names of the database objects you are using into the procedure above.
5. Click Run.
The transcode operation runs to completion.
6. Open the LiveRamp app worksheet.
7. Locate the Check for output procedure shown below and run that SQL.

```

// Check for output
/*
Check for output
*/
call <native_app_database_name>.lr_app_schema.check_for_output(
  '<OUTPUT_TABLE>',

```

```
);
/*
END SECTION
*/
```

8. Click **Run**.
9. Wait until Snowflake returns a status message of `success` or `error`.
Once the app returns a `success` message, the output should be displayed in the native app database under `lr_app_schema`.

The results end up in the output table in the same database, the `TRANSCODING_TEST_TABLE` shown in the example above (see Step 2).

Table 3. Output table columns for transcoding.

Column	Sample	Description
RampID	XYT999RkQ3MEY1RUYtNUIy-Mi00QjJGLUFDNjgtQjQ3QUEwMTNEM-TA1CgMjVBMkNEMTktRD	Returns the original RampID included in the input table.
Transcoded_identifier	XYT001k0MS00MDc1LUI4NjEtMjl-COUI0MUY3MENBCgNjVGQjE0MTMtRkFBMC00Qz1ELUJF	Transcoded RampID or NULL (NULL due to unreadable native RampID or unauthorized domain, etc.).

Perform Identity Resolution in Snowflake

Abstract

LiveRamp's Identity Resolution capability in Snowflake allows for the translation of various identifiers to *RampIDs*. This allows you to resolve personally-identifiable information (PII) or device identifiers to a persistent pseudonymous identifier for persons and households. You can also input an individual-based RampID and get back any household-based RampID that might be associated with that individual.

LiveRamp's Identity Resolution capability in Snowflake allows for the translation of various identifiers to *RampIDs*. This allows you to resolve personally-identifiable information (PII) or device identifiers to a persistent pseudonymous identifier for persons and households. You can also input an individual-based RampID and get back any household-based RampID that might be associated with that individual.



NOTE

For more information about RampID, see "[RampID Methodology](#)".

Once you've translated your data to RampIDs, you can then share that data to your LiveRamp account for activation. For more information, see "[Share Data from Snowflake to Your LiveRamp Account](#)".

The following identifiers can be resolved:

- Names
- Postal addresses
- Email addresses
- Phone numbers
- Cookies
- MAIDs (mobile device IDs)
- CTV IDs (Connected TV Device IDs)
- CIDs (custom identifiers)
- Person-based, maintained RampIDs (for resolution to household RampIDs)

These capabilities are available within Snowflake through a native app, which creates a share to your account, opening up a view to query the reference data set from within your own Snowflake environment. See "[LiveRamp Embedded Identity in Snowflake \[3\]](#)" for more information.

Overall Steps

After you've set up the identity resolution native app in Snowflake (see "[Set Up a Native App \[6\]](#)" for instructions), perform the following steps to perform identity resolution:

1. Verify that you've performed all the native app setup tasks in the completion checklist.
2. Prepare and deploy the appropriate input and metadata tables for identity resolution.
 - **To resolve PII identifiers** (such as name, address, phone, and/or email), follow the instructions on input table creation in the ["Input Table Columns for PII Resolution \[28\]"](#) section below.
 - **To resolve SHA-256 hashed email addresses only**, follow the instructions on input table creation in the ["Input Table Columns for Email-Only Resolution \[30\]"](#) section below.

**NOTE**

For email-only data, this execution type may be more performant than the PII resolution process, depending on your warehouse set up.

- **To resolve device identifiers** (such as mobile device IDs, CTV IDs, or cookies) **or to resolve an individual RampID to a household RampID**, follow the instructions on input table creation in the ["Input Table Columns for Device Resolution \[31\]"](#) section below.
 - **To resolve custom identifiers** (CIDs), follow the instructions on input table creation in the ["Input Table Columns for CID Resolution \[31\]"](#) section below.
3. Perform the appropriate identity resolution process, depending on the identifiers being resolved:
 4. View the output table.

See the sections below for information on performing these tasks.

Completion Checklist for Native App Setup

Before performing an identity resolution operation for the first time, verify that you've completed the following tasks to set up an identity resolution native app in Snowflake:

**NOTE**

For instructions on completing these tasks, see ["Set Up a Native App \[6\]"](#).

- ☐ Accepted the terms and conditions of the application.
- ☐ Installed the appropriate LiveRamp Snowflake native application.
- ☐ Updated the firewall configuration for authentication calls.
- ☐ Created and granted permissions to a role that gives the App Share database access to the appropriate tables.
- ☐ Created logging and metrics tables, added them to a share, and shared that share back to LiveRamp.

After these tasks have been completed, you are ready to prepare the tables and perform the operation. See the sections below for more information.

Prepare the Tables for Identity Resolution

Performing identity resolution requires two tables:

- A metadata table, indicated in the code as `$customer_meta_table_name`.

**NOTE**

A metadata table can be reused for multiple operations, but a separate metadata table must be prepared for each different identity resolution operation you want to perform. For example, if you're going to perform identity resolution on both MAIDs and hashed emails, you'll need a different metadata table for each operation.

- An input table, indicated in the code as `$customer_input_table_name`.

**NOTE**

An input table needs to be prepared for each identity resolution operation and can only contain one type of identifiers.

To create the input and metadata tables:

1. Click in the Create input and metadata table procedure.

```
set customer_db_name = 'RESOLUTION_DEMO';
set customer_schema_name=concat($customer_db_name,'.','public');
set customer_input_table_name=concat($customer_schema_name,'.','RESOLUTION_INPUT_TABLE');
set customer_meta_table_name=concat($customer_schema_name>','.','RESOLUTION_META_TABLE');
set customer_metrics_table_name=concat($customer_schema_name,'.','METRICS_TABLE');
set customer_logging_table_name=concat($customer_schema_name,'.','LOGGING_TABLE');
set customer_output_table_name='RESOLUTION_OUTPUT_TABLE';
```

2. Make any necessary changes to the variables:
 - `RESOLUTION_DEMO`: The name of your database.
 - `public`: The name of the schema that holds the tables for identity resolution.
 - `RESOLUTION_INPUT_TABLE`: The name of the input table to use for the operation.
 - `RESOLUTION_META_TABLE`: The name of the metadata table to use for the operation.
 - `METRICS_TABLE`: The name of the metrics table to use for the operation.
 - `LOGGING_TABLE`: The name of the logging table to use for the operation.
 - `RESOLUTION_OUTPUT_TABLE`: The name of the output table that will be created after the identity resolution operation has been run.
3. Click Run.

You can create these tables inside Snowflake or import the tables into your database using Snowflake's standard methods. The variables should be replaced with your own values. Make sure to reference the names correctly in the metadata table and make sure that the column names also match up correctly.

When creating tables, keep the following guidelines in mind:


- Every column name must be unique in a table.
- Try not to use additional columns in the input tables required for the identity resolution operation.
- Having extra columns slows down processing.
- Per Snowflake guidelines, table names cannot begin with a number.

Metadata Table Columns and Descriptions

A metadata table can be reused for multiple operations, but a separate metadata table must be prepared for each different identity resolution operation you want to perform. For example, if you're going to perform identity resolution on both MAIDs and hashed emails, you'll need a different metadata table for each operation.

Metadata column names must match those shown in the table below. The column names are not case sensitive, and should not be enclosed in single or double quotation marks.

See the table below for a list of the metadata table columns and descriptions:

Example		
CLIENT_ID	liveramp_client	
CLIENT_SE- CRET	84159be2- ab93-4bf8-24c9-2g123ef08815	
EXECU- TION_MODE	Resolution	Resolution is the only option.
EXECU- TION_TYPE	PII	Options include: <ul style="list-style-type: none"> • PII • Cookies • MAID • CTV • Email • CID name (provided to you by LiveRamp and specific to the mapping created) • HHLink (to resolve individual RampIDs into household RampIDs)
<div>  <div> NOTE Currently, each identifier type has to be separated into its own input table (including CID types) and only one option above can be chosen for each metadata table. </div> </div>		

Example		
TARGET_COLUMN	{ "name": ["FIRSTNAME", "LAST-NAME"], "streetAddress": ["ADDRESSLINE"], "city": "CITY", "state": "STATE", "zipCode": "ZIPCODE" }	<p>Enter the column name for the input table column(s) which contains the IDs to be resolved.</p> <p>For PII resolution, enter a string that maps each identifier type to the column names for that identifier, separated by commas and enclosed with curly brackets, as shown in the example. When including multiple column names per identifier type, enclose those column names in straight brackets.</p> <p>Inputs for PII can include:</p> <ul style="list-style-type: none"> • name: [First Name, Middle Name, Last Name] • streetAddress: [ADDRESSLINE1, ADDRESSLINE2] • zipCode: [zipcode] • city: [city] • state: [state] • phone: [phone] • email: [email]
LIMIT	5	<p>For email or PII resolution only, you can specify the maximum number of RampIDs to be returned.</p> <p>Enter an integer between 1 and 10 to specify the maximum number of RampID results returned per input identifier (to return only the "best match", returning 1 RampID is sufficient).</p> <p>For other identifiers (device identifiers or CIDs), do not include this column.</p>

**NOTE**

If resolving street addresses, zipcode is a required field along with either name or address.

Input Table Columns and Descriptions

The column names for the input table can be whatever you want to use, as long as the column name(s) for the identifiers matches the values specified in the TARGET_COLUMN column of the metadata table. Do not use any column names that are the same as the columns names returned in the output table for the identity resolution operation you're going to run.

See the sections below for suggested input table columns and descriptions for each resolution type.

The output table is created by the operation that you run. For an example, see the sections in "[View the Output Table \[33\]](#)" below.

Input Table Columns for PII Resolution

The PII resolution process operates similarly to device identifier and CID resolution, with a key difference in the data output: instead of returning the original identifier mapped to its associated RampID, data running through PII resolution will pass through a privacy filter which removes the PII and reswizzles the table. Because of this, any attributes you need to keep associated with the identifier need to be included in the input table. For more information, see the "[Privacy Filter \[34\]](#)" section below.



NOTE

- This offering includes running our Identity Graph on demand. Our testing indicates that a warehouse size of 4XL that is not being used for any other operations performs best for average workloads, but this is dependent on your individual setup. Contact your LiveRamp representative to understand the performance implications of your setup.
- When resolving email data only, using the email-only resolution operation can provide higher throughput compared to full PII resolution. This is dependent on warehouse and setup, so talk with your LiveRamp team to determine the best approach for the use case.

These column names cannot be used in the input table for PII resolution:

- RampID
- Rank
- Filter Name

See the table below for a list of the suggested input table columns and descriptions for PII resolution.

Suggested Column Name	Example	
first_name	John	You can include separate First Name and Last Name columns or you can combine first name and last name in one column (such as "Name").
last_name	Doe	You can include separate First Name and Last Name columns or you can combine first name and last name in one column (such as "Name").
address_1	123 Main St	
address_2	Apt 1	You can include separate Address 1 and Address 2 columns or you can combine all street address information in one column (such as "Address").
city	Smalltown	When matching on address, City is optional.
state	CA	<ul style="list-style-type: none"> • When matching on address, State is optional. • If including State, must be a two-character, capitalized abbreviation ("CA", not "California" or "Ca").
zip	12345	<ul style="list-style-type: none"> • Required when matching on addresses. • Can be in 5-digit format or 9-digit format (ZIP+4).
email	john@email.com	<ul style="list-style-type: none"> • Plaintext emails only. • Only one email per input row is permitted. Other emails must be dropped or included in an additional row. If you include an additional row, repeat the values for the name fields for the best match rates. • All emails must meet these requirements: <ul style="list-style-type: none"> • Have characters before and after the "@" sign • Contain a period character (".") • Have characters after the period character • Examples of valid emails include: <ul style="list-style-type: none"> • a@a.com • A@A.COM • email@account.com • EMAIL@ACCOUNT.COM • email@sub.domain.com • EMAIL@SUB.DOMAIN.COM

Suggested Column Name	Example	
phone	555-123-4567	<ul style="list-style-type: none"> • Plain text phone numbers only. • Only one phone number per input row is permitted. Other phone numbers must be dropped or included in an additional row. If you include an additional row, repeat the values for the name fields for the best match rates. • All phone numbers must meet these requirements: <ul style="list-style-type: none"> • Can be more than 10 characters if leading numbers over 10 characters are "0" or "1" • If no leading numbers are used, must be 10 characters long • Can contain hyphens ("-"), parentheses ("(" or ")", plus signs ("+"), and periods (".") • Examples of valid phone numbers include: <ul style="list-style-type: none"> • 8668533267 • 866.853.3267 • (866) 853-3267 • 8668533267 • +1 (866) 853-3267 • +18668533267 • 18668533267 • 1111111118668533267 • 08668533267 • Examples of invalid phone numbers include: <ul style="list-style-type: none"> • 987654321 (fewer than 10 characters) • 98765432109 (more than 10 characters) • 1234567890 (after removing the leading "1", less than 10 characters remain) • 0987654321 (after removing the leading "0", less than 10 characters remain)
attribute_1		For PII resolution, you can include columns with attribute data. These columns will be returned in the output table (for more information, see the "View the PII Resolution Output Table [33]" section below).

Input Table Columns for Email-Only Resolution

The email-only resolution process operates similarly to device identifier and CID resolution, with a key difference in the data output: instead of returning the original identifier mapped to its associated RampID, data running through email-only resolution will pass through a privacy filter which removes the PII and reswizzles the table. Because of this, any attributes you need to keep associated with the identifier need to be included in the input table. For more information, see the ["Privacy Filter \[34\]"](#) section below.



NOTE

- When resolving email data only, using email-only resolution can provide higher throughput compared to full PII resolution. This is dependent on warehouse and setup, so talk with your LiveRamp team to determine the best approach for the use case.
- To perform identity resolution across additional PII touchpoints, see the ["View the PII Resolution Output Table \[33\]"](#) section above.

See the table below for a list of the suggested input table columns and descriptions for email-only resolution.

Suggested Column Name	Example	Description
hashed_email	a64bleicQ3MEYck2Yt-NUIyco1500QjJGLUs-zlUGlNjgtQjQ3QUEwMTNEM-TA1CgNE	<ul style="list-style-type: none"> • SHA-256 hashed emails only. • Email addresses should be uppercased and UTF-8 encoded prior to hashing. • After hashing, convert the resulting hash into lowercase hexadecimal representation.
Attribute_1	Male	For email address resolution, you can include columns with attribute data. These columns will be returned in the output table (for more information, see the " View the Email-Only Resolution Output Table [34] " section below).

Input Table Columns for Device Resolution

The device resolution operation can be used for the following purposes:

- To translate device identifiers (cookies, MAIDs, and CTV IDs) into individual RampIDs
- To translate individual RampIDs into their associated household RampIDs

See the tables below for a list of the suggested input table columns and descriptions for these device resolution options.



NOTE

- Each device resolution input table should contain only one identifier column (either a device identifier or a maintained RampID).
- You can include columns with attribute data, but these columns will not be returned in the output table.

See the table below for a list of the suggested input table columns and descriptions for translating device identifiers.

Suggested Column Name	Example	Description
device_identifier	1f4d256c-1f08-41f6-a108-bbe511de9497	<p>Can be one of the following identifiers:</p> <ul style="list-style-type: none"> • Cookie • MAID • CTV ID

See the table below for a list of the suggested input table columns and descriptions for translating individual RampIDs into their associated household RampIDs.

Suggested Column Name	Example	Description
RampID	XYT999wXyWPB1SgpMUKlpzA013Ua-LEz2lg0wFAR1PWK7FMhsd	<ul style="list-style-type: none"> • The RampID for translation to a Household RampID. • Must be a maintained RampID (to have an associated with a Household RampID).

Input Table Columns for CID Resolution

See the table below for a list of the suggested input table columns and descriptions for CID resolution.

**NOTE**

You can include columns with attribute data, but these columns will not be returned in the output table.

Suggested Column Name	Example	Description
cid	b916clarib la1;bINj10gtQjQ3QUEwMTNEMTcakt-boEc0g9022cxoiaklr20185	The CID for translation to a RampID

Perform the Identity Resolution Operation

Once you've completed the previous steps, you're ready to perform the identity resolution operation.

You perform an identity resolution operation by running the identifier resolution procedure shown below, which includes checking that the output has succeeded. You can then view the output table to check the results.

The output tables vary somewhat, depending on the type of identifiers being resolved.

To perform the identity resolution operation:

1. On your worksheet, switch to the native app database and LR_APP_SCHEMA under it.
2. Locate the `lr_resolution_and_transcoding` procedure shown below and click on that code block.

```
call lr_resolution_and_transcoding
(
    $customer_input_table_name,
    $customer_meta_table_name,
    $output_table_name,
    $customer_logging_table_name,
    $customer_metrics_table_name
);
```

3. Click Run.
The resolution operation runs to completion.

4.


```
// Check for output
/*
Check for output
*/
-
```

```
call check_for_output(
$output_table_name
);
/*
END SECTION
*/
```

5. Click Run.

- Once Snowflake returns a status message of `success`, you can view the output table in the native app database under `lr_app_schema`. See the sections below for more information.



NOTE

If an `error` status message is returned, run the `Check for output` procedure again.

The results end up in the output table in the same database, with the fields shown in the appropriate section below.

View the Output Table

The identity resolution results end up in the output table in the same database you specified previously.

Once you've confirmed that the output table has been generated, see the appropriate section below for information on the output table format for the type of identity resolution operation that was run.

View the PII Resolution Output Table

The PII resolution process passes the input table through a privacy filter which removes the PII and reswizzles the table (in addition to other operations). Because of this, any attributes you need to keep associated with the identifier need to be included in the input table. For more information, see the "[Privacy Filter \[34\]](#)" section below.

Identity resolution of PII provides supplemental match metadata for additional insight into customer data that can provide powerful signals for making decisions based on RampIDs.

For PII resolution, the output table includes the fields shown in the table below.

Column	Sample	Description
<code>Ramp_ID</code>	XYT999wXyWPB1SgpMUKlp-zA013Ua-LEz2lg0wFAR1PWK7FMhsd	Returns the resolved RampID in your domain encoding.
<code>attribute_1</code>	Male	Any attribute columns passed through the service are returned.
<code>__lr_rank</code>	1	Provides insight on the match cascade level associated with the identifiers. If no maintained RampID is found, this value will be "null".
<code>__lr_filter_name</code>	name_phone	Returns the filter name where the match occurred, which will be one of the following options: <ul style="list-style-type: none"> <code>name_address_zip</code> <code>name_email</code> <code>name_phone</code> <code>partial_name_email</code> <code>partial_name_phone</code> <code>strict_name (name + zip)</code> <code>email</code> <code>phone</code> <code>last_name_address</code> If no maintained RampID is found, this value will be "null".

View the Email-Only Resolution Output Table

The email-only resolution process operates similarly to device identifier and CID resolution, with a key difference in the data output: instead of returning the original identifier mapped to its associated RampID, data running through email-only resolution will pass through a privacy filter which removes the PII and reswizzles the table. Because of this, any attributes you need to keep associated with the identifier need to be included in the input table. For more information, see the "[Privacy Filter \[34\]](#)" section below.

For email-only resolution, the results end up in the output table in the same database, with the following fields (as shown below):

- RAMPID (resolved email data)
- ATTRIBUTES (based on other data passed through the service).

Column	Sample	Description
Ramp_ID	XYT999wXyWPB1SgpMUKlpzA013Ua-LEz2lg0wFAR1PWK7FMhsd	Returns the resolved RampID in your domain encoding.
Attribute 1	Male	Any attribute columns passed through the service are returned.

View the Device Identifier Resolution Output Table

For device identifier resolution, the results end up in the output table in the same database, with the following fields (as shown below):

- DEVICE_ID (original input)
- RAMPID (converted values).

Device_ID	93abc799-a0a5-40b5-80dd-d2ab61d4d072
Ramp_ID	XYT999wXyWPB1SgpMUKlpzA013UaLEz2lg0wFAR1PWK7FMhsd

View the CID Resolution Output Table

For CID resolution, the results end up in the output table in the same database, with the following fields (as shown below):

- CID_ID (original input)
- RAMPID (converted values).

Column	Sample	Description
CID_ID	93abc799-a0a5-40b5-80dd-d2ab61d4d072	Original identifier passed in.
Ramp_ID	XYT999wXyWPB1SgpMUKlpzA013Ua-LEz2lg0wFAR1PWK7FMhsd	Returns the resolved RampID in your domain encoding.

Privacy Filter

To minimize the risk of re-identification (the ability to tie PII directly to a RampID), the service includes the following processes when resolving PII identifiers (PII resolution or email-only resolution):

- **Column Values:** The process evaluates the combination of all the column values on a per row basis for unique values. If a particular combination of column values occurs 3 or fewer times, the rows containing those column values will not be matchable and will not be returned in the output table.

- **>5% of the table unmatchable:** If, based on column value uniqueness, >5% of the file rows are unmatchable, the job will fail.
- **Number of Unique RampIDs:** If fewer than 100 unique RampIDs would be returned, the job will fail.
- **Reswizzle full table:** Upon completion, the full table will be reswizzled to return the rows `RampID` | `attribute_1` | `attribute_2` | `attribute_n` in a different order than what was submitted in the input table.