π - Ingest

Version : 01
Author : Pibythree Developers

Table of Contents

Table of Contents 1
1 Introduction 1
2 Key Considerations 1
2.1 Scope 1
2.2 Out of Scope 1
2.3 Assumptions 1
2.4 Dependencies 2
2.5 Risks 3
3 Business Feature Details 4
3.1 Business Process Flow 4
3.2 RBAC & UBAC 4
3.3 Refresh Schedule 4
3.4 Monitoring & Operational Visibility 4
4 Technical Feature Details 4
4.1 Technical Architecture 4
4.2 RBAC & UBAC 4
4.3 Logging & Monitoring 4
4.4 Secrets & Secure Access Management 4
4.4.1 Salesforce Secrets Management 4
4.4.2 Network Rule Configuration 4
4.4.3 External Access Integration 4
4.4.4 Secure Function for Credential Access 4
5 Technical Details - Backend 4
5.1 Database Details 4
5.1.1 Database Details in Snowflake Environment 4
5.2 Schema Details 5
5.2.1 Schema Details in Snowflake Environment 5
5.3 Table Details 6
5.3.1 Tables Details in Snowflake Environment 6
5.4 Objects Details 7
5.4.1 Object Details in Snowflake Environment 7
5.5 Snowflake Container Services (SPCS) Deployment 8
5.5.1 Compute Pool Configuration 8
5.5.2 Snowflake Service Details 9

6 Troubleshooting Scenarios & Recommendations 10
7 Glossary 11
8 Reference Artifacts & Editable Links 12
9 Appendix 13
9.1 Usage Guidelines for Contributors 13
9.2 Environment Deployment Guide 13
9.2.1 Prerequisites 13
9.2.2 Deployment Steps 13
9.2.3 Environment Configuration 13
9.2.4 Validation 13

1 Introduction

This document outlines the requirements, architecture, and implementation details for the **Pi-Ingest** solution. The objective of this initiative is to deliver a secure, scalable, and reliable data ingestion framework—**distributed as a Snowflake Native App**—that enables seamless extraction of data from Salesforce into the consumer's Snowflake account for downstream analytics and operational monitoring.

The document serves as a single source of truth for all stakeholders involved in the project, including business users, data engineers, solution architects, testers, and project managers. It captures both functional and non-functional requirements, along with detailed backend and frontend implementation approaches within the **Snowflake Native App Framework**.

The Pi-Ingest solution leverages the **Snowflake Native App distribution model** and high-performance capabilities such as **Snowpark Container Services (SPCS)**, Snowflake Image Repositories, and **Reference Binding** for secure secrets management and external access. This approach ensures that all logic is executed within an isolated, secure environment in the consumer's account, maintaining strict data governance and operational efficiency.

This initiative is part of the broader **Pi-Snow** roadmap and is designed to improve data ingestion reliability, observability, and scalability through a packaged application model, supporting current and future reporting and analytics needs while minimizing manual configuration for the end-user.

2 Key Considerations

This section outlines the foundational elements that influence the scope, planning, and execution of the Pi-Ingest project, including scope boundaries, assumptions, dependencies, and risks.

2.1 Scope

The scope of this project includes the following:

- Data ingestion from Salesforce into Snowflake
- Incremental and full data extraction using Salesforce APIs
- Data loading into raw and curated Snowflake layers
- Execution of ingestion and transformation jobs using Snowflake Container Services
- Centralized logging of ingestion and transformation job execution

2.2 Out of Scope

The following items are not included in the current scope:

- Data ingestion from sources other than Salesforce
- Advanced analytics, reporting, or dashboarding for business insights

2.3 Assumptions

Sr. No	Description	Implications, if not met
1.	Active Salesforce account with required API access	Unable to extract Salesforce data
2.	Snowflake account with Enterprise edition and SPCS enabled	Application cannot be deployed or executed
3.	Required Snowflake roles and permissions are available	Objects and services cannot be created
4.	Snowflake Network access to Salesforce endpoints is allowed Account (Enterprise Paid Edition)	External connectivity will fail

2.4 Dependencies

Sr. No	Description	Owner	Implications, if not met
1.	Salesforce API credentials and connected app	Salesforce Admin Team	Data extraction cannot be initiated
2.	**Application installation and granting of account-level privileges**	**Consumer AccountAdmin**	App cannot provision infrastructure or execute jobs
3.	**Security approval and Reference Binding** for Secrets and External Access	**Consumer AccountAdmin**	Application is blocked from connecting to Salesforce
4.	Compute pool and warehouse availability	**Consumer AccountAdmin**	Job execution delays or failures

2.5 Risks

Sr. No	Description	Mitigation Strategy
1.	Salesforce API rate limits	Implement incremental loads and retry logic
2.	**Credential exposure or misconfiguration**	**Use Snowflake Secrets with Reference Binding** to ensure credentials remain encrypted and never leave the consumer account boundary
3.	**Consumer region incompatibility for SPCS**	Validate Snowpark Container Service availability in the target consumer region prior to app installation

3 Business Feature Details

This section outlines the key business functionalities delivered by the Pi-Ingest solution. Each feature is aligned with business objectives to ensure reliable, secure, and timely availability of Salesforce data for downstream analytics and operational monitoring.

3.1 Business Process Flow

The Pi-Ingest solution enables end-to-end movement of data from Salesforce into Snowflake. Salesforce data is securely extracted using standard APIs and ingested into Snowflake. The data is validated, transformed as required, and stored in structured layers to support analytics and reporting.

Processed data and operational metadata are made available for downstream consumption and monitoring, ensuring transparency and reliability of data ingestion processes.

3.2 RBAC & UBAC

Role Based Access Control (RBAC) and User Based Access Control (UBAC) are implemented to ensure secure and governed access to data within Snowflake.

RBAC

- Access is granted based on predefined roles (e.g., Data Engineer, Data Analyst, Read-Only User).
- Roles control access to databases, schemas, tables, views, and compute resources.

UBAC

- User access is mapped to roles based on business needs.
- Access is provisioned following the principle of least privilege.

3.3 Refresh Schedule

- Full Load: Initial one-time historical data load
- Incremental Load: Scheduled periodic refreshes based on last modified/Created date
- Refresh Frequency: Daily / Hourly (configurable based on business requirements)
- Failure Handling: Failed loads will be logged

3.4 Monitoring & Operational Visibility

The solution includes monitoring and logging capabilities to provide visibility into ingestion and transformation activities.

- Execution status of ingestion and transformation jobs is captured centrally.
- Success and failure details are logged for audit and troubleshooting.
- Operational metrics such as load duration and record counts are tracked.
- Consumers can configure Snowflake Alerts on top of the logging tables to enable timely resolution of critical ingestion failures.

4 Technical Feature Details

This section outlines the technical interpretation of the business features. It translates business needs into technical components, ensuring alignment between business goals and system capabilities within the Snowflake Native App Framework. It includes data modeling, transformation logic, and the integration of Snowflake-native compute resources.

4.1 Technical Architecture

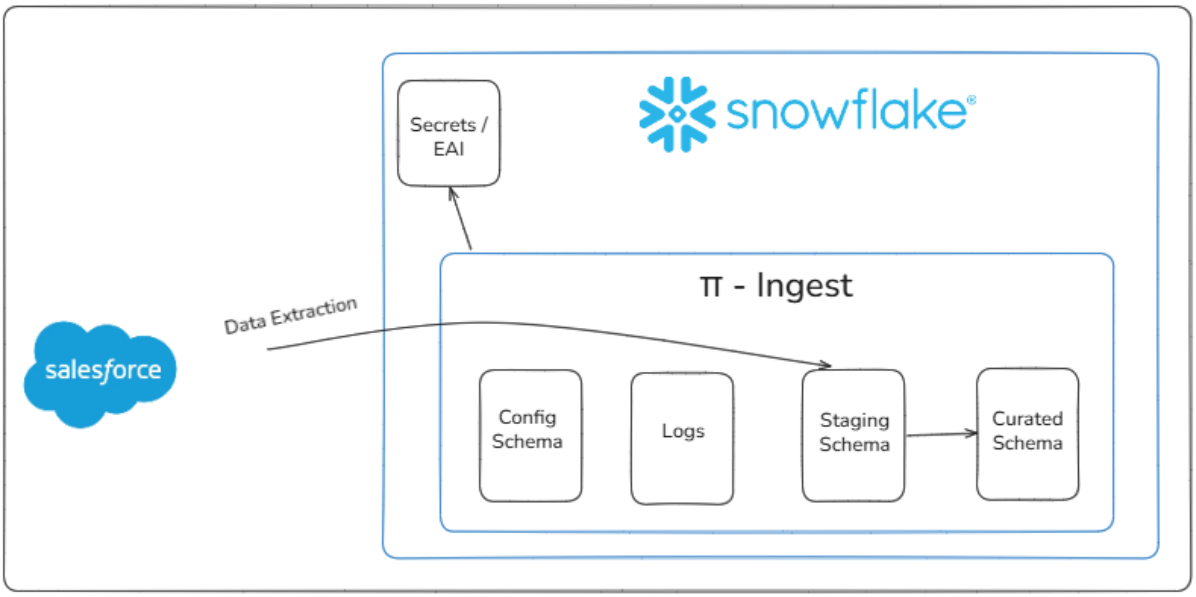

Fig. Pi-Ingest Architecture

The solution follows a Snowflake Native App architecture, where all application logic, container images, and database objects are bundled into an Application Package and installed directly into the consumer's Snowflake account.

- Salesforce Connectivity: Acts as the source system; data is extracted via APIs (Bulk/REST) using Snowflake External Access Integrations (EAI).
- Execution Engine: Ingestion and transformation logic are executed using Snowpark Container Services (SPCS), providing a high-performance, isolated environment for Python-based processing.
- Secure Credential Management: Salesforce credentials are stored as Snowflake Secrets and linked to the application via Reference Binding, ensuring they never leave the consumer's security perimeter.
- Data Tiering: Extracted data is landed into an internal Landing Schema before transformation logic is applied to move data into final curated schemas within the application instance.
- Automated Provisioning: Ingestion jobs are orchestrated using Snowflake Tasks to trigger the SPCS service on a configurable schedule.

4.2 RBAC & UBAC

Access control is implemented using Application Roles defined within the Native App, which are then mapped to the consumer's account-level roles to ensure governed access.

- Application Roles: Defines internal roles (e.g., APP_ADMIN) that control access to the application's internal schemas, stored procedures, and compute pools.
- Consumer Mapping: The consumer account administrator maps these Application Roles to their own internal business roles (Data Engineer, Analyst) using Snowflake's RBAC.
- Data Isolation: Sensitive Salesforce objects are stored in restricted schemas, ensuring that only authorized roles can view raw or curated datasets.

4.3 Logging & Monitoring

Logging and monitoring are implemented to track data ingestion, health and operational issues.

- Audit Logging: Ingestion job execution status and container logs are captured in the LOGS_SF_INGEST schema.
- Data Reconciliation: Record counts and load timestamps are captured for every run to ensure data parity between Salesforce and Snowflake.
- Error Handling: Detailed error logs are generated for failed API calls or container crashes, providing the consumer with self-service troubleshooting capabilities.

4.4 Secrets & Secure Access Management

This solution implements secure credential management and controls external connectivity to Salesforce without exposing sensitive information in the application code or to the application provider.

- Consumer-Owned Secrets: Salesforce authentication details are stored as Snowflake Secrets within the consumer's account. These are linked to the application via Reference Binding, ensuring the provider never has access to the raw credentials.
- External Access Integration (EAI): Outbound connectivity to Salesforce APIs is managed through an EAI reference. This allows the consumer to define and audit the specific network rules and domains the application is permitted to access.
- Secure Reference at Runtime: Internal Secure Functions are utilized to programmatically access the bound secrets only at the moment of execution, maintaining a high level of security isolation.
- Zero-Hardcoding Policy: No credentials, tokens, or sensitive endpoint configurations are hardcoded within the application package or its metadata files, meeting Snowflake's strict security review requirements.

- RBAC-Governed Security: Access to the required secrets and network integrations is governed through the application's internal roles, which the consumer maps to their account-level RBAC for centralized control.

4.4.1 Salesforce Secrets Management

Salesforce authentication credentials are securely managed using Snowflake Secrets within the consumer's account and are accessed by the application via Reference Binding.

- Secret Type: Salesforce credentials are stored as Snowflake GENERIC_STRING secrets.
- Payload: Secrets include the username, password (including security token), client_id, and client_secret.
- Provider Isolation: Secrets are bound to the application through references at runtime; the application logic uses them without ever exposing the raw values to the application provider.
- Access Control: Access to create and manage these secrets is restricted using the consumer's Snowflake RBAC.

4.4.2 Network Rule Configuration

Network rules are defined by the consumer to control outbound connectivity from the application instance to Salesforce endpoints.

- Egress Control: Egress network rules allow outbound access only to approved Salesforce domains as specified in the application manifest.
- Whitelisting: Salesforce login (login.salesforce.com) and specific instance URLs (e.g., *.my.salesforce.com) are explicitly whitelisted to prevent unauthorized data egress.
- Granular Security: Network rules ensure that the application's container services can only communicate with verified API endpoints.

4.4.3 External Access Integration

The External Access Integration acts as the secure bridge enabling outbound communication from the application's internal services to Salesforce APIs.

- Reference Binding: The application requests access to a consumer-defined EAI through Reference Binding, allowing the consumer to audit and approve connectivity.
- Security Association: The EAI explicitly links the approved Network Rules and Salesforce Secrets into a single security context.
- Restricted Execution: Only the authorized internal services and secure functions within the application instance are permitted to utilize the integration.
- Governance: The integration is enabled and governed by the consumer's RBAC, providing a "kill-switch" capability to the consumer at the account level.

4.4.4 Secure Function for Credential Access

Secure functions are used to retrieve Salesforce access tokens at runtime.

- Secure functions reference Snowflake Secrets internally.
- External Access Integration is used to allow API calls within the function.
- The function returns access tokens without exposing credentials.
- Only authorized roles can execute the secure function.

5 Technical Details - Backend

The secure, versioned instance of the Pi-Ingest app installed in the consumer account. It encapsulates all internal schemas used to store configurations, operational logs, raw landing data, and curated Salesforce datasets.

5.1 Database Details

Below is the list of technical objects used for this module:

5.1.1 Database Details in Snowflake Environment

Sr. No	Object Name	Object Type	Description
1.	PI_INGEST_APP	APPLICATION_DATABASE	The secure, versioned instance of the Pi-Ingest app installed in the consumer account. It encapsulates all internal schemas used to store configurations, operational logs, raw landing data, and curated Salesforce datasets.

5.2 Schema Details

Below is the list of technical objects used for this module:

5.2.1 Schema Details in Snowflake Environment

Sr. No	Layer	Object Name	Object Type	Description
1.	Configuration	CONFIG_SF_INGEST	SCHEMA	Internal Application Schema used to store ingestion job metadata, control tables, and configuration files for the Pi-Ingest framework.
2.	Logs	LOGS_SF_INGEST	SCHEMA	Internal Application Schema that captures audit trails, error logs, and operational telemetry for all ingestion and transformation activities
3.	Raw	LANDING_SF_INGEST	SCHEMA	Internal Staging Schema where raw data extracted from the Salesforce APIs is initially landed before processing.
4.	Curated	SF_INGEST	SCHEMA	Target Analytics Schema containing the final transformed, cleansed, and

				curated Salesforce datasets ready for business consumption.

5.3 Table Details

Below is the list of technical objects used for this module:

5.3.1 Tables Details in Snowflake Environment

Sr. No	Layer	Object Name	Object Type	Description
1.	Configuration	CONFIG_SF_INGEST.JOB_CONFIGS	TABLE	Internal Metadata Table containing ingestion job parameters, refresh schedules, and Salesforce API configuration details.
2.	Logs	LOGS_SF_INGEST.AUDIT_LOG	TABLE	Operational Telemetry Table tracking every execution status, run IDs, record counts, and ingestion timestamps.
3.	Logs	LOGS_SF_INGEST.ERROR_LOG	TABLE	Exception Handling Table capturing detailed error messages, API failure codes, and stack traces for troubleshooting
4.	Raw	SALESFORCE_RAW_DB. LANDING_SF_INGEST.ACCOUNT	TABLE	Internal Staging Table containing the raw, unmodified data extracted from the Salesforce Account object via Bulk/REST APIs.
5.	Curated	SALESFORCE_RAW_DB.SF_INGEST.ACCOUNT	TABLE	Final Presentation Table containing cleaned, de-duplicated, and structured Account data ready for downstream analytics.

5.4 Objects Details

Below is the list of technical objects used for this module:

5.4.1 Object Details in Snowflake Environment

Sr. No	Object Name	Object Type	Description
1.	CONFIG_SF_INGEST.CONFIG_STAGE	STAGE	Internal Application Stage where the consumer uploads the ingestion metadata file (sf_ingest_config.csv) to define extraction logic.
2.	CONFIG_SF_INGEST.SF_GET_ACCESS_TOKEN	FUNCTION	Secure UDF that utilizes Reference Binding to safely retrieve OAuth tokens from Salesforce using consumer-owned secrets
3.	CONFIG_SF_INGEST.FF_SF_CSV_AUTO	FILE FORMAT	Standardized file format used by the application to parse configuration files and landed CSV data from Salesforce
4.	SRV_SF_INGEST_DAILY	SERVICE	Snowpark Container Service (SPCS) instance that executes the core Pi-Ingest Python logic within an isolated compute environment.

5.5 Snowflake Container Services (SPCS) Deployment

This section describes the backend deployment of the application using Snowpark Container Services within the Snowflake Native App Framework.

- Packaging: The ingestion logic is containerized using Docker and bundled into the Application Package.
- Image Security: Docker images are stored in an internal Application Image Repository, which is scanned by Snowflake for security vulnerabilities prior to app distribution.
- Service Instantiation: Snowflake Services are created within the Application Instance in the consumer account to run the containerized logic.
- Isolated Compute: The service executes ingestion and transformation jobs within an isolated, Snowflake-managed environment, ensuring the consumer's data never leaves their Snowflake security boundary.
- Reference-Based Configuration: Resource requirements and environment-specific configurations are managed through the application's manifest and service specifications.

5.5.1 Compute Pool Configuration

The application is executed using a Snowflake-managed compute pool to support containerized workloads.

- A dedicated compute pool is created for Pi-Ingest execution.
- The compute pool is configured with fixed minimum and maximum nodes to control resource usage.

Sr. No	Object Name	Object Type	Description
1	PI_INGEST_COMPUTE_POOL	COMPUTE POOL	An account-level compute resource provisioned by the consumer and bound to the application to provide the physical infrastructure for the ingestion service.

5.5.2 Snowflake Service Details

The ingestion logic is executed as a Snowpark Container Service (SPCS) within the secure boundary of the Application Instance.

- Internal Service: Internal Snowflake services are instantiated within the application instance to run the containerized Pi-Ingest logic.
- Infrastructure Binding: The services execute on the consumer-provisioned compute pool, which is linked to the application via Reference Binding.
- Secure Egress: Salesforce connectivity is enabled by binding a consumer-defined External Access Integration (EAI) to the service at runtime.
- Lifecycle Management: Service controls (Start, Suspend, Resume) are managed through the application's secure stored procedures, allowing for automated orchestration and centralized logging.

Sr. No	Service Name	Description
1	SRV_SF_INGEST_DAILY	Executes the primary daily batch ingestion and transformation jobs within the application container
2	SRV_SF_INGEST_DAILY_NRT	Executes near-real-time (NRT) ingestion and transformation jobs on an hourly schedule for high-frequency data syncs.

6 Troubleshooting Scenarios & Recommendations

This section outlines common technical issues that may arise during system operation along with recommended steps for diagnosis and resolution.

Scenario	Description	Possible Causes	Recommended Actions
Ingestion failure	Ingestion job does not complete	Schema changes in Salesforce, data type mismatch	Review error logs, update ingestion logic
Connectivity Error	Application fails to authenticate or connect to Salesforce APIs	Reference binding for Secret or EAI is missing/invalid; Credentials expired	Verify and re-bind application references in Settings; Validate Salesforce Secret values

7 Glossary

This section defines key terms, acronyms, and technical language used throughout the document. It ensures a common understanding among all contributors and stakeholders.

Term	Definition
Application Instance	The deployed manifestation of the application within the Consumer's Snowflake account, encapsulating all data, logic, and metadata.
Consumer Account	The Snowflake account where the Pi-Ingest application is installed, configured, and executed.
DAG	Directed Acyclic Graph – A mathematical representation of the workflow orchestration logic, defining the sequence and dependencies of ingestion tasks.
EAI	External Access Integration – A Snowflake security object that allows the application to securely communicate with external Salesforce API endpoints.

8 Reference Artifacts & Editable Links

This section provides direct access to editable project artifacts such as process diagrams, data mappings, and technical documentation. These resources are maintained in shared locations to ensure version control and collaborative updates.

Asset Type	Description	Official Doc Link
Snowflake Platform Documentation	Official Snowflake documentation portal for SQL, architecture, security, administration, and advanced features	Snowflake Documentation Home
Getting Started	Guide for onboarding and initial setup with Snowflake	Getting Started with Snowflake
User Guides	Deep documentation covering Snowflake usage, warehouses, tables, and data loading	Snowflake User Guide & Tasks
Architecture & Core Concepts	Explains Snowflake's distributed architecture and core platform design	Snowflake Key Concepts & Architecture
Integration / Connectors	Official docs related to connectors, drivers, and integrating external systems	Snowflake Integration & Connectors Guide
ETL / Data Loading Patterns	Guidance on Snowpipe, bulk loads, and third-party integration methods	Snowflake Data Integration Guide
Snowflake Container Services (SPCS)	Overview and concepts for running containerized workloads in Snowflake	https://docs.snowflake.com/en/developer-guide/snowflake-container-services/overview
Snowflake Services	Documentation for creating, managing, and operating Snowflake services	https://docs.snowflake.com/en/developer-guide/snowflake-container-services/services
Compute Pools	Guide for configuring and managing compute pools for container execution	https://docs.snowflake.com/en/developer-guide/snowflake-container-services/compute-pools
Image Repository	Documentation for storing and managing Docker images in Snowflake	https://docs.snowflake.com/en/developer-guide/snowflake-container-services/image-repository

External Access Integration	Enabling secure outbound network access from Snowflake	https://docs.snowflake.com/en/sql-reference/sql/create-external-access-integration
Network Rules	Defining egress network rules for external connectivity	https://docs.snowflake.com/en/sql-reference/sql/create-network-rule
Snowflake Secrets	Secure storage and usage of credentials in Snowflake	https://docs.snowflake.com/en/sql-reference/sql/create-secret
Secure Functions	Creating secure functions to protect sensitive logic and credentials	https://docs.snowflake.com/en/sql-reference/sql/create-function

9 Appendix

This section includes supporting information, usage guidelines, and references.

9.1 Usage Guidelines for Contributors

The following guidelines should be followed by all contributors working on the Application Package:

- Follow established naming conventions for Snowflake objects, application roles, and manifest-defined references.
- Ensure all ingestion and transformation jobs write detailed execution telemetry to the internal LOGS_SF_INGEST schema.
- All container images must undergo a mandatory security scan and patch update before a new Application Version or Patch is published.
- Do not modify the setup.sql or manifest.yml files without an approved change request and versioning plan.

9.2 Environment Deployment Guide

This section provides guidance for installing and configuring the Pi-Ingest application instance within a consumer environment.

9.2.1 Prerequisites

- Snowflake Account: Enterprise Edition (or higher) with Snowpark Container Services (SPCS) enabled in the target region.
- Administrative Privileges: The installing user must have roles with ACCOUNTADMIN or the ability to grant CREATE COMPUTE POOL and BIND STATE to the application.
- Security Objects: Salesforce Connected App credentials must be stored as a Snowflake Secret (GENERIC_STRING) within the consumer account.
- Network Governance: Pre-defined Network Rules and an External Access Integration (EAI) must be available for Reference Binding to permit Salesforce API egress.

9.2.2 Deployment Steps

The deployment of a Snowflake Native App follows a provider-consumer model. The following steps outline the end-to-end process:

- Image Provisioning (Provider): Build and push the Pi-Ingest Docker images to the Image Repository within the Application Package.
- Version Creation (Provider): Define the application logic in the setup.sql and manifest.yml files, then create a new Application Version.
- App Installation (Consumer): Install the Application Instance from the Snowflake Marketplace or a Private Listing.
- Reference Binding (Consumer): Securely bind account-level objects (Salesforce Secrets and External Access Integrations) to the application's required references via the Settings UI or SQL.
- Privilege Granting (Consumer): Grant the application the CREATE COMPUTE POOL and BIND STATE privileges to allow the app to manage its own infrastructure.

- App Initialization (Consumer): Execute the internal application procedures (e.g., CALL CREATE_SF_OBJECTS()) to instantiate the secure functions and services.

9.2.3 Environment Configuration

Configuration in the Native App model is governed by isolated application roles and reference associations:

- Service Isolation: Separate Snowpark Container Services are instantiated for different ingestion schedules (Daily vs. NRT).
- Metadata Management: Run-specific parameters are configured within the application's internal metadata tables via the CONFIG_STAGE.
- Role Mapping: Access control is enforced by mapping Application Roles (e.g., APP_ADMIN) to the consumer's existing Snowflake RBAC structure.

9.2.4 Validation

Once the installation is complete, the following checks ensure the application is operational:

- Reference Verification: Confirm that all required references (Secrets, EAI, Compute Pool) show a status of 'ASSOCIATED'.
- Service Status: Verify that the Snowpark Container Service status is RUNNING using the application's monitoring procedures.
- Ingestion Health: Confirm successful data extraction and transformation of job execution by reviewing the AUDIT_LOG table.
- Log Validation: Ensure that container logs and error entries are being correctly populated in the LOGS_SF_INGEST schema.
- Availability: Confirm that all application objects (Tables, Views, and Procedures) are visible and accessible to authorized consumer roles.

10 Support & Contact Information

10.1 Technical Support

For issues related to ingestion failures, connectivity troubleshooting, or bug reporting, the Pi-Snow Support Team is available to assist:

- Email: Admin@Pibythree.com, Alliance@Pibythree.com

Note: As this is a Snowflake Native App, please ensure you have checked the internal LOGS_SF_INGEST.ERROR_LOG table before reaching out, as this will help our team expedite your request.

10.2 Business & Sales Inquiries

For inquiries regarding the broader Pi-Snow roadmap, licensing for additional Salesforce objects, or custom integration needs:

- Contact: Alliance@pibythree.com
- Website: www.pibythree.com

10.3 Documentation & Feedback

To access the latest version of this document or to provide feedback on the application's performance:

- Documentation Portal:Pi Accelerators by PibyThree: Boost Cloud & AI Growth.