[EXTERNAL] Snowflake Marketplace Provider Guide: Native App Approval Process

Last updated: Nov 5, 2025

This guide is a step-by-step manual for successfully developing, testing, and submitting a Snowflake Native Application for approval on the Marketplace.

Table of contents

Phase 1: Design and develop

Before you begin developing your application, it is important to understand the guidelines and standards for Native Apps on Snowflake Marketplace. All apps are evaluated against the **Enforced Requirements**.

Please take special note to the following:

1. **Application setup:** Majority of the application setup and configuration must be executed on Snowflake. Applications that require consumers to go off Snowflake's platform for the majority of the application configuration are not allowed.
2. **Core functionality**: Apps must deliver their core product experience on Snowflake. The core experience must remain on the Snowflake Native app.
3. **Authentication method:** Applications that require consumer user authentication should follow the standards below:
    - **Username and password:** Snowflake has deprecated the usage of username and password.
    - **Key-pair authentication:** If you are requesting the consumer to generate the public and private key and share the private key externally, **this is not allowed**. If you are intending to use key-pair, providers should generate the key-pair for the consumer and share the public key with the consumer to alter the user.

i. Other acceptable authentication methods for uploading credentials into Snowflake from an external System include PAT tokens and oAuth.

Your First Steps: Creating the App Package

To start building, reference the official documentation:

- Native Application Quickstart Guide
- Native Applications Overview
- Getting started with Native Applications
- Github Native Application templates
- Step by step guide for submitting your application on to the Marketplace (Medium post)

Phase 2: Test your application

This is the most common failure point for providers. An app that fails to install or runs into immediate errors during our functional review will be rejected. You must test your app from a ***consumer's point of view*** before submitting.

How to test your application

After developing your app, share the application package with a **separate test account** (not your provider development account) to simulate the consumer experience. Ensure that there are no replication and listing auto-fulfillment errors when sharing your app to external consumers.

Test these scenarios:

	Internal Organization	**External Organization**
Same region	☐	☐
Different region	☐	☐

Note that if you share your app with a trial account, current limitations for trial accounts may prevent you from testing the full consumer experience.

Common Errors to Look For

Use this checklist to catch common problems:

- ☐ **Installation Failures:** Check for any installation errors. A consumer must be able to install the app without error (e.g., no setup script errors or initial privilege issues).

- [ ] **README & Configuration:** Ensure your `readme.md` file is complete and clear. It **must** contain exact steps for what a consumer should do immediately after installation to configure and use the app.
- [ ] **Credential Handling:** If your app requires credentials, confirm that your README provides a clear, step-by-step process for a consumer to get and use them.
    - **Please note:** For functional reviews, our team requires that you provide test credentials and any sample data required for testing.
- [ ] **Privilege errors:** Does the app use all the privileges requested from the consumer? Does the app fail because it's trying to perform an action it didn't request privileges for?
- [ ] **Permissions SDK:** If your application requests account level privileges or object level privileges from the consumer, are you using the Permissions SDK package to request these privileges via Streamlit UI?
    - Request references and object-level privileges from consumers
    - Request access to objects and privileges in a consumer account

---

Phase 3: Pass the Automated Security Review

Before an app can be submitted for functional review, it must pass the **security review.**

1. **Review Security Best Practices:** Ensure your app conforms to all security requirements.
    a. Native app: Security requirements and guidelines for a Snowflake NA
    b. NA+SPCS: Secure a Snowflake NA with SPCS
2. **Trigger the Automated Security Review:** The scan starts automatically when you do one of the following for an app package:

```
SQL
ALTER APPLICATION PACKAGE my_app_pkg set DISTRIBUTION = EXTERNAL;
```

    a. **Add a new version or patch:** The new version is scanned automatically.
    b. **Set `DISTRIBUTION` to `EXTERNAL`:** The 10 most recent versions are scanned. All patches for these versions are also scanned.
3. **Check the scan results:** Wait for the scan to complete. You can check the status using:
    a. **Snowsight UI:** You can navigate to **Projects** > **App Packages** > Select the **application** > Go to the **Security Scan Status** column
    b. **Worksheet:** Run the below query in a worksheet:

```sql
SHOW VERSIONS IN APPLICATION PACKAGE my_app_pkg;
```

Look at the `review_status` column.

- **If `REJECTED`:** indicates that the automated security scan completed, but the application package was not approved.
    - When an automated security scan fails, the Snowflake will conduct a **manual review** on the application package.

Phase 4: Submit Your App for Functional Review

Once your app is developed, tested, and has passed the security review, you're ready to submit it to the Marketplace Operations team for a functional review.

1. **Create Your Listing:** In Snowsight, go to the '**Data Sharing**' > '**Provider Studio**' > Select the '+ Create listing' button.
2. **Attach Your Application Package:** Under '**Add data product**', attach the application package and version that passed the security scan to your listing.
3. **Submit for Review:** Complete the additional fields for your listing. Once you've fully configured your listing, select the '**Submit for approval**' button and our MPOps team will begin the process for conducting the functional review of your application.

Phase 5: The Functional Review Process

This is the final review by the Marketplace Operations team. The goal is to ensure your app provides a secure, functional, and high-quality experience for consumers.

What Happens Next: The Review Workflow

Here is what you can expect after you click '**Submit for approval**':

1. **Assignment:** Your listing will be assigned to a reviewer within the Marketplace Ops team within 24 hours.
2. **Review:** The reviewer will install, configure, and test your application in a reviewer account from the perspective of a new consumer.
3. **Communication** (via support case (snowflake@support.com))**:**

    - **If we have questions or find minor issues:** We will reach out to you via email to the contacts outlined in your Marketplace Profile.
        i. *Please ensure these emails are up to date and accurate.*
4. **Decision:**
    - **Approved:** Your listing is approved, and you will be notified by email. You can then publish the listing to the Marketplace.
    - **Rejected:** You will receive an email detailing **all required changes**. This may include feedback on functionality, security, or listing metadata. You must fix all issues and resubmit a new version for review.

Note that functional reviews may take up to 2 weeks depending on the updates required. Please follow the testing steps outlined in Phase 2 in order to ensure a quicker and smoother review process.

FAQ

Design and develop:

1. ***What does "core functionality must be on Snowflake" really mean? Can my app call my external API?***
   Yes, your app can and often should call external APIs. The "core functionality" rule means the consumer's primary experience and interaction - the user interface (UI), the configuration, and the execution of logic - must happen inside Snowflake.
    - **Allowed**: A Streamlit UI inside Snowflake that calls your company's API to get a prediction and then writes that prediction to a Snowflake table.
    - **Not Allowed**: An "app" that is just a link telling the consumer to "Go to mycompany.com/configure" to set up the integration and use the platform. If configuration is happening off-platform, the majority of the application logic must exist on Snowflake.
2. ***For manifest v1: When am I required to use the Permissions SDK?***
   You must use the Permissions SDK if your application uses a Streamlit and needs to:
    - Request any account-level privileges (e.g., CREATE WAREHOUSE).
    - Request object-level privileges on a consumer's existing objects (e.g., SELECT access on a table they own).
3. ***What are the key differences I should know between manifest_version: 1 and manifest_version: 2?***
   The primary difference between the two manifests is how your application requests and handles privileges from the consumer.
    - **Manifest_version 1:** This version requires additional work for the consumer. As a developer, to utilize account-level / object-level privileges in consumer's account, your application must:
        1. Request account-level or object-level privileges from the consumer,
        2. Consumer will grant the privileges through the application's UI (via Python Permissions SDK), and

3. Application is now able to utilize the account-level / object-level privileges.
    - **Manifest_version 2:** This version introduces automated granting of privileges. As a developer, you are now required to explicitly declare every privilege your application requires directly within the privileges section of the manifest.yml file. There is no need for consumers to go into the UI and grant those privileges. Instead, the application presents a list of all requested privileges to the consumer for approval upon installation or upgrade.
        1. See documentation here.
4. ***What new features are enabled by app specs in manifest_version: 2?***
   Application specifications (App specs) allow consumers to review and approve or decline requests for the following actions:
    - **External access integration:** Allow secure access to external network endpoints within a user-defined function or stored procedure. External access integrations use network rules to restrict access to specific external network locations. For additional information, see here.
    - **Security integration:** Allow secure access to third-party authentication providers such as OAuth. Security integrations provide secure authentication and access control. For additional information, see here.
    - **Data sharing with other Snowflake accounts (Shares and listings):** Allow apps to share data back to providers or third-party Snowflake accounts. Shares contain database objects to be shared, and listings provide the mechanism to share data across accounts and regions. For additional information, see here.
5. ***How are app specs in manifest_version: 2 different from manifest_version 1?***
   When using automated granting of privileges, an app has the required privileges to create these objects when running the setup script. However, because these objects enable external connections or data sharing, consumers must approve these operations when configuring the app.
6. ***What are the benefits of using manifest_version: 2 with app specs?***
   Using automated granting of privileges with app specifications has the following benefits:
    - Consumers do not have to manually create integrations, shares, or listings required by the app and approve access to them using references.
    - Providers do not have to write code that checks for the existence of the required privileges and objects during installation or upgrade.
    - Consumers have clear visibility and control over external connections and data sharing requests.
7. ***My application needs a warehouse but I am unsure if I should have the application create the warehouse or the consumer grant privileges to their warehouse to the application. What are some things to consider?***

Consideration	**Application creates own warehouse**	**Consumer grants their warehouse to the application**
Installation	Using manifest_version: 2, the consumer simply approves the CREATE	The consumer must find a warehouse and run GRANT USAGE... commands or grant usage on

	WAREHOUSE privilege once during installation.	warehouse via UI (permissions SDK).
Performance	As the provider, you can control the default warehouse size (e.g., X-SMALL) and settings to guarantee a baseline performance.	The app's performance depends on a shared warehouse that Providers don't control. It could be overloaded or undersized, leading to slow performance.
Workload	Application queries do not interfere with the consumer's other work.	This can cause some 'noise' for both the application and the consumer as it is difficult to attribute where the consumption is coming from.
Cost	Consumers will be able to attribute the cost specifically to the application by monitoring the dedicated warehouse.	Consumers will not be able to directly attribute cost to the application within their existing warehouse spend.

Test your application:

1. ***My app works perfectly on my provider account. Isn't that enough?***
   No. This is the single most common reason for rejection. Your provider account has owner-level privileges on the application package, which masks many permission and dependency errors. You **must** test from a separate consumer account to experience what a real customer will.
2. ***What is the best way to test to avoid a functional review failure?***
   Create a brand new Snowflake account in a different region from your provider account. Do not use an account that is in the same organization. This setup most accurately simulates an external consumer and is the best way to find installation errors, privilege issues, and cross-region failures.

Security review:

1. ***I'm a Snowpark Container Services (SPCS) provider and my app failed with the following error when I set distribution to EXTERNAL. What do I do?***

```
None
Error Code: 093197 Account is not allowed to create application package
versions or patches with
Snowpark Container Services for EXTERNAL distribution
```

   - This error means your account has not yet been approved to publish an app with containers. If you see this error, submit a security questionnaire to begin the approval process.

2. ***My security review and manual review were both `REJECTED`. What do I do in terms of next steps?***
   If the manual review is **REJECTED**, you can appeal the rejection by opening a **severity 4 support ticket**. When appealing a CVE-based rejection, providers must submit detailed documentation explaining the following things:
   - Why the CVE is not exploitable in the application

- Reachability analysis report, if available
- A plan for updating to the fixed version
- If there are no plans for an update, a detailed explanation of why a vulnerable version cannot be updated

Submitting your listing with application:

1. ***What test credentials or sample data exactly do I need to provide?***
   You must provide **everything** our team needs to use the app as a new consumer, end-to-end. Failure to do this can cause an immediate rejection. For example:
   - **Test credentials:** A non-expiring API key, username/password, or any other credentials.
   - **Sample data:** If the app doesn't generate its own data, provide sample data we can use for testing.
   - **Configuration values:** Any specific values needed during setup (e.g., "Account ID," "External URL," etc.).
   - **A test step by step:** A simple step-by-step list of "what to do" to test the main feature.

Functional review:

1. ***I'm going through the functional review. I've made the updates to the application. What are the next steps?***
   Once you've addressed the feedback from our team, please reach back out to the case our team used for communication and let us know that you've pushed the changes. Our team will then go in and test the application to make sure that it is fully functional and working. Our team will also let you know the next steps if your application has been approved.
2. ***My app was rejected. Do I have to go through the entire process (including the security scan) again?***
   Yes. A rejection requires you to fix the issues, create a new version or patch of your application package, and resubmit.
   1. This new version **will trigger a new Automated Security Review.**
   2. Once it passes the security scan, you must update your release directive to point to the new and approved version or patch and resubmit the listing. The MPOps team will then re-test your application, focusing on the feedback provided previously.
3. ***How long does the functional review last?***
   Maximum 2 weeks
4. ***It's been over 24 hours and no one has been assigned to my review. Who do I contact?***
   You will be assigned a reviewer from the MPOps team within 24 hours after listing submission. You will only hear from the reviewer if they have a question or encounter an error. Please do not open a support ticket to ask for a status update unless the 2-week review window has passed.