Partner Integration Guide for Databricks AI/BI (Dashboards + Genie)

Author(s)	**Prasad Kona** **Maria Pere-Perez**
Reviewers	Ken Wong Miranda Luna
Creation date	Mar 24, 2025
Last updated	Jun 17, 2025

Overview

What if your users could ask questions about their data — and get instant answers, with full context and governance? That's the power of Databricks AI/BI.

This guide helps you bring that power directly into your product. You'll learn how to build a secure, scalable integration using two key paths: real-time queries with Genie to chat with your data and discover dashboards available for users. Whether you're enabling natural language Q&A or surfacing dashboards through search, we'll show you how to make the experience seamless for your users.

Scope of this Guide

This guide is designed for technology partners integrating with Databricks AI/BI — specifically AI/BI Genie and AI/BI Dashboards. While Databricks supports many products/features for integration this document focuses solely on integrating with AI/BI.

Partner Onboarding Checklist

Task	**Owner**	**Status**
Apply for ISV Partnership, specifically as a Technology Partner.	Partner	☐
Define integration scope: AI/BI (Genie ***and*** Dashboards). See below "Why Metadata Indexing Is Essential".	Partner	☐
Support OAuth for Authentication (U2M for end users, M2M for backend jobs, both)	Partner	☐
Set up Databricks named connector in your system	Partner	☐
Decide metadata indexing method (API polling or Databricks job)	Partner	☐
Allow connector to support different Genie spaces and AI/BI dashboards at runtime	Partner	☐
Deploy Databricks job to extract metadata (if using Option 2 - Databricks job)	Partner	☐
Index AI/BI dashboard metadata (title, queries, datasets, visualization metadata)	Partner	☐
Index Genie metadata (space details, associated datasets)	Partner	☐
Support and test user flows across search and assistant use cases	Partner	☐
Validate authentication and data access enforcement	Partner + Databricks	☐
Submit for Validation	Partner	☐
Finalize joint go-to-market plan	Partner + Databricks	☐

Important: Why metadata indexing around AI/BI Is Essential

Currently metadata around AI/BI Genie and Dashboards is available via rest api's and not available via say a systems table. A system's table might be easier to query from a UI or application.

[If relevant to your product integration] It might be important for you to periodically index the metadata into your own table (in your application) or as a table Databricks that can be queried realtime.

Databricks supports two approaches to indexing the metadata around AI/BI:

- *API* – Invoke Databricks api's. Easier to implement, but at scale,may hit performance or rate limits
- *Databricks jobs* – Recommended for production; extracts metadata and writes to Delta tables for efficient querying

---

Governance & Security

Unity Catalog governs access across both Genie and Dashboards:

- Users only see what they're permitted to
- OAuth and service principals provide secure, auditable access control in partner integrations

---

What You Miss If Only Genie Is Integrated

- ❌ Can't run ad hoc NLP queries against **curated datasets** tied to dashboards
- ❌ Miss out on shared business logic and **trusted KPIs** defined in dashboards
- ❌ Limited metadata — no access to dashboard **titles, owners, or visual configuration**
- ❌ Dashboards provide a curated view of pre-defined metrics. Without them, users **lose the ability to explore insights visually** or drill down beyond what Genie can present in text.

While Genie indirectly provides secure access to chat your dataset, it doesn't expose the full **BI layer** — meaning the dashboards that have visuals, and curated business context.

What You Miss If Only Dashboards Are Integrated

- ❌ No natural language Q&A over governed data
- ❌ No dynamic SQL generation based on user queries
- ❌ No real-time chat or follow-up questions to explore insights further
- ❌ No AI-generated summaries or contextual help to explain data or trends

Dashboards offer **trusted visualizations**, but they're static. Without Genie, partners miss the **interactive, AI-native experience** — something increasingly expected as platforms like Glean as they move toward **agent-based automation** and decision support.

Based on a user query you would want to provide to your end users a link to a AI/BI dashboard.

Integration Details and Technical Guidance

---

1. Integration Architecture

- **Genie:** Enables natural language questions over data, powered by Databricks' LLM and metadata-aware execution.
- **AI/BI Dashboards:** Interactive dashboards created in Databricks for visual insights and monitoring.
- **OAuth + Service Principals:** All integrations should use user-based OAuth U2M for interactive use and OAuth M2M service principals for backend or batch metadata extraction jobs.

---

2. Genie Integration

2.1 Genie :

- Chat with Genie: Most interactions in Genie space happen in a chat window or using the Genie API (Public Preview).
    - In the Genie space UI, each user can access a threaded record of their conversations. Each conversation retains context from previous interactions in that thread, which helps Genie understand follow-up questions and assist users in refining or exploring their results.
    - Integrations with Genie API provide a similar experience to end users.
- Refer to the documentation page for Genie for additional information
    - https://docs.databricks.com/aws/en/genie/

2.2 Required APIs:

Some of the key APIs that would be leveraged while building the integration are as follows:

- GET /api/2.0/genie/spaces/{space_id}
    - Get details of a Genie Space.
    - https://docs.databricks.com/api/workspace/genie/getspace
- GET /api/2.0/permissions/{workspace_object_type}/{workspace_object_id}/permissionLevels
    - Get the permission levels that a user can have on an object.
    - https://docs.databricks.com/api/workspace/workspace/getpermissionlevels
- GET /api/2.0/genie/spaces

    - A "list all spaces" API is available in public preview. It is not listed on the api docs yet but is available for ISV integrations.
- Unity Catalog Metadata
    - Databricks API / sdk /connectors : For tables, schemas, access control.
    - SQL-based access to INFORMATION_SCHEMA and system tables for advanced metadata.

Genie Conversation API's

1. Start Conversation

Details on the API Request and response are available at

- https://docs.databricks.com/api/workspace/genie/createmessage

2. Get conversation message

Details on the API Request and response are available at

- https://docs.databricks.com/api/workspace/genie/getmessage

3. Create conversation message

Details on the API request and response are available at

- https://docs.databricks.com/api/workspace/genie/createmessage

4. Get conversation message SQL query result & Execute SQL query in a conversation message

** Both endpoints respond with the same format*

Details on the API request and response are available at

- https://docs.databricks.com/api/workspace/genie/executemessageattachmentquery
- https://docs.databricks.com/api/workspace/genie/getmessageattachmentqueryresult

5. Refer to the full API Documentation at

- https://docs.databricks.com/api/workspace/genie
- https://docs.databricks.com/gcp/en/genie/conversation-api

- **Most important link has best practices around using conversation api and limitations**

2.4. Genie Conversation API - Deep Dive

How do permissions work with the Genie API?

By default, your Genie space will be using Run as viewer permissions as documented below.

Run as viewer: If you want different users of your Genie application to have different access to data, make sure your Genie space has the "Run as viewer" settings option selected. Each different user must have Databricks workspace access and authentication tokens. Your application must manage these different tokens when their associated users call on the API. Finally, these different users must also have the permissions on the following:

- Genie space: "Can Run" or above permissions
- Attached SQL warehouse: "Can Use" or above permissions on the warehouse
- Included tables: "Select" or above permissions on all of the tables added to the Genie space. This is where you can apply RLS permissions for different users so that they see different data.

If your Genie space is enabled to "Run as viewer," but only one user's authentication token is used by your application, everyone will use that one user's credentials on the data (the same effect as embedding that user's credentials).

NOTE: If you have also enabled the Genie Embedded Credentials private preview, the Genie API can leverage Embedded Credentials. Reach out to your account team if you want this feature and haven't been enabled yet

Embedded credentials: If you want all of your users to have access to the same data, you can configure your Genie space + API to do so (assuming you are in the Embedded Credentials private preview). You can achieve this by selecting the "Embedded Credentials" option on the Genie space. Additionally, you must ensure that the authentication tokens used by your application are associated with users that have Can Run or above permissions on the Genie space.

- Details: Embedded credentials will create one service principal that mirrors the last editor's permission attached tables only. The last editor is the person who most recently edited the data room settings such as changing warehouse, list of attached tables, descriptions, etc. Modifications to Instructions do not make a user the "last editor."

How to use the Conversation APIs

The documentation page provides a deep dive on using the conversation api's

https://docs.databricks.com/aws/en/genie/conversation-api

1. Setup

- Locate the ID for your Genie space. It can be found in the URL
    - https://{PLACEHOLDER}.databricks.com/genie/rooms/01ef274d35a310b5bffd01dadcbaf577
- Ensure your application is configured to use Databricks REST APIs.
    - Databricks authentication information, such as a Databricks personal access token
    - The workspace instance name of your Databricks deployment

```
HOST= "<WORKSPACE_INSTANCE_NAME>"
TOKEN="<your_authentication_token>"
HEADERS = {'Authorization': 'Bearer {}'.format(TOKEN)}
```

2. Ask an initial question + retrieve SQL results

Create a new chat thread with an initial question

A `POST` request to create a new chat thread with an initial question will look as follows:

```
POST /api/2.0/genie/spaces/{space_id}/start-conversation
Authorization: <your_authentication_token>
{
    "content": "What are the biggest open sales opportunities this year?",
}
```

An example API response and detailed descriptions are available in the appendix.

The important fields for next steps are:

- `conversation_id:` unique id for the created conversation thread. This ID is a necessary path parameter for all of the other Genie conversation functions
- `message_id:` unique id of the submitted question. Use this ID to track its execution status and get SQL query results in next steps.

Get status of your submitted question + retrieve generated SQL
Now that you have submitted your question, you can poll the "Get conversation message" endpoint to fetch its execution status.

A `GET` request to fetch status and SQL statement with description (if available) will look as follows:

```
GET /api/2.0/genie/spaces/{space_id}/conversations/{conversation_id}/messages/{message_id}
Authorization: Bearer <your_authentication_token>
```

An example API response and the detailed descriptions are available in the appendix.

The important field for next steps is the `status` field, which reports the status of the execution.

The options below delineate the different statuses and what they mean:

- Generation still in progress
    - `SUBMITTED`: The initial question has been submitted to the LLM.
    - `FILTERING_CONTEXT`: Genie is determining relevant context to be included for the LLM.
    - `FETCHING_METADATA`: The LLM is fetching metadata from the data sources.
    - `ASKING_AI`: Waiting for the LLM to come up with a response based on metadata + question.
    - `PENDING_WAREHOUSE`: Waiting for the space's warehouse to activate before executing the query.
- Executing SQL query
    - `EXECUTING_QUERY`: Genie will respond with a generated SQL statement, but the query is still executing. The SQL statement and its description will be available in the `attachments` field.
- Conclusive status
    - `COMPLETED`: The entire messaging process has completed:
        - Genie can respond with a follow-up question rather than an answer. In these cases, the `attachments` array field will contain a `text` object with the follow-up question in the `content` field.
        - Genie can respond with a generated SQL statement. In these cases, the `attachments` array field will have a `query` object that contains...
            - `title` string: Name of the query
            - `query` string: AI generated SQL query
            - `description` string: Description of the response
            - `last_updated_timestamp` int64: Time when the user updated the query last
    - `FAILED`: Generating a response or executing the query failed. Additional details will be available in the `error` field.
    - `CANCELLED`: Message has been terminated.

            - This condition can be produced when SQL warehouse is stopped, or other external factors.
            - Currently no way to cancel a submitted question via API.
    - `QUERY_RESULT_EXPIRED`: The query results for the message have expired (after 7 days) and the user needs to execute the query again.

Fetch query results
Once the "Get conversation message" endpoint shows a `status` field that is "COMPLETED", you can retrieve the actual query result from the "Get conversation message SQL query result" endpoint.

A `GET` request to fetch SQL results (if available) will look as follows:

```
GET
/api/2.0/genie/spaces/{space_id}/conversations/{conversation_id}/messages/{message_id}/quer
y-result/{attachment_id}
Authorization: Bearer <your_authentication_token>
```

An example API response and the detailed descriptions are available in the appendix

The important fields to note are:

- `status:` object describing various aspects of SQL statement execution
    - `state:` the statement execution state:
        - PENDING: waiting for warehouse
        - RUNNING: running the query
        - SUCCEEDED: execution was successful, result data available for fetch
        - FAILED: execution failed; reason for failure described in accompanying error message
        - CANCELED: user canceled; can come from explicit cancel call, or timeout with on_wait_timeout=CANCEL
        - CLOSED: execution successful, and statement closed; result no longer available for fetch
    - `error:` the different error codes that can be generated from the SQL execution.
        - `Error_code: UNKNOWN | INTERNAL_ERROR | TEMPORARILY_UNAVAILABLE | IO_ERROR | BAD_REQUEST | SERVICE_UNDER_MAINTENANCE | WORKSPACE_TEMPORARILY_UNAVAILABLE | DEADLINE_EXCEEDED | CANCELLED | RESOURCE_EXHAUSTED | ABORTED | NOT_FOUND | ALREADY_EXISTS | UNAUTHENTICATED`
- `manifest:` object providing the schema and metadata for the result set
- `result:` object containing the result data

Once the `state` field is "SUCCEEDED", you can retrieve the query result. The code snippet below is an example of how to retrieve the result set in Python. There is a placeholder get() function that manages the GET request:

```
while True:
     resp =
     get(f"/api/2.0/genie/spaces/{SPACE_ID}/conversations/{conversation_id}/messages
     /{message_id}/query-result")['statement_response']
     state = resp['status']['state']
     if state == 'SUCCEEDED':
       data = resp['result']
       meta = resp['manifest']
       rows = [[c['str'] for c in r['values']] for r in data['data_typed_array']]
       columns = [c['name'] for c in meta['schema']['columns']]
       df = spark.createDataFrame(rows, schema=columns)
       display(df)
       return df
     elif state == 'RUNNING' or state == 'PENDING':
       print(f"Waiting for query result...")
       time.sleep(5)
     else:
       print(f"No query result: {resp['state']}")
       return None
```

3. Ask a follow-up question + retrieve SQL results

You can also ask follow up questions to an existing conversation thread for deeper data insights (just like in the UI) by leveraging the "Create conversation message" function.

Use an existing chat thread to ask a follow-up question

To ask a follow-up question, you need an existing conversation thread.

This POST request to create a new chat thread with an initial question will look as follows:

```
POST /api/2.0/genie/spaces/{space_id}/conversations/{conversation_id}/messages
Authorization: <your_authentication_token>
{
    "content": "What about the year before?",
}
```

An example API response and the detailed descriptions are available here.

The important field to note:

- `message_id:` unique message id of the submitted question. Use this ID to track its status and get SQL query results in future steps.

Get status of your submitted question + retrieve generated SQL
The steps are the same as outlined in the steps above: linked here. Be sure to use the new message id generated from this follow up question.

Fetch query results
The steps are the same as outlined in the steps above: linked here

4. Re-execute SQL from previous message

At times, you may want to re-execute a SQL query that has expired query results (previously executed SQL becomes expired after 7 days). You can do this with the "Execute SQL query in a conversation message" endpoint.

It's important to understand that this re-executes the previously generated SQL for the message. It doesn't regenerate a new SQL response.

Execute SQL query in a conversation message

This `POST` request will re-execute a message's SQL will look as follows::

```
POST
/api/2.0/genie/spaces/{space_id}/conversations/{conversation_id}/messages/{message_id}/exec
ute-query
```

An example API response and the detailed descriptions are available in the appendix.

The important field to note:

- `statement_id:` this statement ID field is returned upon successfully submitting a SQL statement

Fetch query results
The steps here are the same as outlined in the steps above: linked here

---

2.5 Reference Code

Example notebook that uses Genie Conversation API (REST Api)

3. AI/BI Dashboard Integration

3.1 AI/BI Dashboards

Databricks AI/BI Dashboards are an AI-powered, low-code business intelligence solution integrated into the Databricks Data Intelligence Platform, offering features for creating interactive visualizations and analytical datasets. They leverage AI-assisted authoring, allowing users to create and iterate on visualizations using natural language, while providing high-performance interactions across various data volumes. These dashboards integrate seamlessly with Databricks' Unity Catalog for unified governance, can be easily shared within organizations, and work alongside AI/BI Genie to enable natural language querying of data.

Doc page

- https://docs.databricks.com/aws/en/dashboards/

3.2 APIs and Access Methods:

- `GET /api/2.0/lakeview/dashboards`
  - `List all Dashboards`
  - https://docs.databricks.com/api/workspace/lakeview/list
- `GET /api/2.0/lakeview/dashboards/{dashboard_id}/published`
  - To get details on a published dashboard
  - https://docs.databricks.com/api/workspace/lakeview/getpublished
- `GET /api/2.0/lakeview/dashboards/{dashboard_id}`
  - To get the metadata for a dashboard
  - https://docs.databricks.com/api/workspace/lakeview/get
- `GET /api/2.0/permissions/{workspace_object_type}/{workspace_object_id}/permissionLevels`
  - Get the permission levels that a user can have on an object.
  - https://docs.databricks.com/api/workspace/workspace/getpermissionlevels
- Unity Catalog Metadata
  - Databricks API / sdk /connectors : For tables, schemas, access control.
  - SQL-based access to `INFORMATION_SCHEMA` and system tables for advanced metadata.

Refer to full api documentation around AI/BI dashboards, at https://docs.databricks.com/api/workspace/lakeview

4. Metadata integration

4.1 Metadata Indexing Options:

⚠️ Databricks workspaces often contain thousands of dashboards and Genie spaces. Therefore, integrations must scale using a centralized connector. Avoid designs that require configuring each dashboard or space individually.

💡 Note: Visualizations in AI/BI dashboards are derived from datasets defined by SQL queries. You only need to extract metadata for the datasets — not separately for both visualizations and datasets.

- **Option 1 (API polling):** Use REST APIs or SDKs to crawl dashboards and metadata.
  - Simpler but may hit rate limits.
- **Option 2 (recommended):** Run a Databricks job to extract metadata into a Delta table, then query it.
  - You can package this logic as a Python wheel (.whl) or a Scala/Java .jar, instead of a notebook. This makes deployment easier and decouples it from the Databricks UI. Operationalize submitting and scheduling as a Databricks Job using the REST api
  - More scalable, better performance, and avoids throttling.
  - Product UI's can query these tables using a SQL warehouse

4.2 What to Index:

Here is a partial list of what you can index

- **Datasets referred to:** catalog > schema > table > column
- **Dashboard metadata**: title, descriptions, owner, associated SQL queries, associated datasets, visualizations, access controls and others.
- **Genie spaces:** names, descriptions, associated datasets, access controls and others.
- **Unity Catalog metadata:** catalog, schema, table, table metadata, column names and column types and others.

4.3 Reference Code:

- Example notebook for metadata extraction

---

5. Search & Assistant Integration

- Search results and assistant responses should include both dashboard and Genie content, each with **clear source attribution** (links to the relevant Databricks asset).

---

6. Security & Access Control

6.1 Support for OAuth

- **Support both OAuth modes:**

Auth Flow	**Use Case**
U2M (User-to-Machine)	For end-user queries via search or assistant.
M2M (Machine-to-Machine)	Metadata indexing using service principals

- **OAuth 2.0** U2M is required to ensure data access is properly scoped to the authenticated user.

6.2 Others

- **Personal Access Tokens (PATs)** may be used temporarily for legacy customers.
- **Metadata jobs** should run under Service Principals with read-only access.
- Use a **dedicated catalog and schema** for staging extracted metadata (recommended for organization and security). This catalog and schema should have ACL's set as per organization guidelines.
- **Respect Unity Catalog ACLs**: Only return results users are authorized to see.

---

7. Validation Requirements

Validation is a standard part of Databricks' integration process. It ensures your solution meets our quality, security, and performance standards — and delivers a great user experience for shared customers.

To pass validation, your integration must:

- Include support for both Genie and AI/BI Dashboards.
- Show clear citations and links back to original Databricks content.
- Avoid non-scalable patterns (e.g., per-dashboard connections).
- Demonstrate working OAuth authentication (U2M and/or M2M).
- Use Unity Catalog metadata to enforce governance across Genie, Dashboards, and data assets (spaces, tables, and other objects).
- ⚠️ For any APIs not yet GA (e.g., **Genie list spaces**), you must use private APIs with approval or wait for public release

When you are ready to begin the Databricks Validation Process, open a support ticket here.

---

8. Testing & QA Recommendations

8.1 Functional Testing:

- Use a test workspace with sample dashboards and Genie spaces.
- Perform user acceptance testing (UAT) with real dashboards and queries.
- Confirm assistant and search experiences behave as expected.

8.2 Performance & Scale:

- The design patterns of metadata indexing jobs will ensure that you don't hit Databricks API limits (or metadata APIs).
- Using SQL warehouse to query the metadata from your custom metadata tables allows for scalability

8.3 Access & Permissions:

- Validate OAuth tokens and correct user-to-space routing.
- Ensure all results respect Unity Catalog access controls.

Conclusion

Integrating with Databricks AI/BI isn't just about adding functionality — it's about giving your users a smarter, faster, and more intuitive way to explore data. By connecting Genie and Dashboards into your platform, you're enabling governed, AI-powered analytics where your users already work.

Follow the steps in this guide to build a scalable, secure, and approved ("validated") integration.

Next steps:

✅ Complete your onboarding checklist

✅ Validate your integration with the Databricks team

✅ Contact your Partner Development Manager to align on your go-to-market plan and launch with confidence

Appendix/References

📘 API Docs

- Databricks REST API Reference
  - Genie REST API Docs
  - AI/BI Dashboards REST API Docs
- Databricks Python SDK
  - Genie API

🛠️ SDKs & Connectors

- Databricks SDK for Python
  - Genie API Docs
  - AI/BI Dashboards API Docs
- Databricks Python SQL connector
  - Execute SQL to get metadata for catalogs, schemas, tables
  - To get metadata about catalog,schemas, tables and columns, Query information schema for a catalog. Information schema | Databricks Documentation

📊 Metadata Extraction

- Information Schema Overview: Using SQL to query for metadata for catalog, schema, table, and columns.

📄 Sample Code

- Notebook: Extract Dashboard Metadata (by Prasad Kona): Uses Python SDK to write metadata to a Delta table for easy querying.
- Notebook: Conversation with data using Databricks AI/BI Genie (by Prasad Kona): Uses the REST api for Genie to start a conversation and have followup conversations