Conversational Analytics API architecture and key concepts

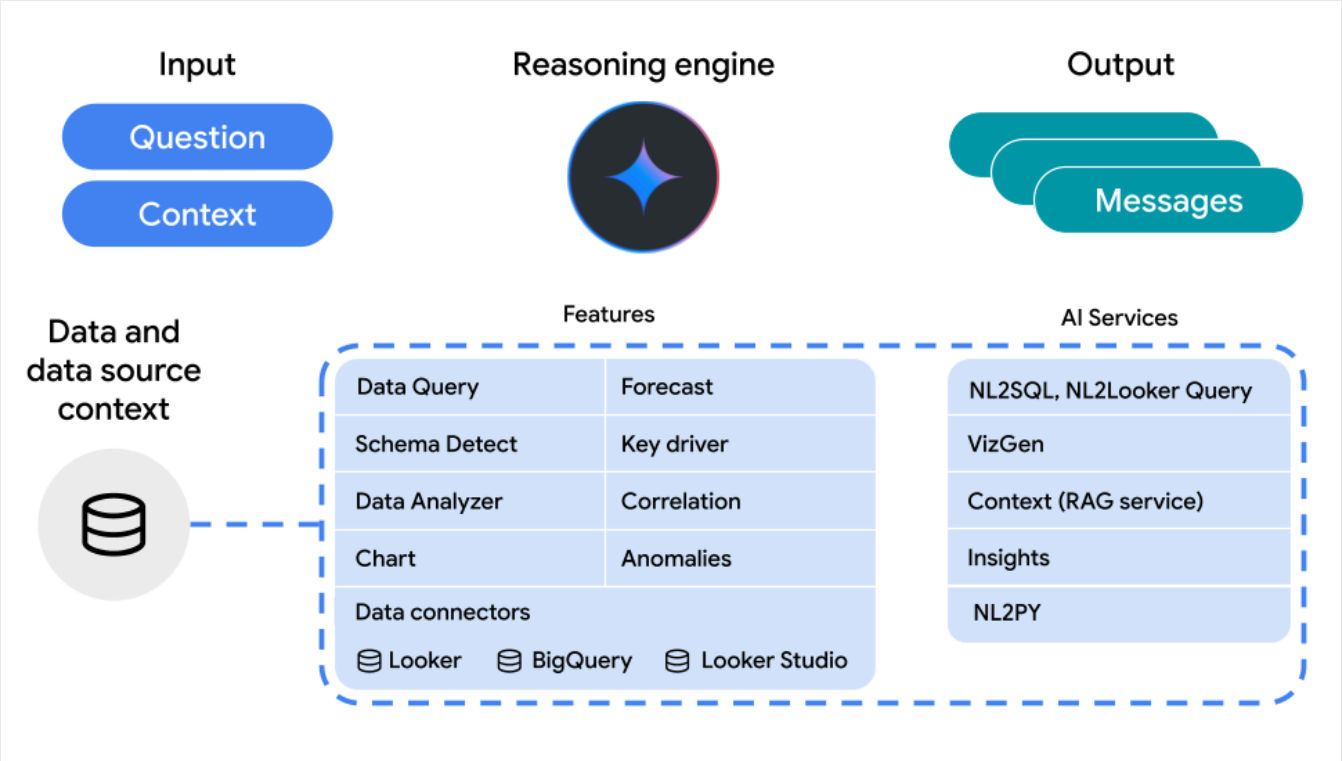
This product or feature is subject to the "Pre-GA Offerings Terms" in the General Service Terms section of the Service Specific Terms (/terms/service-terms#1). Pre-GA products and features are available "as is" and might have limited support. For more information, see the launch stage descriptions (/products#product-launch-stages).

This document describes the key concepts for using the Conversational Analytics API (`geminidataanalytics.googleapis.com`), which lets you create and interact with data agents that use natural language to answer questions about your structured data. This document describes how agents work, typical workflows, conversation modes, Identity and Access Management (IAM) roles, and how to design systems with multiple agents.

How data agents work

Conversational Analytics API data agents use *context* that you provide (business information and data) and tools (such as SQL and Python) to interpret natural language questions and generate responses from your structured data.

The following diagram illustrates the stages of an agent's workflow when a user asks a question:



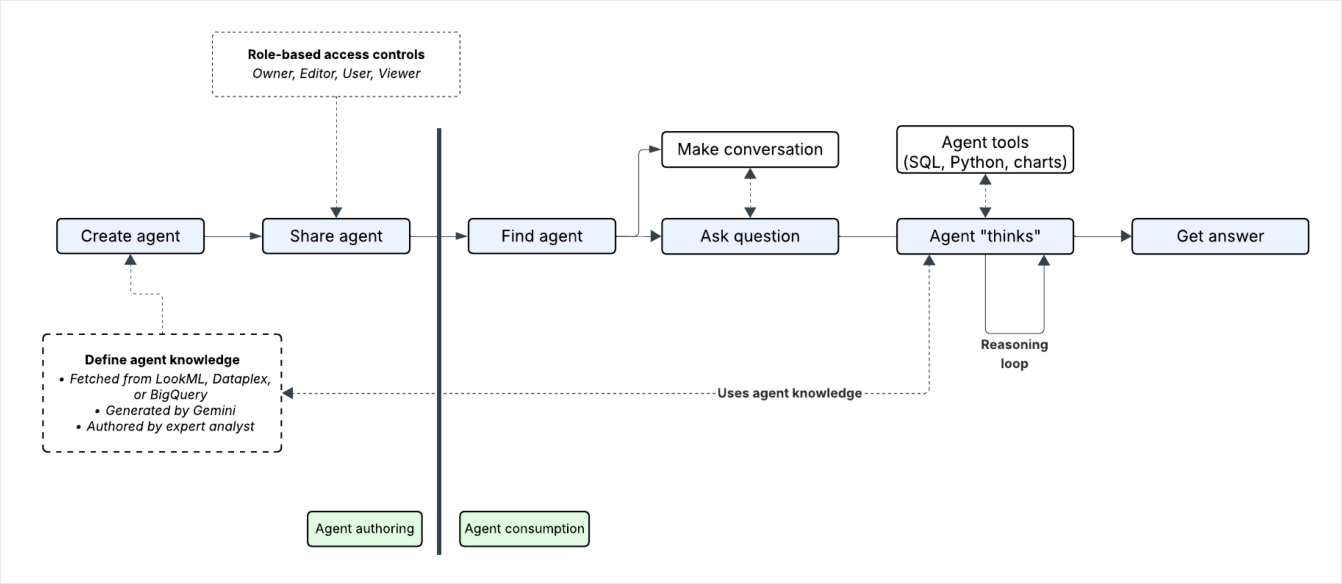
As shown in the diagram, when a user asks a question, the agent processes the request in the following stages:

1. **User input**: The user submits a question in natural language, along with any additional context you provide.
2. **Data sources**: The agent connects to your data in Looker, BigQuery, and Looker Studio to retrieve the necessary information.
3. **Reasoning engine**: The core of the agent processes the user's question by using available tools to generate an answer.
4. **Agent output**: The agent generates a response, which can include text, data tables, or specifications for charts.

Workflows for designing and using agents

The Conversational Analytics API supports workflows for agent creators (who build and configure agents) and for agent consumers (who interact with agents).

The following diagram illustrates the end-to-end process, from the initial setup by an agent creator to the final interactions from an agent consumer:



The following sections describe the workflows for agent creators and agent consumers in more detail.

The agent creator workflow

The agent creator is responsible for setting up and configuring agents. This workflow involves the following steps:

1. **Create agent**: The creator starts by creating a new agent and providing the necessary context, including system instructions and connections to data sources. This step is crucial for enabling the agent to understand and respond to user questions effectively.
2. **Share the agent**: Once the agent is configured, the creator shares it with other users and sets the appropriate role-based access controls to manage permissions.

The agent consumer workflow

The agent consumer is typically a business user who needs to get answers from a configured agent. This workflow involves the following steps:

1. **Find an agent**: The user starts by finding an agent that has been shared with them.
2. **Ask a question**: The user asks a question in natural language. This question can be a single query or part of a multi-turn conversation.

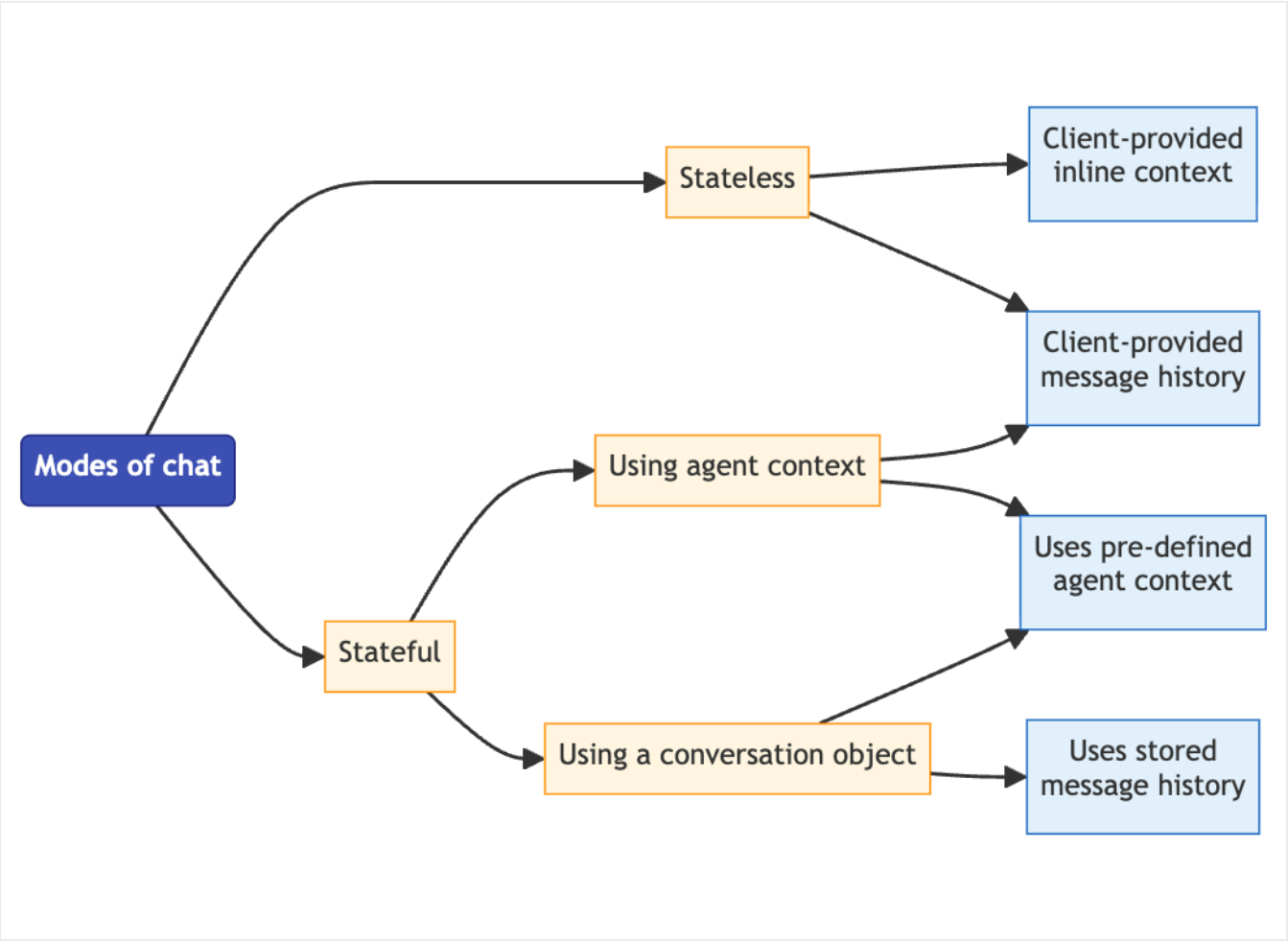
3. **Agent "thinks"**: The agent's reasoning engine processes the question. The reasoning engine uses the agent's predefined knowledge and available agent tools (like SQL, Python, and charts) in a "reasoning loop" to determine the best way to answer the question.
4. **Agent responds**: The agent returns an answer, which can include text, data tables, or charts.

Conversation modes

Conversational Analytics API agents support different conversation modes that determine how an agent handles conversation history and the persistence of context across interactions. The following conversation modes are available:

- **Stateless mode**: The agent doesn't store conversation history. Each interaction is treated independently. This mode is useful for applications where you don't need to maintain context across multiple turns.
- **Stateful mode**: The agent retains context and conversation history, allowing for more contextualized interactions. This mode is useful for applications where you need to maintain context across multiple turns. Using stateful mode is recommended for better accuracy and personalized responses.

Choose a conversation mode based on your application's requirements for conversation history and context persistence.



IAM roles

IAM roles control who can create, manage, share, and interact with Conversational Analytics API agents. The following table describes the key IAM roles for the Conversational Analytics API (/gemini/docs/conversational-analytics-api/access-control):

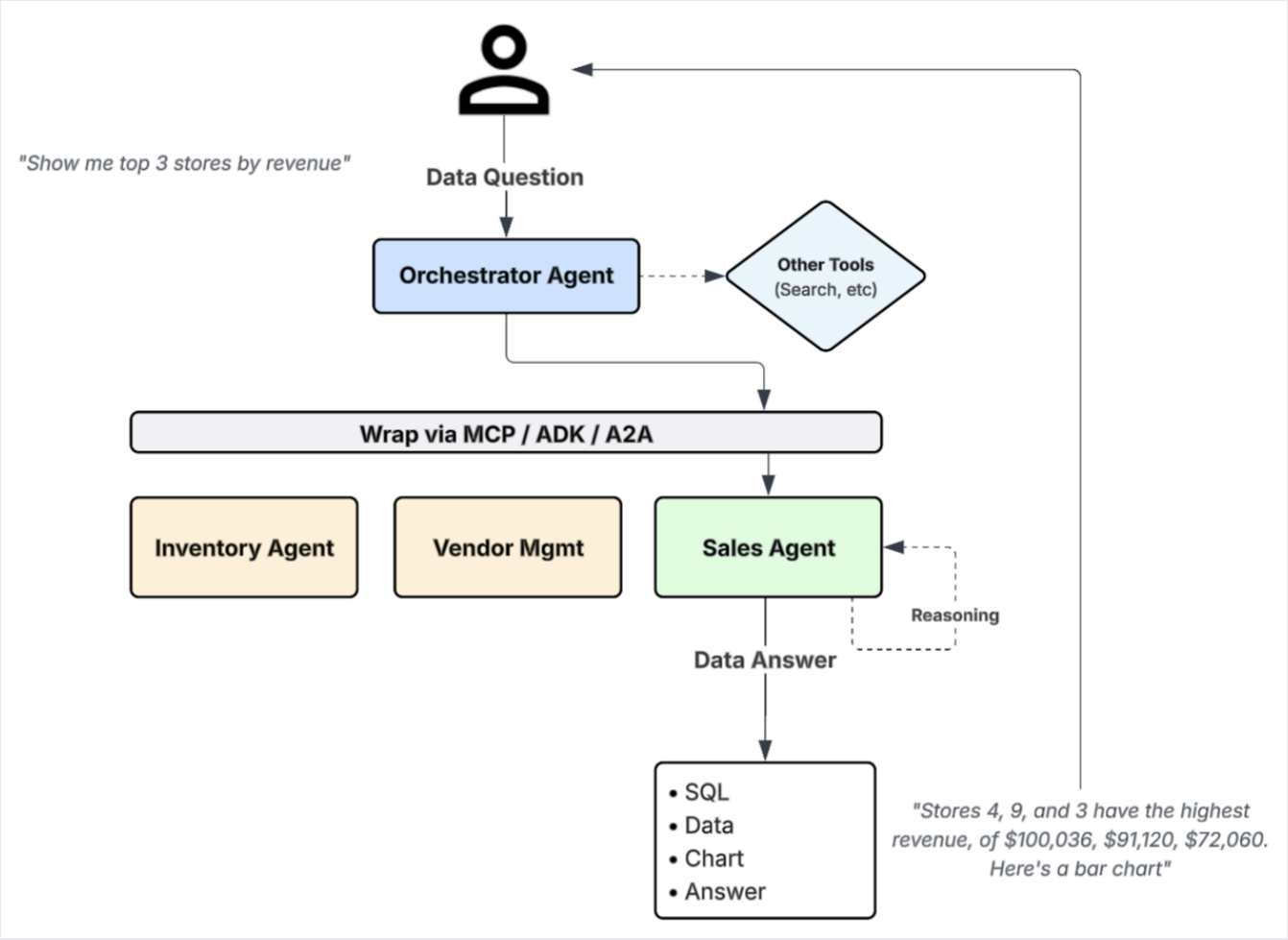
Role	Typical scope	What the role enables	Who might use this role
Gemini Data Analytics Data Agent Creator (`roles/geminidataanalytics.dataAgentCreator`)	Project	Create agents and inherit owner permissions on the agent.	Any data analyst
Gemini Data Analytics Data Agent Owner (`roles/geminidataanalytics.dataAgentOwner`)	Project, Agent	Edit, share, or delete agents with other users.	Senior data analyst

Role	Typical scope	What the role enables	Who might use this role
Gemini Data Analytics Data Agent Editor (`roles/geminidataanalytics.dataAgentEditor`)	Agent, Project	Update an agent's configuration or context.	Junior data analyst
Gemini Data Analytics Data Agent User (`roles/geminidataanalytics.dataAgentUser`)	Agent, Project	Chat with an agent.	Marketer, store owner
Gemini Data Analytics Data Agent Viewer (`roles/geminidataanalytics.dataAgentViewer`)	Project, Agent	List agents and get their details.	Any user
Gemini Data Analytics Data Agent Stateless User (`roles/geminidataanalytics.dataAgentStatelessUser`)	Project	Chat with an agent without storage of context or conversation history.	Any user

Systems with multiple agents

You can design complex systems by integrating multiple Conversational Analytics API agents. A common pattern is to use a primary "orchestrator" agent that delegates tasks to one or more specialized agents that handle specific domains, such as sales or marketing data. This approach lets you build a system that can handle a wide range of questions by combining the strengths of multiple agents.

The following diagram illustrates this multi-agent pattern and shows how a primary agent can delegate a data question to a specialized Conversational Analytics agent:



The typical workflow for a multi-agent system involves the following steps:

1. A business user or data analyst asks a question in natural language, such as "Show me the top three stores by revenue."
2. A primary "orchestrator" agent delegates the request to the appropriate specialized agent.
3. A specialized agent receives the delegated request, connects to the relevant data sources, uses its tools to generate the necessary SQL queries and charts, and generates a response.
4. The specialized agent's response is returned to the user, such as "Stores 4, 9, and 3 have the highest revenue. Here's a chart."

What's next

After understanding the core concepts of the Conversational Analytics API, explore how to implement these features:

- Explore how to authenticate and connect to a data source (/gemini/docs/conversational-analytics-api/authentication).
- Learn how to create and configure an agent with HTTP (/gemini/docs/conversational-analytics-api/build-agent-http).
- Learn how to create and configure an agent with Python (/gemini/docs/conversational-analytics-api/build-agent-sdk).
- Learn more about guiding an agent's behavior with authored context (/gemini/docs/conversational-analytics-api/data-agent-system-instructions).
- Understand access control with IAM for the Conversational Analytics API (/gemini/docs/conversational-analytics-api/access-control).

Last updated 2025-09-12 UTC.