Guide agent behavior with authored context

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 page describes the recommended structure for writing effective prompts for your Conversational Analytics API (/gemini/docs/conversational-analytics-api/overview) data agents. These prompts are authored context that you define as strings by using the `system_instruction` parameter. Well-structured system instructions can improve the accuracy and relevance of the responses that the API provides.

For examples of authored context in different environments, see the following documentation pages:

- Guide agent behavior with authored context for BigQuery data sources (/gemini/docs/conversational-analytics-api/data-agent-authored-context-bq)
- Guide agent behavior with authored context for Looker data sources (/gemini/docs/conversational-analytics-api/data-agent-authored-context-looker)

What are system instructions?

System instructions are user-defined guidance that developers can provide to shape the behavior of a data agent and to refine the API's responses. System instructions are part of the context that the API uses to answer questions. This context also includes connected data sources (BigQuery tables, Looker Explores, Looker Studio data sources), and conversation history (for multi-turn conversations).

By providing clear, structured guidance through system instructions, you can improve the agent's ability to interpret user questions and generate useful and accurate responses. Well-defined system instructions are especially useful if you're connecting to data such as BigQuery tables, where there may not be a predefined semantic layer like there is with a Looker Explore.

For example, you can use system instructions to provide the following types of guidance to an agent:

- **Business-specific logic**: Define a "loyal" customer as one who has made more than five purchases within a certain timeframe.
- **Response formatting**: Summarize all responses from your data agent in 20 words or fewer to save your users time.
- **Data presentation**: Format all numbers to match the company's style guide.

Provide system instructions

You can provide system instructions to the Conversational Analytics API as a YAML-formatted string by using the `system_instruction` parameter. While the `system_instruction` parameter is optional, providing well-structured system instructions is recommended for accurate and relevant responses.

You can define the YAML-formatted string in your code during initial setup, as shown in Configure initial settings and authentication (/gemini/docs/conversational-analytics-api/build-agent-http#configure_initial_settings_and_authentication) (HTTP) or Specify the billing project and system instructions (/gemini/docs/conversational-analytics-api/build-agent-sdk#specify_the_billing_project_and_system_instructions) (Python SDK). You can then include the `system_instruction` parameter in the following API calls:

- **Creating a persistent data agent**: Include the `system_instruction` string within the `published_context` object in the request body to configure agent behavior that persists across multiple conversations. For more information, see Create a data agent (/gemini/docs/conversational-analytics-api/build-agent-http#create_a_data_agent) (HTTP) or Set up context for stateful or stateless chat (/gemini/docs/conversational-analytics-api/build-agent-sdk#set_up_context_for_stateful_or_stateless_chat) (Python SDK).
- **Sending a stateless request**: Provide the `system_instruction` string within the `inline_context` object in the chat request to define the agent's behavior and context for the duration of that specific API call. For more information, see Create a stateless multi-

turn conversation (/gemini/docs/conversational-analytics-api/build-agent-http#create-a-stateless-multi-turn-conversation) (HTTP) or Send a stateless chat request with inline context (/gemini/docs/conversational-analytics-api/build-agent-sdk#send_a_stateless_chat_request_with_inline_context) (Python SDK).

YAML template for system instructions

The following template shows the recommended YAML structure for the string that you provide to the **`system_instruction`** parameter, including the available keys and expected data types.

You can use the following YAML template to provide your own system instructions:

```
- system_instruction: str # A description of the expected behavior of the agent.
- tables: # A list of tables to describe for the agent.
    - table: # Details about a single table that is relevant for the agent.
        - name: str # The name of the table.
        - description: str # A description of the table.
        - synonyms: list[str] # Alternative terms for referring to the table.
        - tags: list[str] # Keywords or tags that are associated with the table.
        - fields: # Details about columns (fields) within the table.
            - field: # Details about a single column within the current table.
                - name: str # The name of the column.
                - description: str # A description of the column.
                - synonyms: list[str] # Alternative terms for referring to the c
                - tags: list[str] # Keywords or tags that are associated with th
                - sample_values: list[str] # Sample values that are present with
                - aggregations: list[str] # Commonly used or default aggregation
        - measures: # A list of calculated metrics (measures) for the table.
            - measure: # Details about a single measure within the table.
                - name: str # The name of the measure.
                - description: str # A description of the measure.
                - exp: str # The expression that is used to construct the measur
                - synonyms: list[str] # Alternative terms for referring to the m
        - golden_queries: # A list of important or popular ("golden") queries fo
            - golden_query: # Details about a single golden query.
                - natural_language_query: str # The natural language query.
                - sql_query: str # The SQL query that corresponds to the natural
```

```
        - golden_action_plans: # A list of suggested multi-step plans for answer
            - golden_action_plan: # Details about a single action plan.
                - natural_language_query: str # The natural language query.
                - action_plan: # A list of the steps for this action plan.
                    - step: str # A single step within the action plan.
    - relationships: # A list of join relationships between tables.
        - relationship: # Details about a single join relationship.
            - name: str # The name of this join relationship.
            - description: str # A description of the relationship.
            - relationship_type: str # The join relationship type: one-to-one, o
            - join_type: str # The join type: inner, outer, left, right, or full
            - left_table: str # The name of the left table in the join.
            - right_table: str # The name of the right table in the join.
            - relationship_columns: # A list of columns that are used for the jo
                - left_column: str # The join column from the left table.
                - right_column: str # The join column from the right table.
- glossaries: # A list of definitions for glossary business terms, jargon, and a
    - glossary: # The definition for a single glossary item.
        - term: str # The term, phrase, or abbreviation to define.
        - description: str # A description or definition of the term.
        - synonyms: list[str] # Alternative terms for the glossary entry.
- additional_descriptions: # A list of any other general instructions or content
    - text: str # Any additional general instructions or context not covered els
```

Key components of system instructions

The following sections describe the key components of system instructions and how to use them to improve the quality of your agent's responses. These keys include the following:

- `system_instruction` (#define-the-agents-role-and-persona)
- `tables` (#describe-your-data-with-tables)
- `fields` (#describe-fields)
- `measures` (#define-measures)
- `golden_queries` (#golden-queries)
- `golden_action_plans` (#golden-action-plans)
- `relationships` (#define-relationships)

- **`glossaries`** (#explain-glossaries)
- **`additional_descriptions`** (#additional-descriptions)

Define the agent's role and persona with `system_instruction`

Define the agent's role and persona by using the **`system_instruction`** key. This initial instruction sets the tone and style for the API's responses and helps the agent understand its core purpose.

The following YAML code block shows the basic structure for the **`system_instruction`** key:

```
- system_instruction: str
```

For example, you can define an agent as a sales analyst for a fictitious ecommerce store as follows:

```
- system_instruction: >-
    You are an expert sales analyst for a fictitious ecommerce store. You will a
```

Describe your data with `tables`

The **`tables`** key contains a list of table descriptions for the agent and provides details about the specific data that is available to the agent for answering questions. Each **`table`** object within this list contains the metadata for a specific table, including that table's name, description, synonyms, tags, fields, measures, golden queries, and golden action plans.

The following YAML code block shows the basic structure for the **`tables`** key:

```
- tables:
    - table:
      - name: str # The name of the table.
      - description: str # A description of the table.
```

```
    - synonyms: list[str] # Alternative terms for referring to the table.
    - tags: list[str] # Keywords or tags that are associated with the table.
    - fields: # A list of the fields in the table.
    - measures: # A list of the measures in the table.
    - golden_queries: # A list of golden queries for the table.
    - golden_action_plans: # A list of golden action plans for the table.
```

Describe commonly used fields with `fields`

The `fields` key, which is nested within a `table` object, takes a list of field objects to describe individual columns. Not all fields need additional context; however, for commonly used fields, including additional details can help improve the agent's performance.

The following YAML code block shows the basic structure for the `fields` key:

```
- fields: # Details about columns (fields) within the table.
    - field: # Details about a single column within the current table.
        - name: str # The name of the column.
        - description: str # A description of the column.
        - synonyms: list[str] # Alternative terms for referring to the column.
        - tags: list[str] # Keywords or tags that are associated with the column
        - sample_values: list[str] # Sample values that are present within the c
        - aggregations: list[str] # Commonly used or default aggregations for th
```

Define business metrics with `measures`

The `measures` key, which is nested within a `table` object, defines custom business metrics or calculations that aren't directly present as columns in your tables. Providing context for measures helps the agent answer questions about key performance indicators (KPIs) or other calculated values.

The following YAML code block shows the basic structure for the `measures` key:

```
- measures: # A list of calculated metrics (measures) for the table.
    - measure: # Details about a single measure within the table.
        - name: str # The name of the measure.
```

```
        - description: str # A description of the measure.
        - exp: str # The expression that is used to construct the measure.
        - synonyms: list[str] # Alternative terms for referring to the measure.
```

Improve accuracy with `golden_queries`

The **`golden_queries`** key, which is nested within a **`table`** object, takes a list of **`golden_query`** objects. Golden queries help the agent provide more accurate and relevant responses to common or important questions. By providing the agent with both a natural language query and the corresponding SQL query for each golden query, you can guide the agent to provide higher quality and more consistent results.

The following YAML code block shows the basic structure for the **`golden_queries`** key:

```
- golden_queries: # A list of important or popular ("golden") queries for the ta
    - golden_query: # Details about a single golden query.
        - natural_language_query: str # The natural language query.
        - sql_query: str # The SQL query that corresponds to the natural languag
```

Outline multi-step tasks with `golden_action_plans`

The **`golden_action_plans`** key, which is nested within a **`table`** object, takes a list of **`golden_action_plan`** objects. You can use golden action plans to provide the agent with examples of how to handle multi-step requests, such as to fetch data and then create a visualization.

The following YAML code block shows the basic structure for the **`golden_action_plans`** key:

```
- golden_action_plans: # A list of suggested multi-step plans for answering spec
    - golden_action_plan: # Details about a single action plan.
        - natural_language_query: str # The natural language query.
        - action_plan: # A list of the steps for this action plan.
            - step: str # A single step within the action plan.
```

Define table joins with `relationships`

The `relationships` key contains a list of join relationships between tables. By defining join relationships, you can help the agent understand how to join data from multiple tables when answering questions.

The following YAML code block shows the basic structure for the `relationships` key:

```
- relationships: # A list of join relationships between tables.
    - relationship: # Details for a single join relationship.
        - name: str # A unique name for this join relationship.
        - description: str # A description of the relationship.
        - relationship_type: str # The join relationship type (e.g., "one-to-one
        - join_type: str # The join type (e.g., "inner", "outer", "left", "right
        - left_table: str # The name of the left table in the join.
        - right_table: str # The name of the right table in the join.
        - relationship_columns: # A list of columns that are used for the join.
            - left_column: str # The join column from the left table.
            - right_column: str # The join column from the right table.
```

Explain business terms with `glossaries`

The `glossaries` key lists definitions for business terms, jargon, and abbreviations that are relevant to your data and use case. By providing glossary definitions, you can help the agent accurately interpret and answer questions that use specific business language.

The following YAML code block shows the basic structure for the `glossaries` key:

```
- glossaries: # A list of definitions for glossary business terms, jargon, and a
    - glossary: # The definition for a single glossary item.
        - term: str # The term, phrase, or abbreviation to define.
        - description: str # A description or definition of the term.
        - synonyms: list[str] # Alternative terms for the glossary entry.
```

Include further instructions with `additional_descriptions`

The `additional_descriptions` key lists any additional general instructions or context that is not covered elsewhere in the system instructions. By providing additional descriptions, you can help the agent better understand the context of your data and use case.

The following YAML code block shows the basic structure for the `additional_descriptions` key:

```
- additional_descriptions: # A list of any other general instructions or content
    - text: str # Any additional general instructions or context not covered els
```


Last updated 2025-09-12 UTC.