



DATABLAU · DAM

Data governance agent

Product module user manual

Version 9.1.2

Contents

01	Data Governance Agents
02	Data Governance Agents quick start
03	Administer Data Governance Agents
04	Agent templates
05	Approvals and collaboration
06	Validate and publish context
07	Data intelligence spaces
08	End-to-end governance in a space
09	Governance capabilities
10	Set up Data Governance Agents
11	Configure model services
12	Orchestrate and run Routines
13	Design and publish Sprites
14	Sprites and Routines

Data Governance Agents

DIG is the product name for Data Governance Agents. The module in DAM is called **Intelligent Data Governance**. DIG adds AI assistance to DAM governance workflows and organizes governed asset semantics and knowledge into context for AI applications. Existing governance modules, approvals, and data permissions continue to apply. Without the appropriate license, the AI conversation icon and related pages are unavailable.

How DIG supports governance

DAM stores metadata, standards, quality rules, security classifications and levels, and catalog assets. DIG uses configured capabilities to propose changes against a defined objective. Confirmed results enter the relevant DAM module. Prepare accessible objects, business rules, and permissions before starting.

Open AI from a data source, table, or asset detail page for a specific task. For ongoing governance across a business domain, create a data intelligence space to organize assets, documents, and workflows.

Choose a starting point

Goal	Entry point	Guide
Complete a table's business names and descriptions	DAM AI conversation or object detail page	Quick start
Govern a business domain over time	Data intelligence space	Create a space
Assess governance, run workflows, and publish context	Space governance lifecycle	End-to-end governance
Make governed knowledge available to AI queries	Context Management and Publish	Validate and publish context
Review standards or classification proposals	Approval requests	Approvals and collaboration
Reuse governance actions and workflows	Configuration Management > Governance Sprites	Sprites and Routines
Configure models and tools	Configuration Management	Initial setup, model services, administration

Two phases of a space

DG by AI uses AI to assist governance. Add assets and documents, run the space agent or Routines to enrich metadata and organize standards, quality, security, and assets, then review the results.

DG for AI makes governed knowledge available to AI applications. Organize semantics, standards, lineage, and documents into trusted context. Validate it before enabling queries through a space Agent or MCP. Publishing context does not itself deliver underlying business data.

The workspace presents four steps: **Start > Data Governance > Context Management > Publish**. Determine completion from actual results rather than conversation length or execution counts.

Core concepts

Object	Purpose	Relationship
Agent	Understands goals and coordinates execution	Uses models, tools, Skills, Sprites, and Routines
Agent template	Stores a configuration snapshot that cannot execute directly	A published template creates an independent space instance; later updates do not overwrite it
Skill	Packages reusable instructions and execution methods	Guides a task; it is neither a data source nor an approver
Governance Sprite	Performs one bounded, atomic task	Defines inputs, tools, and outputs; can be tested independently
Routine	Combines Sprites into a defined workflow	Controls node order, parameter mappings, and failure handling
Data intelligence space	Organizes scope, documents, and agents for each phase	Supports governance and subsequent context use

MCP (Model Context Protocol) connects agents with external capabilities through defined tools. Configuring an MCP service lets DIG call external tools. Publishing a space MCP lets external AI applications use that space's context tools. Configure their addresses, authorization, and scope separately.

Read by responsibility

Data stewards can start with the [quick start](#) and [governance capabilities](#), then follow [end-to-end governance](#). Approvers should read [approvals](#). Space owners should read [space operations](#) and [context publishing](#). Administrators should complete [setup](#), [templates](#), and [administration](#). Actual roles and permissions are configured by your organization.

Sprite designers should read [Sprite design](#) before workflow designers proceed to [Routine orchestration](#).

Validate the outcome

Check model output against enterprise policies and actual data. Confidence scores, successful task status, and an agent's completion message do not prove that DAM contains the correct result. Available agents, templates, and tools depend on deployment configuration; example names are not guaranteed built-in resources.

Data Governance Agents quick start

Screenshot note: Screenshots are illustrative. Controls or labels may remain untranslated in the demo release, and pages without a faithful English capture retain the source illustration. Control names and layouts can vary by release.

DIG appears in DAM as **Intelligent Data Governance**. This guide uses a single table to demonstrate business metadata enrichment: open a conversation, select scope, generate proposals, review them, and verify DAM results. Use a table you are authorized to govern.

Prerequisites

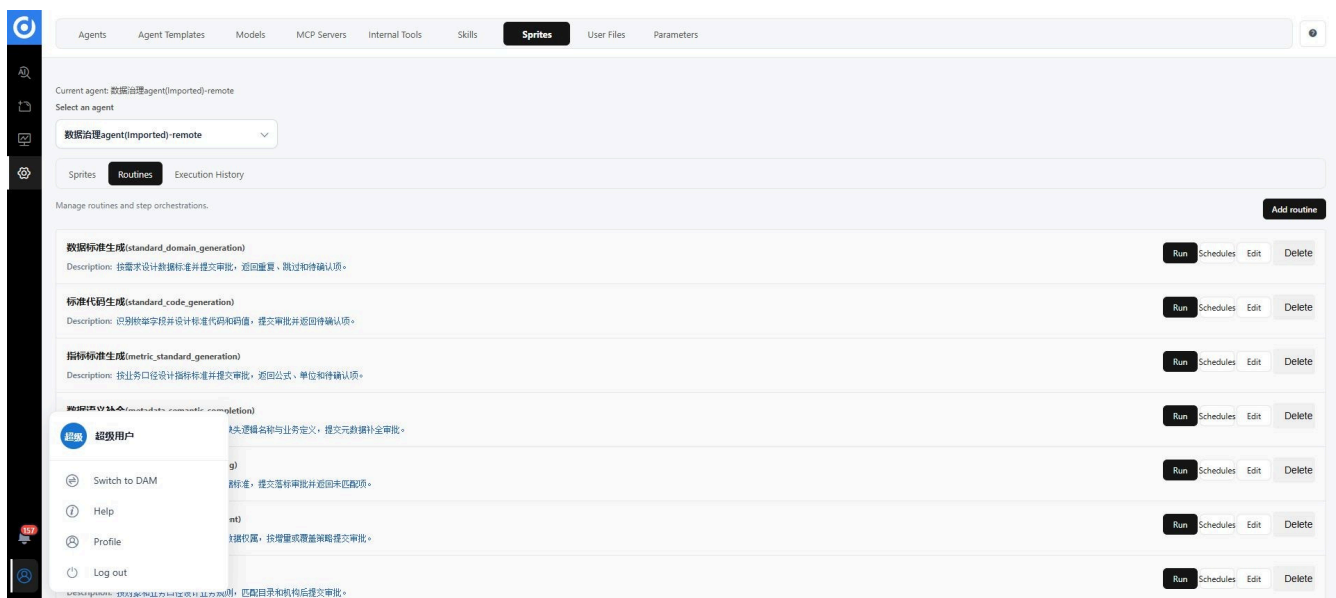
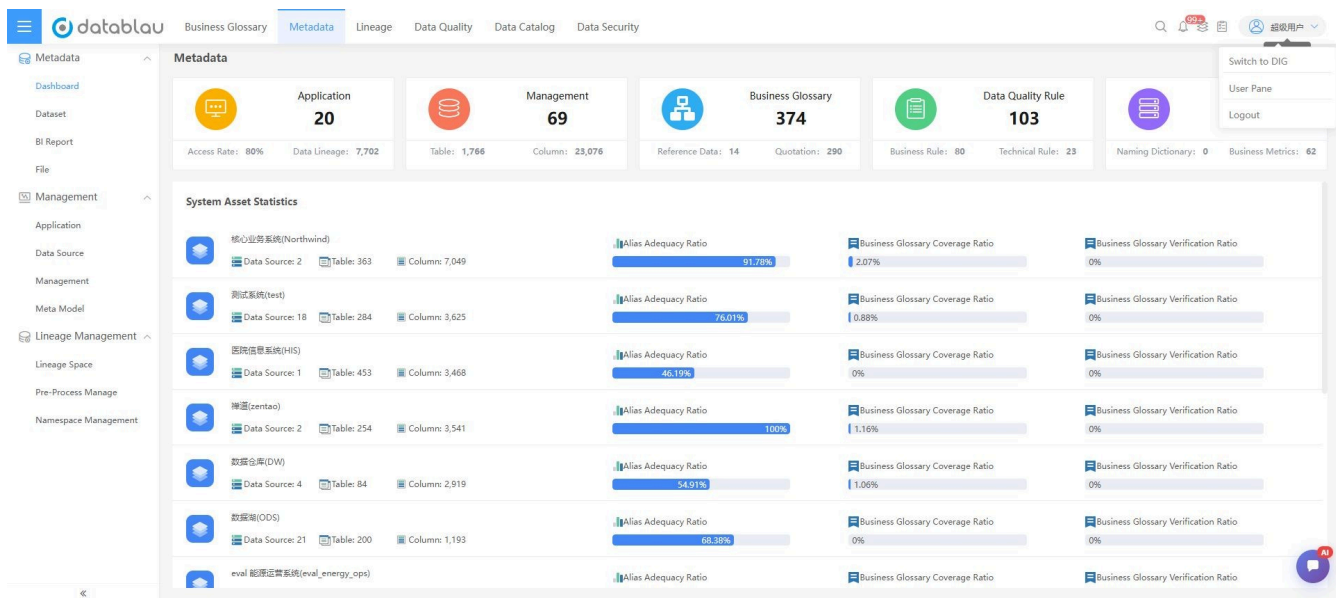
- DIG is deployed, enabled, and licensed.
- You can access the DIG entry point and target objects.
- An administrator has configured a working model, agent, and tools.
- You have defined the goal, scope, reference policies, and acceptance criteria.
- Uploaded material contains only information you are authorized to use.

Without DIG, the AI icon and related pages are hidden. Role names are configurable; this guide describes responsibilities rather than fixed system roles. Administrators should complete [initial setup](#) if no agent is available. For ongoing domain governance, [create a space](#).

Step 1: Open DIG

Open the AI icon at the bottom right of DAM. A magic wand or governance action on an object detail page includes that object's context. Verify the object and its system after opening the conversation.

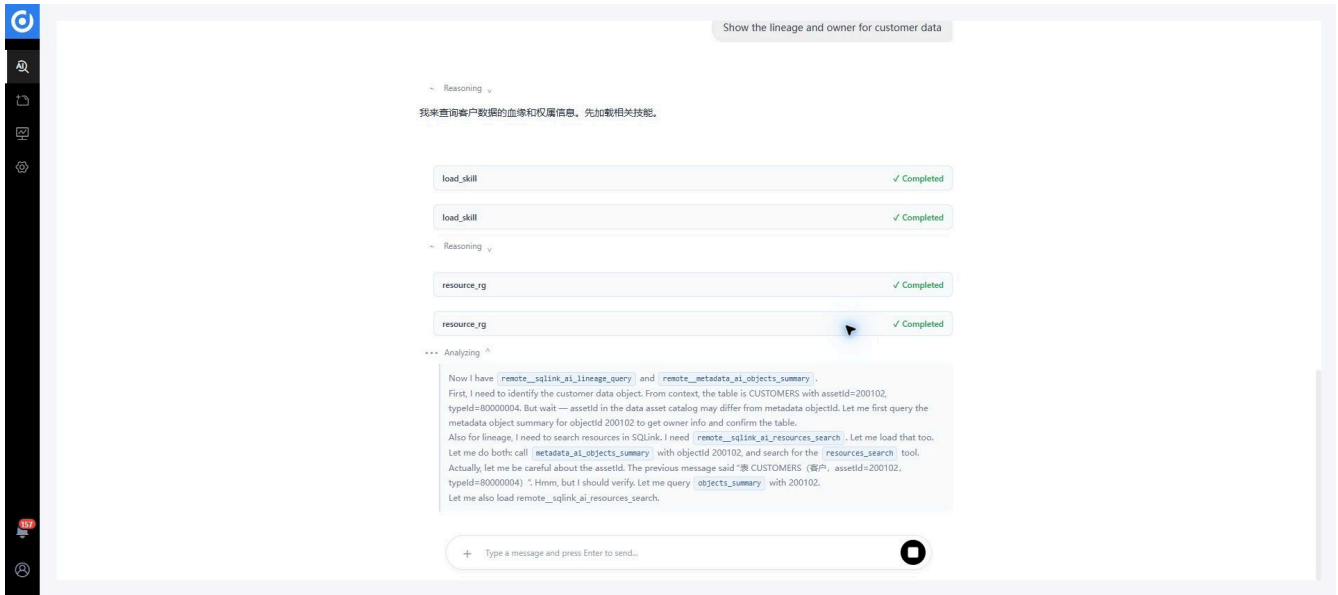
To open the standalone workspace, choose **Switch to DIG** from the DAM user menu at the top right. To return, choose **Switch to DAM** from the user menu at the bottom left of DIG.



Step 2: Start a conversation

Conversation controls

Depending on permissions, the conversation toolbar provides Conversation/Plan, New Session, agent selection, user processes, approval requests, saved prompts, file upload or references, and session history.



[Open the full image.](#) Administrators use **Configuration Management** in DIG to configure agents and resources.

Governance dashboard

The dashboard shows AI governance metrics, approval completion trends for core workflows over the last five days, historical task statistics, and live executions. Use these to track progress, then verify results on the DAM objects.

Create a session

1. Open AI in DAM and select **New Session**.
2. Choose an agent suited to the task.
3. Describe the goal, object scope, references, constraints, and expected result.
4. Upload supporting documents or select authorized files when needed.
5. Submit the request and observe execution.
6. Check proposal scope and errors before reviewing the approval request.

Request element	Include
Goal	Enrich, generate, validate, map standards, or inventory assets
Scope	Data source, table, fields, subject area, or other objects
Evidence	Enterprise policies, published standards, or business context
Constraints	Exclusions, naming rules, or thresholds
Acceptance	How a reviewer can determine whether proposals are usable

Use object selection controls where available. Verify system and environment; manually typing an object name can select the wrong object with the same name.

Step 3: Submit a bounded task

Open an authorized table, use its governance entry point, and verify the table, system, and fields. You can select a Chinese business name enrichment prompt from **My Prompts** and replace its example scope and business context, or use this example:

```
Inspect the selected table. Propose values only for fields that lack a Chinese business name or business description.
Use the business documentation I provide and preserve existing, accurate descriptions.
List each object, current value, proposed value, and rationale. If the meaning is uncertain, identify the missing evidence.
Submit proposed changes for human confirmation.
```

This is an example, not a fixed system rule. Avoid unintended scope such as all data sources. See [governance capabilities](#) for other task inputs.

Session identifiers and states

A session key combines agent identifier and username. Only one session for that key is ACTIVE. Each conversation instance has a separate Session ID; other historical sessions are CLOSED. Include the identifier and incident time when reporting an issue.

State	Meaning	Action
ACTIVE	The session can continue or retains active context	Verify current objects and files before continuing
CLOSED	The session is closed	Reactivate it from history or start a new session

Before reactivating history, confirm that referenced documents, policies, permissions, and object scope remain valid.

Upload and reuse files

1. Upload a file in the session or select one from user files.
2. Check its name, source, purpose, and visibility.

3. Specify which file and sections the task should use.
4. Verify that the output uses the material correctly.

Check revision, effective date, permissions, and sensitive information before reuse. Documents provide context; reviewers still determine policy applicability.

Start governance from DAM

Supported DAM pages provide a magic wand or governance action. It carries object context into the task, but you must still verify scope, system, and task type. Test a small sample before processing a batch.

Step 4: Monitor execution

Inspect tool steps, progress, processed objects, omissions, and errors. **User Processes** shows your task execution. If a task stalls, record its session identifier and ask an administrator to check model and tool services before resubmitting the same large task.

Treat generated content as proposals and submit governance results for human review in this workflow.

Step 5: Review and verify DAM

1. Open **My Approval Requests** from the conversation and find the task.
2. Compare current and proposed values against the business evidence.
3. Edit, approve, or reject items and record reasons when needed.
4. Return to the table and verify approved Chinese business names and descriptions.
5. Supply missing context for unresolved fields and retry only the remaining scope.

See [approvals and collaboration](#) for details. Expand scope gradually or use a [space](#) after this initial task succeeds.

Completion checklist

- The correct agent and objects were used.
- Reference documents and policies remain valid.
- Actual scope matches the request; skipped and failed items have explanations.
- Proposals were reviewed rather than treated as published results.
- Sessions, processes, files, and approvals can be traced to one another.

Troubleshooting

DIG is unavailable. Check deployment, license, functional permissions, and module entry points with an administrator. A DAM account alone does not grant DIG access.

A historical session cannot continue. Check state and try reactivation. If the agent, files, or permissions have changed, create a new session with valid context.

Proposal scope is incorrect. Stop further approval, reselect objects, and specify system, environment, table, and fields. Split large tasks when needed.

Related guides

- [Governance capabilities](#)
- [Approvals and collaboration](#)
- [Administration](#)

Administer Data Governance Agents

Screenshot note: Screenshots are illustrative. Controls or labels may remain untranslated in the demo release, and pages without a faithful English capture retain the source illustration. Control names and layouts can vary by release.

Use **Configuration Management** in DIG to maintain agents and reusable resources. Select the correct agent before maintaining its models, tools, Skills, files, and governance capabilities.

Configuration relationships

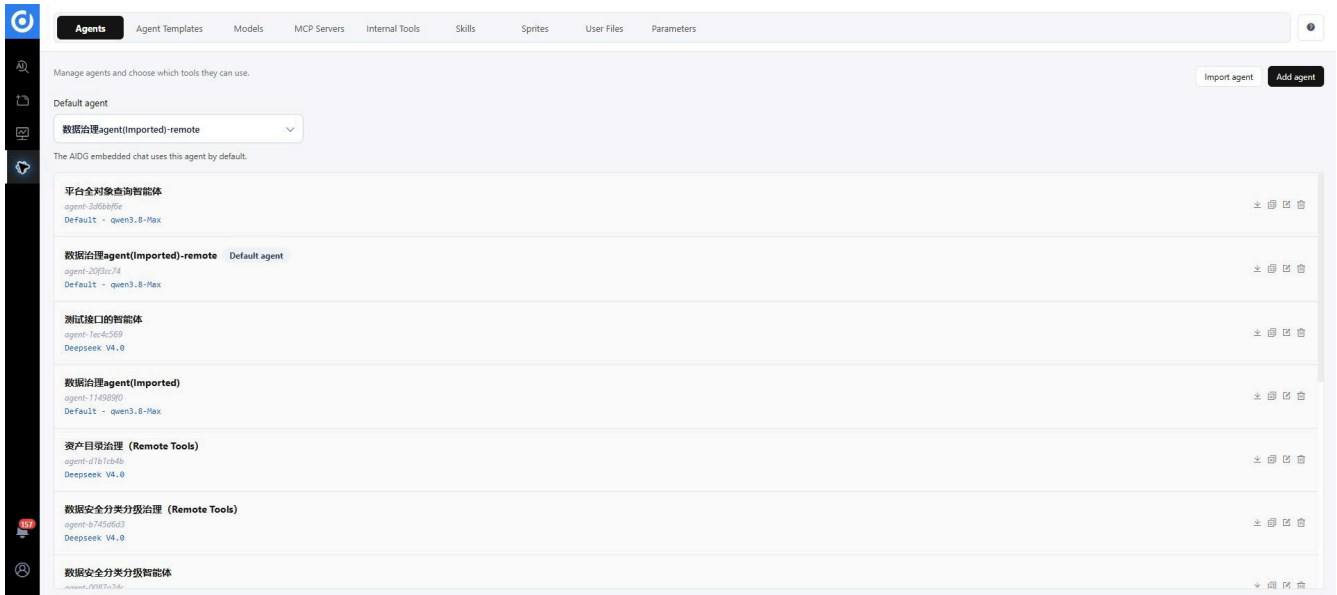
text

```
Agent
|-- Model
|-- MCP services
|-- Internal services
|-- Skills
|-- Prompts and parameters
`-- Workspace

Sprites perform individual tasks. Routines orchestrate Sprites.
User files supply controlled context.
```

Apply least privilege: configure only the models and tools an agent needs.

Agent management



Create or import an agent

1. Open **Configuration Management > Agents** and select **Add Agent** or **Import Agent**.
2. Enter a name and define responsibilities, permitted scope, and output requirements in the system prompt.
3. Select an approved model.
4. Configure the system prompt and required parameters.
5. Bind necessary tools, internal services, and Skills.
6. Configure the Workspace as needed.
7. Test with non-sensitive samples.
8. Save and use the available publishing workflow. Verify agent selection in a new session.

Review package origin, dependencies, permissions, prompts, and environment values before publishing an imported agent.

Agent templates

Templates are configuration snapshots that cannot execute directly. Data Governance Agent templates and Business Query Agent templates serve different phases. Instances are isolated from subsequent template updates, and Routine schedules are not copied. See [agent templates](#).

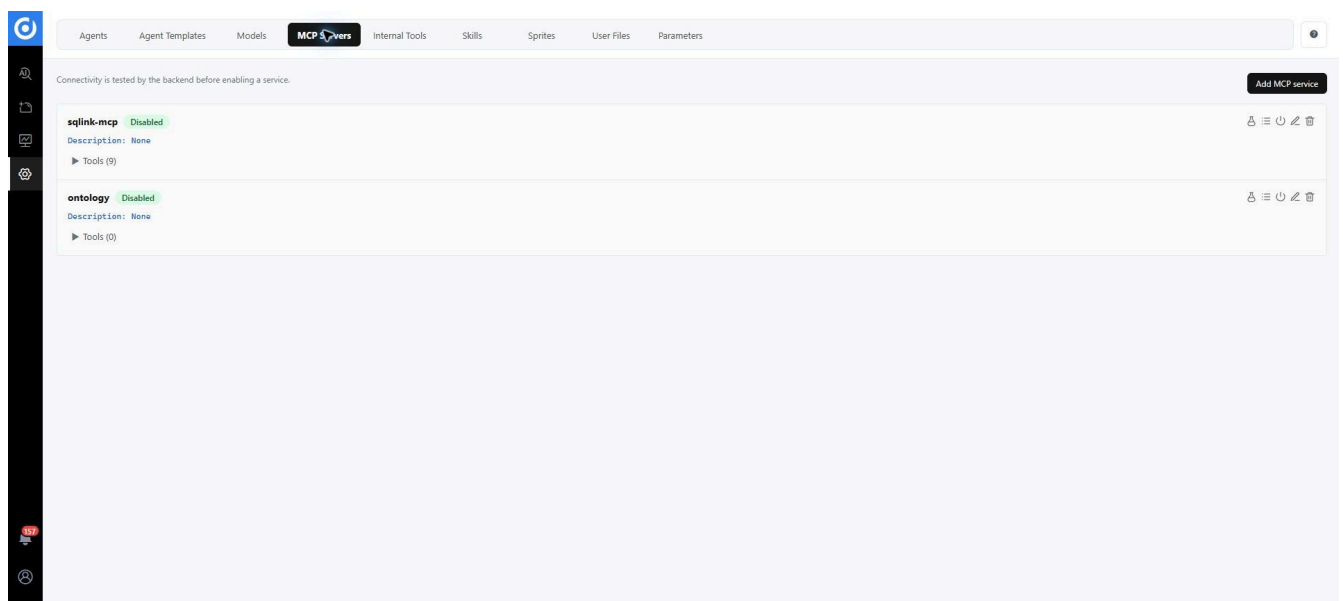
Test and publish

Test normal and empty input, wrong objects, excessive scope, insufficient permissions, and unavailable services. Confirm the required approval workflow and that tools and data remain restricted to authorized users.

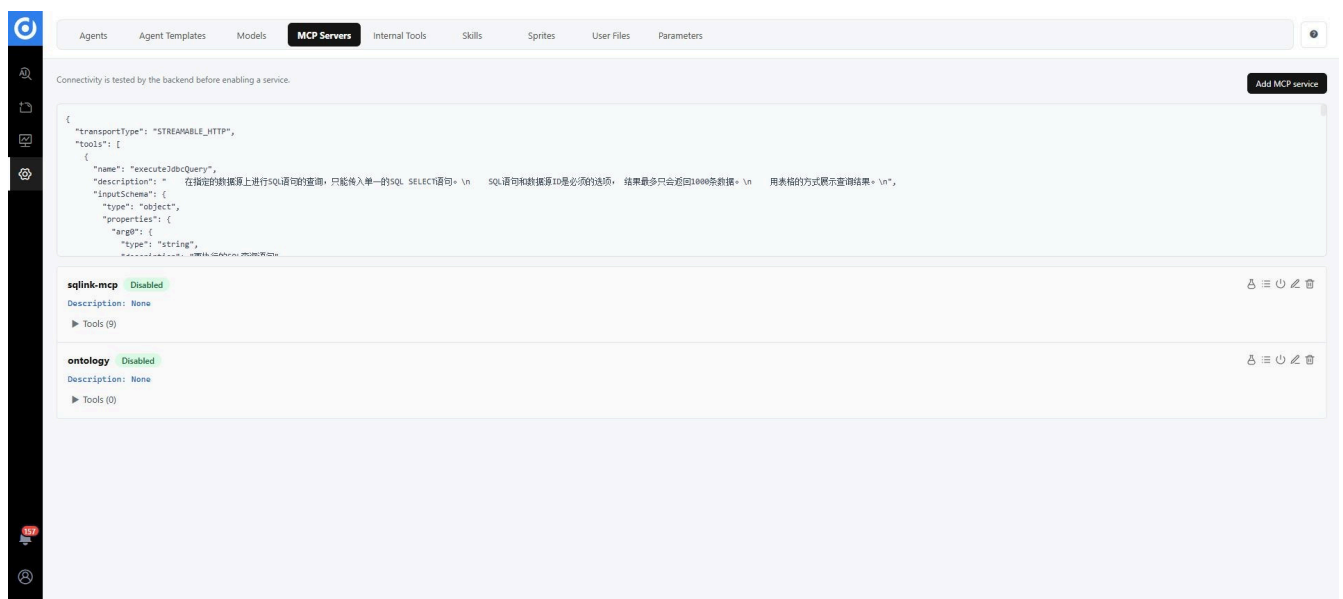
Model management

Use **Model Configuration** for shared LLM and Text Embedding resources. A connection test must pass before saving, and embedding dimensions must match actual output. Review the [recommended LLM parameters](#) and [model service guide](#). Verify a small governance task in the target agent after saving. Default model and default agent are separate settings.

MCP management



MCP services let DIG call tools from other applications. Open **Configuration Management > MCP Services > Add MCP Service**.



[Open the full image](#). Addresses and names are environment examples.

Field	Purpose
Name	Identifies the service in the list
Service address	MCP server address accessible from DIG Server
Connection mode	Streamable HTTP or SSE, matching the server
Connection timeout (ms)	Timeout for a connection test; adjust the initial value for your network and service
Custom settings JSON	Valid JSON for additional server settings
Headers JSON	Required request headers; manage credentials according to enterprise policy
Description	Purpose and applicable agents

Click **Test**, then save. **Tools (count)** lists exposed tools; **View Details** shows definitions. The backend also tests connectivity before enabling the service. Connectivity does not establish correct output or authorization for every tool. Check dependent agents, Skills, and Sprites before deletion or disabling.

Internal services

Open **Configuration Management > Internal Services**. Use **Upload YAML** for bulk import or **Add Internal Service** for one definition.

Field	Meaning
Name	Display name
Target application	DAM or another application that owns the interface
Description	Purpose, scope, and risk
Service interface	Interface declaration or identifier
Service path	Invocation path
Method name	Underlying interface method
Parameters JSON	Input definition

Saved changes take effect immediately. Internal services expose program interfaces without requiring a separate MCP or CLI for every interface. Review inputs, outputs, identity propagation, and write effects, then test a minimal sample. Apply stricter authorization and approval to create, update, publish, and delete services than to read-only queries. Quick invocation does not bypass business permissions.

Skill management

Skills package reusable governance instructions, Markdown prompts, package paths, interface dependencies, tool constraints, and test cases.

Open **Configuration Management > Skill Management** and choose a scope:

- **Agent:** stored in the current agent's working directory and enabled in sessions according to configuration. Scan the workspace, upload ZIP, or choose **Add Agent Skill**. Supply Skill code, name, and content. Existing Skills can be enabled by default for the agent.
- **Global:** add, edit, delete, and inspect system-available Skills by code, name, and content.
- **User:** maintain the current user's Skills; this scope differs from global availability.

Preview Markdown after editing. Check JAR or Python dependency paths and DAM interfaces, then test typical and failure cases. Reuse default Skills where suitable. For interfaces requiring user identity, custom Skills should reference `${session.token}` at runtime rather than embed a fixed token.

Governance Sprites

Open **Configuration Management > Governance Sprites**, select the agent, and switch between Sprites, Routines, and Execution History. Sprites perform bounded tasks; Routines organize order, mappings, and failure handling. See [Sprites and Routines](#).

User files

Files provide context for sessions and tasks. Administrators can manage all user files; ordinary users manage their own, subject to deployment permissions.

In **User File Management**, select the agent to edit content, reference a file in AI, download, rename, or delete. New sessions can reuse uploaded files. Check references before replacement or deletion so historical tasks retain their evidence.

Parameters

Open **Parameter Configuration**. System parameters come from runtime. Global parameters and **My Parameters** are static values substituted when distributing Skills and invoking prompts. They support adding, editing, copying, deletion, and preview.

Examples include `{workspace.dir}` for the agent tool directory and `{usr.name}` for the current username. See [initial setup](#) for `baseUrl` and identity. Check destination user and purpose when copying credentials, and exclude real values from screenshots and feedback.

Saved prompts

Maintain prompts in the interface or import a supported format. Include goals, scope, rules, constraints, and expected output. Preview and test before user release. Prompts are instructions; enforce permissions in the actual tools and services.

Release checks

- Agents are visible only to authorized users.
- Model, MCP, and internal service connections work and errors are diagnosable.
- Agents bind only required tools and Skills.
- Confirm whether each write task requires manual review, batch approval, or permitted automatic approval under enterprise policy. An available approval page does not mean every task enforces manual approval.
- File and parameter visibility follows security requirements.
- Credentials are excluded from ordinary pages, logs, prompts, and exports.
- Configuration, testing, and publication have audit records.

Troubleshooting

Saved configuration appears ineffective. Compare the selected agent with the agent used by the session. Check dependencies and parameters. Template changes do not update existing space instances.

Model tests pass but tasks fail. Check the DAM URL, user identity, and tool logs. Models, MCP, and internal services are separate connection stages.

A new session cannot find a file. Check user, agent, file list, and whether the file was deleted, replaced, or referenced incorrectly.

Related guides

- [Quick start](#)
- [Governance capabilities](#)
- [Approvals and collaboration](#)

- [System management](#)

Agent templates

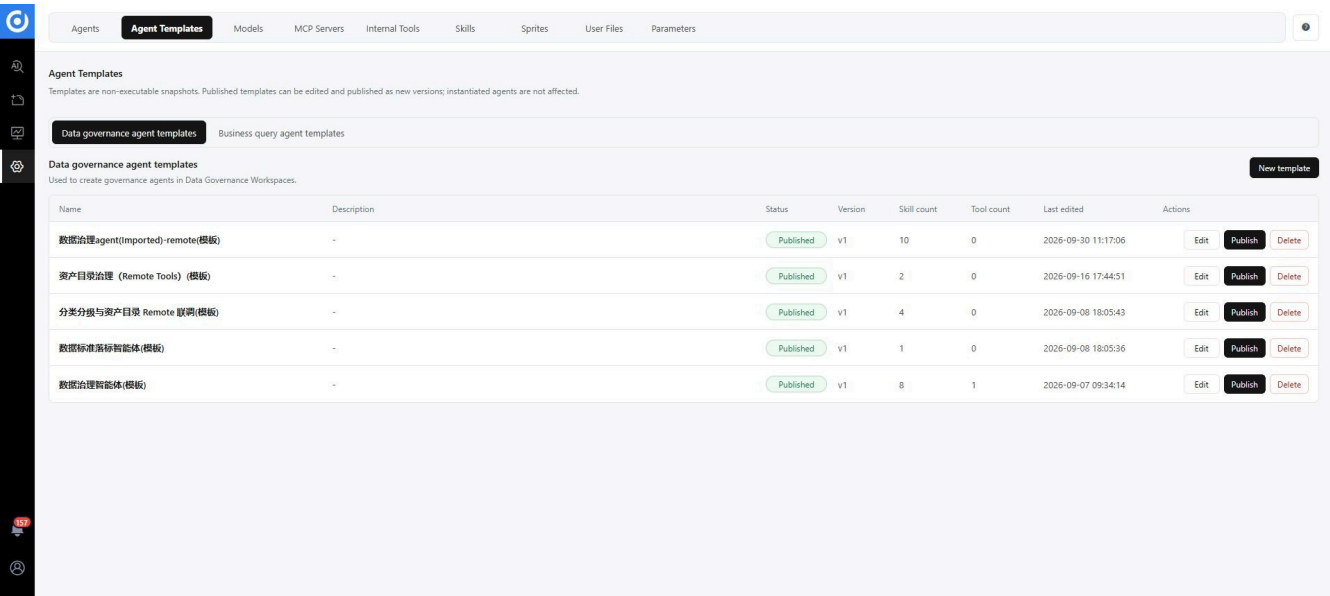
Screenshot note: Screenshots are illustrative. Controls or labels may remain untranslated in the demo release, and pages without a faithful English capture retain the source illustration. Control names and layouts can vary by release.

An agent template is a configuration snapshot that cannot execute directly. A space creates its own agent instance from the template rather than running tasks on the template.

Choose a template type

Open **Configuration Management > Agent Template Management** and select the appropriate section.

Type	Purpose	Used in
Data Governance Agent template	Organize governance capabilities, Sprites, and Routines	Space governance phase
Business Query Agent template	Query and explain governed assets and knowledge	Space context phase



Open the full image. Names and revision labels in the screenshot are examples.

Prepare the source agent

In [agent management](#), prepare and verify the source agent's model, system prompt, MCP, Skills, Sprites, and Routines. Publishing a template does not test or repair those dependencies.

Create a draft

1. Click **New Template** in the appropriate section.
2. Select the source agent whose configuration you want to copy.
3. Enter a name and description that explain business scope and capability boundaries.
4. Click **Create Draft**.
5. Open the draft and review basic information, system prompt, MCP, Skills, Sprites, and Routines.

The system copies these configurations but **does not copy Routine schedules**. Sprites and Routines appear as read-only snapshots in the template.

Publish a template

1. Finish reviewing the draft.
2. Check Skill and tool counts in the list.
3. Click **Publish**.
4. Verify **Published** status, last edit time, and revision history.
5. Check that the template appears under the correct type when creating a space.

The list's version identifies a template configuration revision, not a product release.

Edit and publish a revision

Click **Edit**, save changes, and publish from the list. A new revision does not automatically update existing agents or spaces. Test a new instance before promoting changes. Manage existing instance configurations independently; refreshing the page does not upgrade them.

Configure instance schedules

Schedules are excluded from templates. Configure and enable them on the target agent instance. Verify actual trigger times and execution history rather than assuming that a copied Routine is scheduled.

Before deleting a template

Confirm it is no longer needed to create spaces and check dependent users. Existing instances retain their configurations, but new instances can no longer be created from the deleted template.

Troubleshooting

Symptom	Check
Template missing during space creation	Template type and Published status
Cannot converse with a template	Instantiate it as an agent or space first
Source agent changed but template did not	Refresh and publish the snapshot as appropriate
Existing space unchanged after template update	Inspect the independent instance configuration
Routine exists but does not run periodically	Configure the target instance schedule

Related guides

- [Create a space](#)
- [Sprites and Routines](#)
- [Initial setup](#)

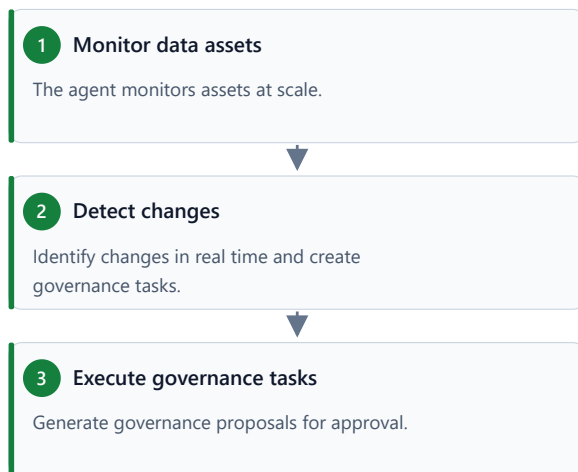
Approvals and collaboration

Screenshot note: Screenshots are illustrative. Controls or labels may remain untranslated in the demo release, and pages without a faithful English capture retain the source illustration. Control names and layouts can vary by release.

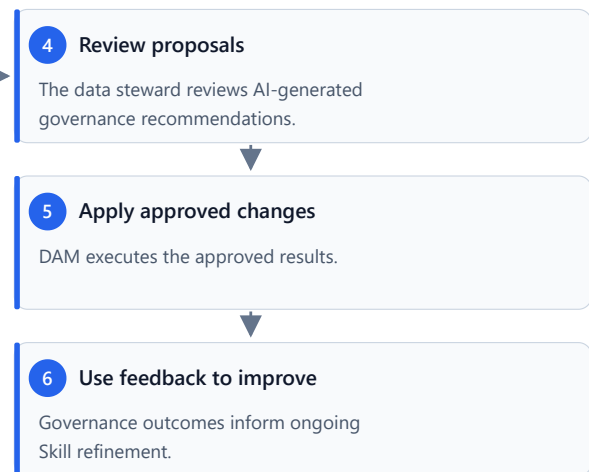
Whether proposals enter manual approval depends on the task and enterprise configuration. Manual review is recommended by default for changes to classification, standards, and metadata. Define the scope of any batch or automatic approval policy before release. For an approval request, reviewers must verify facts, policy, and target objects, then correct or reject unreliable proposals before submitting.

Human-in-the-loop governance

Agent automation



Steward decisions and execution



Data stewards provide judgment and final approval at key decision points.

Open an approval request

1. Open **My Messages** and select a pending request, or enter from a conversation or governance task.
2. Search the list or open **Filter** to enter contains-keyword conditions. Multiple field filters apply together.
3. Verify request name, type, identifier, total rows, and pending count.
4. Switch between **Pending**, **Approved**, and **Rejected** to inspect decisions.

These states describe approval handling rather than publication or execution in DAM. Check the target module after approval.

Review details

The list typically shows assets, classification categories or standards, proposed fields, confidence, and rationale. Click **View** for row-by-row review and editable fields. The page also provides **Approve All**, **Reject All**, **Reset All**, **Export**, and **Submit**, with counts and progress.

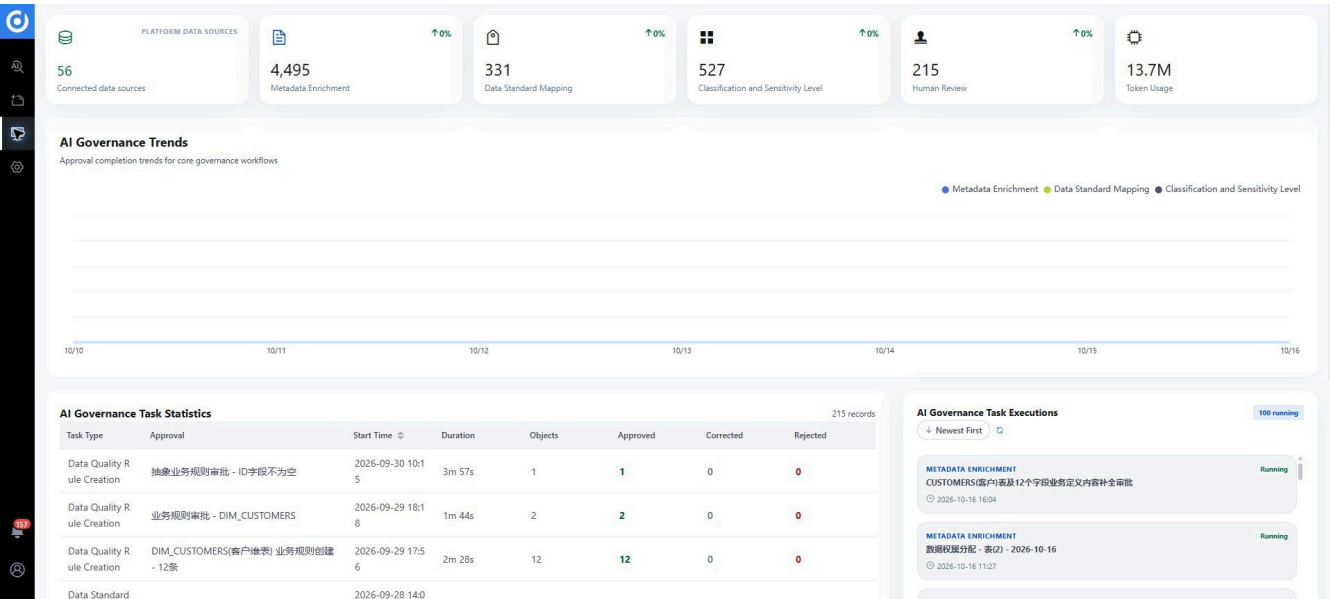
Verify that:

- The object belongs to the correct system, environment, and domain.
- Before values match current DAM data.
- Proposed values satisfy business facts and enterprise policies.
- Proposals do not duplicate or conflict with published content.
- Rationale explains the proposal without fabricated evidence.
- The change is within the reviewer's responsibility.

Edit, approve, or reject

- **Edit:** correct an otherwise useful proposal in row review before deciding.
- **Approve:** approve complete, accurate, compliant proposals individually or in a verified batch.
- **Reject:** reject incorrect objects, evidence, or content and retain the reason.

In **Continuous Mode**, approving a row opens the next row on the current page. Otherwise, the view stays on the same row. Decisions can be reset before submission. Submitted results are read-only. Click **Submit** after completing row decisions to submit the entire request.



[Open the full image](#). Shortcuts apply only to pending rows that have not been submitted.

Give specific reasons, such as incorrect scope, a conflicting policy definition, or incompatible technical properties.

Confidence scores

Scores and ordering help prioritize uncertain proposals. They do not replace business judgment: high-confidence output can lack context, while low-confidence output can contain useful evidence.

Process a batch

1. Filter to the same task, object type, and rule scope.
2. Check counts, exceptions, and low-confidence items.
3. Sample details, before-and-after values, and rationale.
4. Exclude high-risk items that require individual review.
5. Approve or reject the confirmed batch.
6. Check successful, failed, and unprocessed counts.

Increase sampling or review every row for classification, standards publication, key quality rules, and broad metadata changes. Scores alone are insufficient for batch approval.

Close the feedback loop

Approved and rejected lists provide history. Feed recurring edits and rejection reasons into governance rules, prompts, and agent improvements.

1. Define a governance goal.
2. Generate proposals.
3. Compare, edit, approve, or reject.
4. Process results through the relevant DAM module.
5. Verify actual status there.
6. Improve configuration using recurring feedback.

This guide does not assume automatic approval. If enabled, product, governance, and risk owners should confirm scope, sampling, and rollback, and check remaining pending items.

Verify results

- The request matches the session, task, and objects.
- Decisions match edited content and reasons.
- Batch scope excludes exceptions and unrelated records.
- Results or subsequent states are visible in the target DAM module.
- Records retain operator, time, and necessary change information.

Troubleshooting

An approval request is missing

Check functional permissions, data scope, reviewer configuration, and whether the task reached approval.

Before values differ from DAM

Pause approval. Check object identity, concurrent changes, and stale context. Regenerate or resolve the conflict through the applicable process.

A batch partially fails

Record failed objects and errors. Check permissions, states, and concurrent updates. Retry only confirmed failures.

Related guides

- [Quick start](#)
- [Governance capabilities](#)
- [Administration](#)

Validate and publish context

After governance, organize asset semantics, standards, lineage, and reviewed documents into context that AI can cite. This phase validates knowledge, scope, and the query agent's ability to explain assets. It does not automatically deliver underlying business data.

Enter Context Management

Complete [space governance and verification](#), then open **Context Management**. Resolve pending proposals, failed tasks, and business disputes so queries use confirmed results.

Define the context scope

Context can include asset business names and descriptions, standards, lineage and impact relationships, metrics, reports, reviewed documents, dictionaries, and policies. Select materials based on intended questions and exclude unrelated or outdated content. Publishing descriptions and knowledge does not authorize release of underlying data.

Review the Context quality report

The report identifies whether materials support queries across asset counts, semantics, standards, lineage, and classification. It is neither a model answer accuracy measure nor a quality rule execution report.

Dimension	Question	Remediation
Asset count	Are required objects present, with unnecessary objects excluded?	Adjust space scope
Chinese business semantic completeness	Can users understand table and field meanings?	Enrich and review business metadata
Standards coverage	Are key fields associated with correct standards?	Review standards mapping and validation
Lineage coverage	Can sources and flows be explained?	Check metadata lineage or SQLink results
Classification and security levels	Are sensitivity and levels clear before external use?	Complete classification and security review

There is no universal passing score in this guide. Combine the report with task and organizational requirements. Completeness statistics cannot establish that every definition is correct.

Validate the query agent

The query phase uses a space instance of a **Business Query Agent template**, with responsibilities distinct from the governance agent. Check its template, model, and query tools.

1. Ask about a governed table's purpose; verify its business name, description, and ownership.
2. Ask which standard applies to a field; verify the standard and actual association.
3. Ask where an object comes from; verify the lineage path.
4. Ask a question unsupported by the space; verify that missing evidence is stated.

If answers contain plausible definitions without evidence, correct context or tools and repeat validation.

Choose a publishing format

Format	Consumer	Capability
Space Agent	People or integrated AI applications	Natural-language queries and explanations using authorized context
Space MCP	External AI agents	Standard tools for assets, semantics, lineage, and knowledge

Both use the space's governance results, but verify invocation and authorization separately. Space MCP serves external consumers; an external MCP configured in DIG provides tools to DIG.

In **Publish**, review the available settings, intended consumers, and scope. Use actual client connection details from the page rather than screenshot addresses.

Verify after publication

- Authorized consumers can access the capability.
- Responses remain within permitted space context.
- Answers trace to actual assets or reviewed files.
- Unauthorized and unsupported questions do not produce fabricated certainty.
- Recheck context quality and answers after asset, rule, or file changes.

Troubleshooting

The report still shows gaps. Check whether affected objects were in scope, proposals were approved, and DAM results actually changed. Execution success does not complete every dimension.

Answers are irrelevant. Check query template, selected context, and files. Ensure the instance is a query agent for the correct space.

External agents cannot use MCP. Check publication, access scope, client details, and network reachability. Use the space's published endpoint.

Related guides

- [Data intelligence spaces](#)
- [Agent templates](#)
- [Administration](#)

Data intelligence spaces

Screenshot note: Screenshots are illustrative. Controls or labels may remain untranslated in the demo release, and pages without a faithful English capture retain the source illustration. Control names and layouts can vary by release.

A data intelligence space brings together the assets, documents, and governance capabilities needed for a business goal. Create a bounded workspace with its own governance agent and traceable results, then proceed to context validation.

The **source configuration** is the system Agent or published template selected at creation. An **instance** is the independent copy used by the space. Source changes do not overwrite existing instances. **Space context** is the set of assets and documents available for the current task; adding assets does not copy source database data.

Prerequisites

- Define a bounded goal, such as completing business descriptions for a sample customer table.
- Confirm access and governance permissions for target assets.
- Have an administrator prepare an available system Data Governance Agent. Some environments create it from a published template; the wizard lists the sources available to your account.
- Prepare applicable documents and identify the reviewer.

A space organizes context without granting additional source data permissions.

Step 1: Find spaces

Open **Data Intelligence Spaces**. Switch between **All**, **Mine**, and **Shared**, search by name, change **Space Sort**, and choose grid or list view. Mine contains spaces you created or can manage; Shared contains shared spaces you can access. Check filters and sharing permissions if a space is missing.

Step 2: Create a space

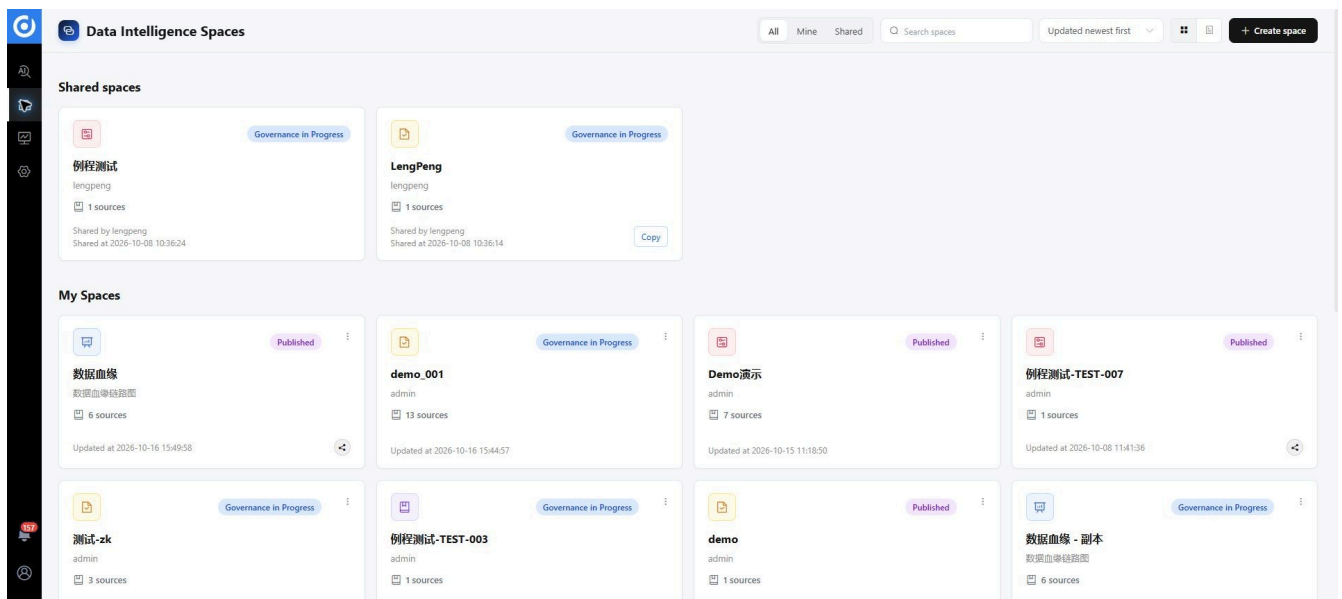
1. Click **Create Governance Space**.
2. In **Start**, enter a name describing the domain and goal, plus a description with scope and acceptance criteria.
3. Under **Select Data Governance Agent**, choose a system Agent and expand it to review capabilities. Use **Go to Template Management** if a source needs to be created.
4. Click **Confirm and Continue** after selection. This action is disabled until an Agent is selected.

5. Verify the space name and dedicated Agent, then configure target objects.

The space receives an independent agent copy. A later source update does not change it.

Step 3: Add assets and documents

The workspace follows **Start > Data Governance > Context Management > Publish**. During governance, assets appear on the left, conversation or the knowledge graph in the center, and workflows and execution records on the right.



[Open the full image](#). Space, asset, and workflow names are examples.

1. Add the systems, data sources, tables, or other objects to govern.
2. Search assets and check the source of objects with matching names.
3. Select objects for the current task. **Added to the space** and **selected as current conversation context** are different states.
4. Add applicable business documentation, dictionaries, and standards.
5. Check the selected count and object list before submitting a task.

Step 4: Run governance

Describe a task in the conversation or select a configured workflow on the right. Start with one table or a few fields.

Inspect the selected sample table. Propose values only for fields missing a Chinese business name or business description.

Use the business terminology documents in this space and preserve existing, accurate descriptions. List each object, current value, proposed value, and rationale. Submit proposed changes for human confirmation.

This prompt is an example. Before running a Routine, read its parameters. Workflow names and counts depend on the template and instance. See [Sprites and Routines](#).

Step 5: Review and verify

1. Inspect execution records for successful, failed, and unprocessed objects.
2. Open [approval requests](#) and compare values and rationale.
3. Edit, approve, or reject proposals and record reasons.
4. Verify actual attributes, rules, or catalog results in DAM.
5. Resolve remaining approvals, failures, and missing business context.

Step 6: Manage context

Choose **Governance Complete** to proceed to [context validation and publishing](#). This action advances the workspace; the owner still needs to confirm actual results and outstanding work.

Completion checks

Area	Expected outcome
Scope	Correct asset sources and current task selection
Agent	Independent instance from a suitable source, with working dependencies
Execution	Traceable records and explanations for failures and omissions
Review	Proposals processed and actual results verified in DAM
Next phase	Context can be organized without publishing out-of-scope content

Troubleshooting

No source Agent is available. Check account visibility of system Agents. If agents are template-managed, check template type and publication in **Agent Template Management**. Agents and templates are separate objects.

An added asset is ignored. Check current context selection, permissions, and prompt scope.

Template updates do not affect the space. Manage the instance or create a new one from an appropriate template.

A shared space is missing. Check view filters, search terms, and sharing scope.

End-to-end governance in a space

Use DAM AI conversations for individual tasks. Create a DIG data intelligence space when you need to govern multiple objects, track changes, or retain policies and results as reusable context. The space connects issue discovery, inputs, execution, validation, and publishing in a traceable workspace.

This guide describes a lifecycle, not a fixed built-in automation. Available agents, Sprites, Routines, graphs, and publishing options depend on license, configuration, and permissions.

Choose a workspace

Task	Entry point	Result handling
Single table, field, or one-time question	DAM AI conversation	Review proposals and verify the relevant DAM module
Multiple object types and documents in a domain	DIG space	Track assets, documents, execution, and acceptance together
Repeated governance or metadata change monitoring	Space Routines and schedules	Review and verify triggered results according to configuration
Context queries from external AI applications	Context Management and Publish	Validate before publishing a space Agent or MCP

Step 1: Assess governance

1. Open DIG directly or use **Switch to DIG** in DAM. Use **Switch to DAM** to return.
 2. Ask AI to assess metadata coverage, business semantic completeness, standards associations, lineage, and classification and security levels.
 3. Define a goal, business scope, acceptance criteria, and owner.
 4. Query candidate data sources, tables, fields, reports, or other objects and add confirmed objects to the governance inventory.
- The assessment proposes scope and priorities; it does not complete governance. Verify each object's source, system, schema, and permissions before adding it.

Step 2: Create the space

1. Create a space named for the business domain and goal. Document scope, owner, and acceptance criteria.
2. Add inventory objects as governance inputs.
3. Add requirements, standards, naming rules, business policies, metric definitions, and acceptance documentation.
4. Before Data Governance, verify sources, effective document revisions, and access scope.

Assets and documents provide context without automatically copying source data or replacing source system, DAM, or document permissions. See [space operations](#).

Step 3: Run governance Routines

Select an appropriate workflow on the right and specify objects and reference documents. Routines combine Sprites to enrich metadata, map standards, generate quality rules, classify data, or organize assets.

- **Run now:** suitable for initial validation and small, observed tasks. Start with a sample.
- **Scheduled execution:** configure frequency, scope, failure handling, and notification owner before enabling recurring runs.
- **Proactive governance:** monitor metadata changes and trigger configured tasks. A trigger does not automatically approve or publish proposals.

Check inputs, Routine revision, object count, and reviewer before execution. Distinguish successful, failed, skipped, and pending-review results afterward. See [Sprites and Routines](#) and [Routine operations](#).

Step 4: Track progress

Capability	Purpose	Verify
Space AI conversation	Ask about status, gaps, and next steps	Answers reference current objects and valid documents
Governance graph	Explore assets, standards, quality, security, and task relationships	Relationships reflect confirmed results
Asset profile	Summarize semantics, ownership, quality, and usage	Gaps are resolved in the appropriate DAM module
Business lineage	Understand sources, flows, and impact	Lineage has been harvested, parsed, and reviewed

Cross-check these views rather than relying on one progress percentage. Resolve missing objects, incomplete semantics, and unreliable relationships in the relevant DAM module before publishing.

Step 5: Review results and scorecards

Review execution, pending approvals, and governance scorecards against organizational targets. A scorecard summarizes completion and gaps; it is not a single verdict on business data quality.

1. Confirm that every inventory object has an outcome, including reasons for failure or skipping.
2. Complete required review and verify results in DAM.
3. Check that semantics, standards, lineage, quality, and security support the target use case.
4. Sample space AI answers. Verify references and that unsupported or out-of-scope questions are identified as such.
5. Assign remaining gaps to the next scope, owner, and acceptance date.

Proceed only after confirming results, context quality, and access scope. Execution success or a score threshold does not replace business acceptance.

Step 6: Publish knowledge for AI

In **Context Management**, organize confirmed semantics, standards, lineage, and reviewed documents, then open **Publish**.

1. Select context and exclude unrelated, expired, or unreviewed content.
2. Validate typical answers, citations, and access boundaries using the query agent.
3. Publish the space knowledge base as MCP for external AI applications, or provide a space Agent.
4. Configure consumers, authorization scope, and client connection details; record the revision.
5. Recheck context and published capabilities when objects, standards, permissions, or documents change.

Space MCP supplies governed semantics, knowledge, and relationships; it does not directly open underlying business data. Inbound space publishing and DIG's outbound MCP calls are separate configurations. See [context publishing](#).

Acceptance criteria

- Object sources, scope, assets, and documents are traceable.
- Executions, schedules, and change triggers have records; unresolved items have outcomes.
- Key results are reviewed and verified in DAM.
- Graphs, profiles, lineage, and scorecards reflect confirmed results.
- Published Agent or MCP tools contain only checked, authorized context.
- Owners and plans cover subsequent governance, validation, and republication.

Related guides

- [DIG overview](#)
- [Quick start](#)
- [Data intelligence spaces](#)
- [Governance capabilities](#)
- [Context publishing](#)
- [Approvals](#)

Governance capabilities

DIG interprets goals, calls configured DAM tools, and generates proposals for review. Available capabilities depend on license, agent, model, Skills, MCP, internal services, and user permissions.

DIG governance capabilities

1 Enrich metadata

Use context to suggest table and column names, descriptions, business tags, ownership, and data stewards.

2 Apply data standards

Use the data dictionary and industry standards to generate, map, and validate data standards.

3 Classify sensitive data

Use semantic models and knowledge to locate sensitive data and assign classifications and security levels.

4 Generate quality rules

Use profiling distributions to recommend business rules and technical rules for data quality.

5 Analyze downstream impact

Use the lineage knowledge graph to assess changes and warn about downstream effects.

6 Inventory data assets

Assess asset value, organize the catalog, and support natural language discovery of data assets.

Enrich metadata

Generate candidate Chinese business names and descriptions for harvested tables or fields with incomplete business metadata.

Prepare target sources, tables or fields, business domain and purpose, terminology and naming rules, and confirmed reference objects or documents.

1. Select an enrichment agent and target objects.
2. Specify attributes and naming constraints.
3. Check omissions and exceptions after generation.
4. Compare current and proposed values in approval, then edit, approve, or reject.

Ensure descriptions convey business meaning rather than restating physical names. Check that unrelated objects have not received the same generic description. Verify subsequent attribute status in metadata management.

Generate data standards

Generate candidate standards from domain semantics and enterprise rules. Candidates still require the standards publication workflow.

Prepare domain and scope, standard type, naming and coding rules, definitions, data types, lengths, value domains, and the existing catalog for duplicate checks.

1. Select standard generation and bound the objects.
2. Reference current policies or templates.
3. Generate candidates and inspect duplicates.
4. Review definitions, technical properties, and applicability.
5. Follow the data standards publication workflow after approval.

Check that definitions are actionable and that types, lengths, and value domains fit actual fields.

Generate quality rules

Generate candidate business and technical rules from structures and business requirements. Prepare tables, fields, schemas, validation requirements, quality dimensions, value domains, thresholds, null and uniqueness constraints, execution scope, and owners.

Verify business-to-technical rule associations, expressions against the target source, and evidence for thresholds. After approval, check rule configuration, tasks, and execution in Data Quality. A natural-language proposal is not an executed rule.

Classify data and assign security levels

Propose classifications and levels using enterprise taxonomies, level rules, and usage. Prepare current classification and level schemes, objects and use cases, regulations, policies, and exceptions.

1. Select data scope and the classification capability.
2. Specify the taxonomy and level scheme.
3. Generate proposals and prioritize sensitive or high-level objects.
4. Have the security owner review missed classifications, false positives, and level rationale.
5. Verify approved results in Data Security.

Map and validate standards

Standards mapping proposes associations between fields and applicable standards. Validation checks whether existing mappings remain appropriate.

Prepare current published standards, field business and physical names, descriptions and technical properties, scope, and exclusions. Review candidate standards, rationale, and confidence. Prioritize low-confidence proposals, incompatible types, retired standards, and fields lacking semantics. Verify actual mappings in Data Standards or Metadata Management after approval.

Inventory assets

Identify and organize candidate assets for cataloging, ownership confirmation, and publication. Prepare systems, domains, scope, enterprise asset definitions, catalog categories, owners, use cases, and value information.

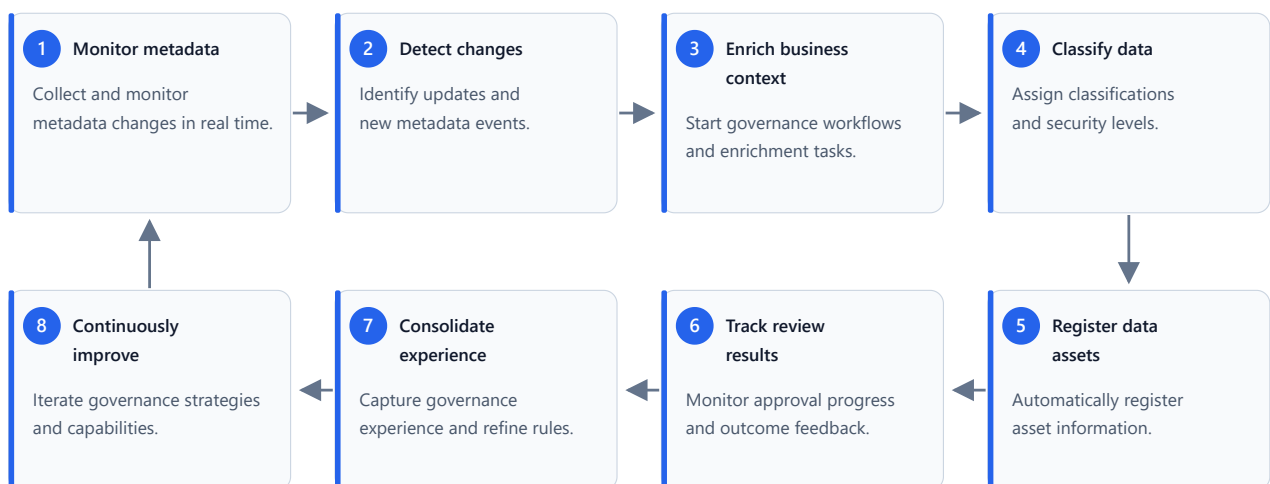
1. Select inventory scope and asset perspective.
2. Provide asset definitions and catalog rules.
3. Review candidates and missing information.
4. Have business and asset owners confirm ownership, names, value, and access scope.
5. Enter the formal catalog workflow after approval.

DIG does not determine ownership or publication scope on behalf of accountable owners.

Proactive governance

Resident background tasks detect metadata changes and initiate configured governance rather than waiting for individual requests. Proactive triggering does not authorize proposals to bypass review.

Proactive governance lifecycle



An administrator enables the resident task in Nacos. It listens for changes and runs business metadata enrichment, standards mapping, classification and levels, and asset organization sequentially, generating approval requests. Verify sources, agent settings, and reviewers first. Obtain the actual deployment setting rather than guessing its name in a conversation.

Match metadata changes to generated tasks, then inspect proposals and decisions. Investigate unexpected proposal volume before batch approval. Complete the cycle through discovery, proposal, review, target verification, and rule improvement.

Assist lineage analysis

Use available SQL, lineage, and metadata context to analyze parsing failures, explain operators, or summarize impact.

In a SQLink lineage space, hover over the status of a failed or partially successful SQL file. In the failure dialog, choose **Generate Preprocessing with AI** to open a conversation. Check the generated script added to the task, compare original and processed SQL, and parse again. Script generation does not establish correct parsing or lineage.

To explain an operator, click its magic wand. For impact analysis, specify the object and intended change, then compare the answer with actual SQLink nodes, relationships, and code.

Completion checks

- Actual objects match input scope.
- Referenced policies are current and applicable.
- Omissions, failures, and low-confidence proposals have outcomes.
- Reviewers have domain expertise and data permissions.
- Approved results are verified in the target module.
- Conversations, decisions, and results are traceable.

Troubleshooting

Symptom	Action
Proposals conflict with policy	Check documents, prompt, and agent configuration; reject incorrect proposals
Scope is too large or times out	Validate a smaller sample, then process batches
Many objects are unprocessed	Check permissions, metadata completeness, and tool errors
Similar errors recur	Summarize review feedback for prompt, Skill, or service improvements

Related guides

- [Quick start](#)
- [Approvals](#)
- [Administration](#)
- [SQLink lineage](#)

Set up Data Governance Agents

Administrators configure models, DAM connectivity, user identity, agents, and templates so users can start conversations and create spaces. Users with an available agent can go directly to the [quick start](#).

Configuration sequence

Order	Task	Verify
1	Configure a model and set the default	Connection test and sample response succeed
2	Configure global baseUr1	The agent reaches the correct DAM environment
3	Configure user identity	Calls use the current user's authorized identity
4	Import governance and asset query agents	Agents are listed with correct models and tools
5	Prepare saved prompts	Users can select prompts and replace example scope
6	Publish both template types	Governance and query templates are available to spaces

Configure a model

1. Obtain the protocol, base URL, model identifier, and credentials from the model service owner.
2. Open **Configuration Management > Model Configuration**.
3. Add the model using the service's parameters.
4. Test the connection and set the default model.
5. Verify responses with a small, non-sensitive sample before governance use.

The provider selects a communication protocol; do not infer it from the model brand alone. Enter keys in authentication fields. See [model services](#) for discovery, LLM and embedding parameters, and testing.

Set the DAM root URL

In **Configuration Management > Parameter Configuration**, add global parameter baseUr1 with the DAM root URL accessible to the agent.

```
Parameter: baseUrl  
Example value: https://dam.example.com
```

Replace the example with your confirmed environment address. Use the DAM root rather than the DIG page, a complete business page route, or another environment.

Configure user identity

Obtain a valid user token in DAM under **Personal Workspace > Token Management**. Configure the corresponding credentials in DIG's personal parameters under **Parameter Configuration**. Use the identity of the user executing the task.

When custom agents use official Skills, verify the runtime identity reference required by the Skill prompt: `${session.token}`. This literal reference is resolved at runtime; it is not a real token to copy into examples.

Static personal credentials and runtime session parameters have different purposes. If authentication fails, check token validity, user permissions, and whether the agent actually uses the parameter. Keep real tokens out of conversations and screenshots.

Import agents

1. Open **Configuration Management > Agents > Import Agent**.
2. Import the governance or asset query agent package supplied by your implementation team.
3. Give it a name that describes its responsibility.
4. Review system prompt, model, MCP, Skills, and dependent tools.
5. Set a default agent if required for embedded conversations.
6. Start a test session to verify object queries and proposal generation before user release.

Check imported addresses and paths for environment differences. Successful import alone does not validate dependencies.

Prepare prompts and templates

Administrators can import saved prompts for Chinese business name enrichment, classification and security levels, standard creation, standards mapping, rule creation, and asset inventory. Users must replace example scope and constraints before submitting prompts from **My Prompts**.

Follow [agent templates](#) to publish a Data Governance Agent template and a Business Query Agent template. An available agent does not mean a template has been published.

Verify setup

- DAM users can open AI and use the intended default or selected agent.
- The agent can query a test object within the user's permissions.
- A small governance proposal can be reviewed and verified in DAM.
- A published governance template is available when creating a space.
- An appropriate business query template is available for the query phase.
- User identity, files, and instance settings belong to the intended user and environment.

Diagnose failures

Check license and permissions, model connection, baseUr1, user identity, current agent, then tool and Skill dependencies. A successful model connection only validates the model service.

Configure model services

Model configurations are shared resources that agents can reuse. Configure and test the service first, then select it in an agent. A model response and a successful DAM governance tool call are separate acceptance checks.

LLM (large language model) configurations support text understanding and generation. **Text Embedding** configurations convert text into numeric vectors for tasks such as semantic retrieval; they do not generate conversation answers. Token limits count model text units, not characters. Vector dimensions specify the number of numeric values returned and must match the actual service.

Before you start

Obtain the protocol, API base URL, server model identifier, and credentials from the service owner. Confirm network access from DIG and whether the task needs a generative or embedding model. A web chat URL is not an API base URL.

Add a configuration

1. Open **Configuration Management > Model Configuration > Add Model**.
2. Enter a configuration name and choose model type and provider. The provider represents the protocol, not just the model brand.
3. Enter the base URL and an API key if required.
4. Allow automatic discovery or click **Discover Models**. Select the target model, or enter its exact server identifier if discovery is unavailable.
5. Set type-specific parameters and optionally make it the system default.
6. Click **Test Connection**. The current form requires a successful test before **Save**.
7. Verify the saved name, provider, and model identifier, then test it in the intended agent.

Use **Cancel** or **Close** to discard changes. Discovery does not replace a connection test or task validation.

Recommended LLM parameters

DIG governance results depend heavily on the underlying large language model (LLM). The following starting values are intended for data governance workloads and most production environments. Always follow the context window, output limit, and supported parameter range of the selected model service.

Parameter	Recommended value	Guidance
Model name	DeepSeek-V4-Pro, Kimi K2.6, GLM-5.1, Qwen3.6	Prefer a model with a long context window ($\geq 128K$) and strong Chinese-language support
Temperature	0.1–0.3	Governance tasks need stable, reproducible output; a high temperature adds randomness
Top P	0.9–1.0	Retain moderate vocabulary diversity without excessive truncation
Max Tokens	4096–8192	Keep long prompts and responses from being truncated, especially for batch asset governance
Frequency Penalty	0.0–0.2	Reduce repeated content; keep the value low
Presence Penalty	0.0–0.2	Encourage new topics; keep the value low
Stop Sequences	No configuration required	Usually leave this unset unless output must end at a specific sequence
System Prompt	See the system prompt example	Fix the agent role and output-format constraints

Hardware and deployment recommendations

For a locally deployed model, use the following as a starting point and load-test it against the model size, concurrency, and context length:

Resource	Recommended configuration
GPU	At least 1 × A100 (40 GB) or 2 × 4090 (24 GB)
Memory	≥ 32 GB
Storage	≥ 200 GB SSD for model cache and logs
Network	Low-latency access to the LLM, DAM database, and metadata services

Model selection guide

The models below are examples organized by governance workload. Confirm current availability, version, and service capabilities with the provider:

Workload	Suggested models	Rationale
Metadata completion and standard generation	Qwen 3.7, DeepSeek V4, Kimi 2.5, GLM 5.1	Strong Chinese semantic understanding and stable output
Lineage parsing and code interpretation	Claude 3.5 Sonnet	Strong code understanding for SQL and ETL explanations
Classification, grading, and sensitive-data identification	ERNIE 4.0	Built-in security and compliance semantics
Low-cost or test environments	GPT-3.5-Turbo, Qwen-Turbo	Controllable cost for non-critical tasks

These are general recommendations. Tune them for workload type, data volume, response time, and service cost, then validate the result with an approved small-scope governance task before wider use.

Model types

Type	Specific settings	Configuration guidance
LLM	Temperature, maximum tokens	Follow service limits and task requirements; initial form values are not universal recommendations
Text Embedding	Vector dimensions	Match the returned vector length; an LLM token limit is not a vector dimension

Recheck fields after changing type. Before replacing an embedding model or changing dimensions, assess dependent resources and any rebuild requirements.

Fields

Field	Meaning and verification
Configuration name	Identify purpose and service source
Model type	LLM or Text Embedding; controls specific fields
Provider	Protocol supported by the service
Base URL	Actual API root; verify path and reachability
API key	Optional in the form, but required when the service requires authentication
Discovered model name	Candidate returned by discovery; select the intended model
Manual model name	Exact server identifier when discovery is unavailable
Default model	System default model; separate from the default agent
Temperature	LLM sampling parameter within the service's supported range
Maximum tokens	LLM length parameter subject to service limits
Vector dimensions	Actual length of the embedding vector

Keep credentials in authentication fields rather than prompts, descriptions, screenshots, or feedback.

Edit, copy, or delete

The list provides **Edit**, **Copy**, and **Delete**. Check consumers before editing. Retest after changing address, model, or authentication. Rename copied configurations to distinguish their purpose and verify environment-specific values.

Before deletion, confirm that no agent or resource still needs the model. Validate shared changes on an approved small task before wider use.

Validate governance use

1. Verify the selected model in [agent management](#).
2. Start a test session with a known, authorized object and a bounded query or proposal task.
3. Inspect responses, tool calls, and errors.
4. Review proposals through [approvals](#) and verify DAM results.
5. If model testing passes but tool calls fail, check the DAM URL, identity, and dependencies in [initial setup](#).

Troubleshooting

Symptom	Check
Discovery returns no models	Base URL and network, protocol, discovery support, then manual identifier
Authentication fails	Whether an API key is needed, valid, and authorized for the target model
Cannot save	Connection test and form validation requirements
Embedding result mismatch	Actual vector length, configured dimensions, and model type
Conversation uses another model	Actual agent and session configuration; distinguish default model from default agent
Text works but governance has no result	Tools, baseUr1, identity parameters, and approval status

Orchestrate and run Routines

A Routine combines Sprites into a reusable governance workflow. It controls node order, input and output mappings, failure handling, and execution records. The same Sprite can be reused by multiple Routines without changing its responsibility.

Plan a Routine

Map business inputs, steps, and acceptance results before creating nodes. Each node invokes one Sprite and performs a clearly stated action. Independent read-only checks can run in parallel; data and write dependencies require sequential execution.

Element	Definition
Entry parameters	Objects, space, or filters supplied by the caller
Node	One Sprite invocation
Parameter mappings	Entry inputs or upstream outputs passed to a node
Dependencies	Nodes that must finish before execution
Failure policy	Continue or terminate after failure
Results	Structured output for callers or downstream nodes

Create and orchestrate

1. Open **Configuration Management > Governance Sprites > Routines** and create a Routine with a stable code, name, and description.
2. Define entry parameters with types, required status, and scope.
3. Add Sprite nodes with recognizable step names.
4. Select each Sprite and verify inputs and permission dependencies.
5. Map entry inputs or upstream outputs; check names, types, and null handling.
6. Connect nodes according to dependencies. Parallelize only independent nodes.
7. Configure failure policies. Critical writes and approval prerequisites generally terminate on failure; noncritical statistics may continue.
8. Save and test complete paths, branches, and errors.

Execution arrows do not map parameters. Inspect each node's input source, especially arrays, objects, and empty upstream results.

Test and publish

Test normal input, empty results, missing required inputs, unavailable tools, and partial node failure. Check order, actual parameters, and business outcomes. Define confirmation or approval boundaries before publishing Sprites that change metadata, standards, quality, security, or assets.

Avoid editing definitions in active use. If supported, copy or revise the configuration, regress it, then switch consumers. Templates, space instances, and schedules are maintained separately; template publication does not overwrite existing spaces.

Schedules and execution records

Routines can be called by spaces or agents, or run through deployment-specific scheduling. Confirm frequency, enabled state, and first run in the actual environment. Inspect:

- Entry inputs and final status.
- Node start and end times, inputs, outputs, and tool calls.
- Failed nodes and errors.
- Whether Continue policies let downstream nodes consume incomplete results.
- Expected approval requests or changes from write nodes.

Diagnose nodes

Symptom	Possible cause	Action
Routine does not start	Disabled schedule, empty inputs, or unpublished instance	Check instance, publication, and inputs
Node is skipped	Upstream termination or unmet condition	Inspect upstream state and branch conditions
Repeated node failure	Tool, permission, or input structure error	Run the Sprite separately and compare schemas
Successful node returns nothing	No matching objects or wrong output mapping	Inspect actual inputs and scope
Incomplete downstream results	Missing parallel aggregation or ignored failures	Add aggregation and explicit failure handling

Relationship to other objects

Agents interpret goals and choose capabilities. Sprites perform atomic tasks. Routines define order and mappings. Spaces hold business scope, documents, and phases. Templates retain reusable configurations. Names and permissions do not substitute for one another.

Related guides

- [Sprite design](#)
- [Sprites and Routines](#)
- [Agent templates](#)
- [Space governance](#)

Design and publish Sprites

A Sprite is a reusable atomic capability with one bounded responsibility, such as querying objects, enriching descriptions, checking standards associations, or generating proposals. Use a [Routine](#) for multiple steps, conditions, and result passing.

Plan the design

Area	Define
Task boundary	What the Sprite handles and excludes
Inputs	Object identifiers, space, scope, and required parameters
Outputs	Text, structured results, proposals, or pending changes
Tools	DAM internal services, MCP, commands, or Skills
Permissions	Read, write, and approval requirements
Failures	Responses for missing input, unavailable tools, and empty results

Avoid responsibilities as broad as governing an entire system; they make parameters, permissions, and acceptance difficult to define.

Create a Sprite

Open **Configuration Management > Governance Sprites**, select the agent, and create a Sprite. Enter code, name, description, and prompt using available fields. Keep the code stable; use the name for display and description for applicability and outcomes.

Input parameters

Define name, type, required status, default, and business meaning for each parameter. Prompt variables must match names. Keep object IDs, names, and display text distinct. For arrays and objects, define member structure and when empty values are allowed.

Prompt and boundaries

1. State the objective and scope.
2. Explain inputs and missing-input behavior.

- 3. Specify tools and call order.
- 4. Define output fields, format, and evidence.
- 5. Explain how to report conflicts, insufficient permission, or uncertainty.

Enforce access in tools. Do not describe a model inference as an already completed DAM operation.

Tool binding

Depending on deployment, bind DAM tools, MCP, internal services, commands, or Skills. Use only required tools. Check address, mode, headers, timeout, and parameter JSON. A successful connection still requires output validation against a real authorized object.

Test acceptance

Test a normal object, missing required input, empty results, and tool failure. Verify mappings, actual target objects, stable output fields, and absence of unintended writes. For standards, quality, security, or asset changes, generate proposals for human review by default.

Save, enable, and publish

Verify code, schema, tools, and prompt before saving. **Enabled** controls availability; **Expose as Tool** affects discovery by agents or Routines. Record configuration, test samples, and limitations before release. Avoid overwriting configurations in active use.

Some environments support revisions, copying, ZIP import, or generation from Skills. Review generated or imported content, dependencies, and permissions before acceptance.

Troubleshooting

Symptom	Check
Empty Sprite list	Selected agent and configuration permissions
Variables not substituted	Input names, prompt variables, and Routine mappings
Tool connection failure	Address, mode, headers, timeout, and server logs
Unstable output	Tighten schema, reduce unrelated tools, and define edge cases
Task succeeds but object is unchanged	Read-only capability and pending review

Related guides

- [Routine operations](#)
- [Sprites and Routines](#)
- [Administration](#)

Sprites and Routines

Screenshot note: Screenshots are illustrative. Controls or labels may remain untranslated in the demo release, and pages without a faithful English capture retain the source illustration. Control names and layouts can vary by release.

A **Governance Sprite** performs one atomic task, such as querying metadata or retrieving standards. A **Routine** combines Sprites using defined business logic. A Sprite is not itself a periodically running proactive governance task.

An **input schema** defines parameter names, types, and constraints; it is not a database schema. **Parameter mappings** pass Routine inputs or upstream outputs into downstream inputs. **Tools** perform actual reads and writes. Prompts and execution arrows alone do not establish that parameters or tools are configured.

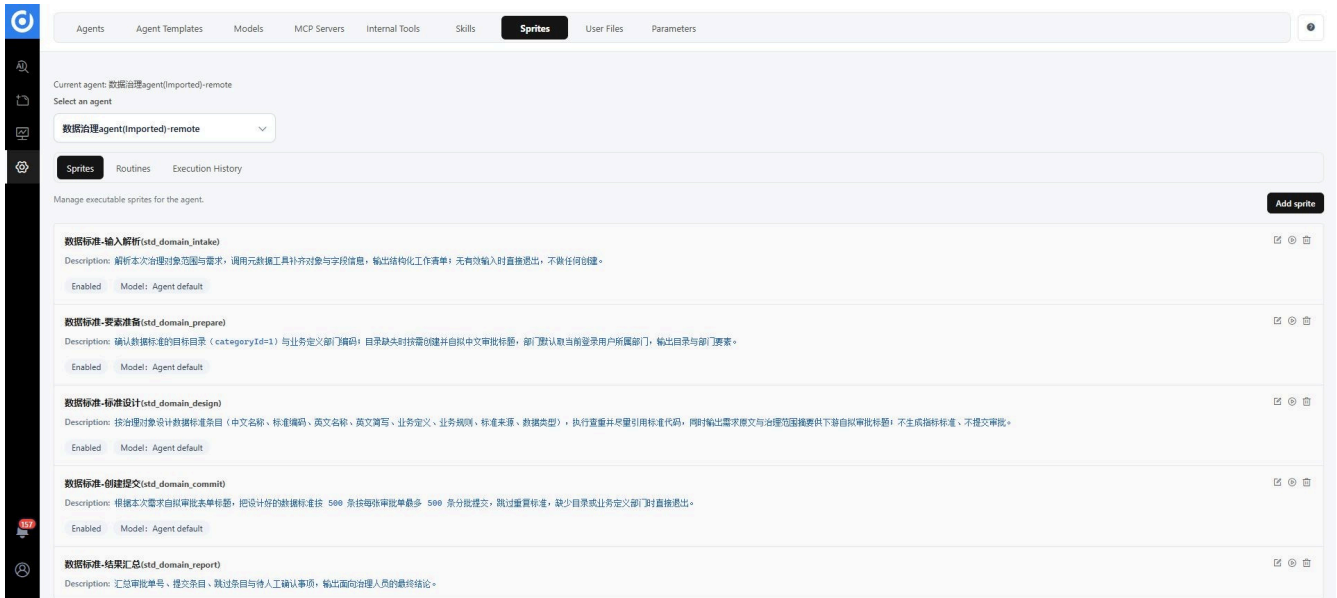
Open management

Open **Configuration Management > Governance Sprites** and select **Current Agent**. The page has **Sprites**, **Routines**, and **Execution History** sections. Resources belong to the selected agent. Verify selection before recreating a missing resource.

Design a Sprite

See [Sprite design and publication](#) for detailed inputs, tools, and release boundaries. Define a bounded task with explicit inputs and usable outputs.

Setting	Purpose	Verify
Code, name, description	Stable identity and task explanation	One clear responsibility
Template prompt	Goal, scope, output, and exceptions	Variables match input schema
Input schema	Names, types, and descriptions	Required fields, defaults, and scope are explicit
Current model	Understanding or transformation	Capability fits the task
MCP, internal service, or Command Line	Execution through protocol tools, interfaces, or commands	Minimal required tools, correct permissions and runtime
Enabled, exposed as a tool	Availability and invocation	Intended consumers only



[Open the full image](#). Commands and resources shown are examples; use validated tools in your environment.

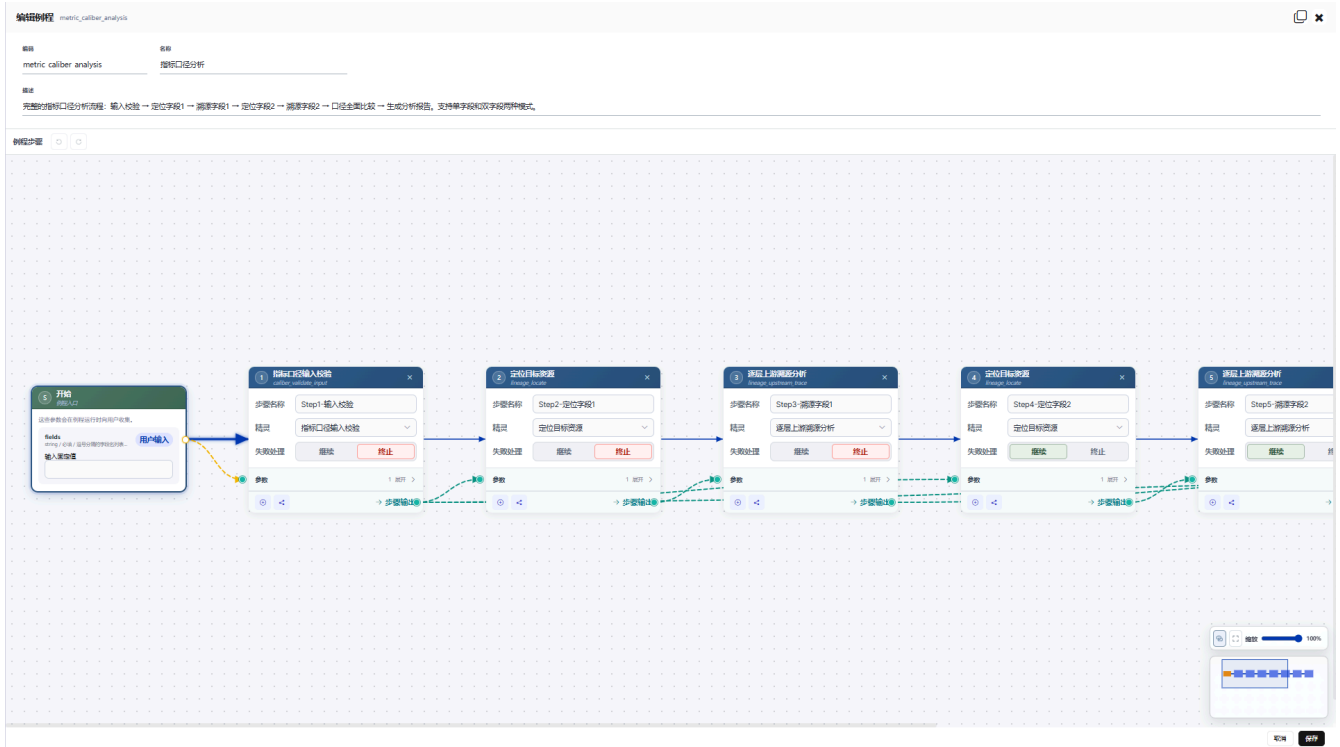
Create and test a Sprite

1. Create a Sprite and enter code, name, and description.
2. Write a prompt for one task, including how to report inability to complete it.
3. Define inputs. Placeholders such as `{{name}}` must match parameter names.
4. Select a model and bind tools.
5. Configure enabled status and tool exposure.
6. Test inputs, outputs, and tool results with a controlled sample, then save.

AI can derive Sprites from Skills, conversations, or processes. Review scope, parameters, and dependencies after generation.

Design a Routine

See [Routine orchestration](#) for inputs, mappings, failure handling, and diagnostics. Define inputs and outputs before arranging steps. For example, a read-only object check could locate metadata, inspect descriptions and standards associations, then summarize gaps. This is a design example, not a guaranteed built-in Routine.



[Open the full image](#). The metric analysis example illustrates the canvas; actual node names are configurable.

Orchestrate steps

1. Create a Routine with code, name, description, and entry parameters.
2. Add Sprite nodes with clear step names.
3. Select a Sprite for each node.
4. Map entry parameters or upstream outputs to node inputs.
5. Connect dependencies. Run dependent steps sequentially; parallelize only independent work.
6. Set failure handling to **Continue** or **Terminate**. Critical writes usually require termination on failure.
7. Use parameter relationship display, zoom, undo, and redo to inspect the flow.
8. Test the complete path, save, and follow the available publishing process.

Check every input's source. An execution arrow does not automatically map required parameters.

Schedules and templates

Schedules belong to an agent instance. Templates copy Sprite and Routine definitions but exclude schedules. Configure recurring runs on the target instance and verify the first trigger record.

Inspect execution history

Find the Sprite or Routine in **Execution History**. Drill into Routine nodes to inspect inputs, outputs, status, tool results, and errors.

Symptom	Check first
Empty history	Current agent and whether tasks have run
Missing parameters	Entry inputs, placeholder names, and mappings
Tool failure	Model, MCP, internal service, or command runtime
Unexecuted nodes	Upstream failure policy and dependencies
Successful flow with wrong business results	Scope, output semantics, and actual DAM records

Acceptance testing

Test normal input, missing required inputs, service failure, and partial node failure. Preserve human confirmation for writes and verify actual objects and properties in DAM. Execution success establishes completion of a call, not business correctness.

Related guides

- [Administration](#)
- [Agent templates](#)
- [Space governance](#)