



DII MCP

Data Infrastructure Insights

NetApp
October 22, 2026

This PDF was generated from https://docs.netapp.com/us-en/data-infrastructure-insights/mcp_overview.html on October 22, 2026. Always check docs.netapp.com for the latest.

Table of Contents

- DII MCP 1
 - DII MCP Server 1
 - What you can investigate 1
 - Choose an entry point 1
 - How DII MCP queries work 2
 - Important limitations 2
 - Next steps 2
 - DII MCP Server 2
 - What you can investigate 2
 - Choose an entry point 3
 - How DII MCP queries work 3
 - Important limitations 3
 - Next steps 4
 - DII MCP concepts and data model 4
 - Tenant scope 4
 - Acquisition units 4
 - Data sources 4
 - Objects 4
 - Logs 5
 - Monitors and alerts 5
 - Maintenance windows 5
 - Notification rules and targets 5
 - Metadata-first query flow 6
 - Filters 6
 - Time ranges and pagination 6
 - Set up the DII MCP Server 6
 - Before you begin 6
 - Configure the MCP client 7
 - Verify access 7
 - Credential safety 7
 - Next steps 7
 - Security and privacy 8
 - Least-privilege access 8
 - Authentication 8
 - Sensitive fields 8
 - Prompt and output hygiene 9
 - Tenant isolation 9
 - Audit and retention 9
 - Incident response 9
 - Related documentation 9
 - Troubleshoot DII MCP 10
 - The server does not connect 10
 - Authentication is required 10

An object query rejects a field	10
A log query rejects an attribute	10
An alert query rejects a field	11
The time range is invalid	11
A query returns too many results	11
A successful call returns no rows	11
An acquisition unit is connected but has never reported	11
A collector is failed	12
Webhook results include complete addresses	12
The AI assistant selected the wrong tool	12
DII MCP tool reference	12
Query conventions	12
Acquisition units	12
Alerts	13
Data collectors	14
Logs	14
Maintenance windows	15
Monitors	16
Notifications	16
Objects and metrics	17
Kubernetes collectors	18
Filter operators	18
Related documentation	19
Use Cases	19
DII MCP operational recipes	19
Identify capacity risk	20
Inventory environment and collection health	22
Check Kubernetes collector currency	24
Review notification posture	26
Troubleshoot performance	28
Assess storage health	31

DII MCP

DII MCP Server

Connect AI assistants/agents to Data Infrastructure Insights (DII) so they can investigate infrastructure inventory, telemetry, alerts, and collection health using natural language.



The DII MCP is a [Preview](#) feature and is therefore subject to change.

The DII MCP Server implements the Model Context Protocol (MCP) and exposes tenant-scoped tools to supported AI clients. The current product toolset is **read-only**: it retrieves and analyzes DII data but does not modify infrastructure or DII resources.

What you can investigate

- **Infrastructure inventory:** storage, virtualization, Kubernetes, ONTAP, Azure NetApp Files, StorageGRID, and related resources.
- **Performance and capacity:** object metrics, grouped aggregates, and time-series behavior.
- **Logs and events:** discover log types, search events, and build event-count histograms.
- **Alerting:** inspect active alerts, monitor definitions, severities, and corrective guidance.
- **Collection health:** review acquisition units, data sources, supported collector types, and Kubernetes collector versions.
- **Notification posture:** inspect maintenance windows, notification rules, and webhook target types.

Choose an entry point

Connect an AI client

Follow [Set up the DII MCP Server](#) for Cursor, Claude Code, VS Code, Codex, or another MCP-compatible client.

Browse available tools

Use the [tool reference](#) to understand all current tools, required discovery calls, query limits, and user-facing names.

Start with an operational workflow

Use a [recipe](#) when you already know the outcome you need:

- Inventory collection health
- Assess storage health
- Identify capacity risk
- Check Kubernetes collector currency
- Troubleshoot performance
- Review notification posture

How DII MCP queries work

Many DII resources have dynamic schemas. A reliable investigation follows a discovery-first sequence:

1. List the available object or log types.
2. Retrieve metadata for the selected type.
3. Use only valid attributes and metrics from that metadata.
4. Query a bounded time range.
5. Refine the result with filters, grouping, sorting, or pagination.

See [Concepts and data model](#) for details.

Important limitations

- All current product tools are read-only.
- Object, performance, and log queries enforce bounded time windows. Use windows of 30 days or less unless a tool page explicitly documents a smaller limit.
- Query fields must match the metadata returned for the selected resource type.
- Large results are paginated or capped.
- Tool availability and returned data depend on the tenant's collectors, enabled features, and user permissions.
- Do not request or reproduce complete webhook endpoint addresses. See [Security and privacy](#).

Next steps

1. [Connect your client](#).
2. Try the [environment inventory recipe](#).
3. Review [Troubleshooting](#) if a query is rejected or returns unexpected results.

DII MCP Server

Connect AI assistants/agents to Data Infrastructure Insights (DII) so they can investigate infrastructure inventory, telemetry, alerts, and collection health using natural language.



The DII MCP is a [Preview](#) feature and is therefore subject to change.

The DII MCP Server implements the Model Context Protocol (MCP) and exposes tenant-scoped tools to supported AI clients. The current product toolset is **read-only**: it retrieves and analyzes DII data but does not modify infrastructure or DII resources.

What you can investigate

- **Infrastructure inventory:** storage, virtualization, Kubernetes, ONTAP, Azure NetApp Files, StorageGRID, and related resources.
- **Performance and capacity:** object metrics, grouped aggregates, and time-series behavior.
- **Logs and events:** discover log types, search events, and build event-count histograms.

- **Alerting:** inspect active alerts, monitor definitions, severities, and corrective guidance.
- **Collection health:** review acquisition units, data sources, supported collector types, and Kubernetes collector versions.
- **Notification posture:** inspect maintenance windows, notification rules, and webhook target types.

Choose an entry point

Connect an AI client

Follow [Set up the DII MCP Server](#) for Cursor, Claude Code, VS Code, Codex, or another MCP-compatible client.

Browse available tools

Use the [tool reference](#) to understand all current tools, required discovery calls, query limits, and user-facing names.

Start with an operational workflow

Use a [recipe](#) when you already know the outcome you need:

- Inventory collection health
- Assess storage health
- Identify capacity risk
- Check Kubernetes collector currency
- Troubleshoot performance
- Review notification posture

How DII MCP queries work

Many DII resources have dynamic schemas. A reliable investigation follows a discovery-first sequence:

1. List the available object or log types.
2. Retrieve metadata for the selected type.
3. Use only valid attributes and metrics from that metadata.
4. Query a bounded time range.
5. Refine the result with filters, grouping, sorting, or pagination.

See [Concepts and data model](#) for details.

Important limitations

- All current product tools are read-only.
- Object, performance, and log queries enforce bounded time windows. Use windows of 30 days or less unless a tool page explicitly documents a smaller limit.
- Query fields must match the metadata returned for the selected resource type.
- Large results are paginated or capped.

- Tool availability and returned data depend on the tenant's collectors, enabled features, and user permissions.
- Do not request or reproduce complete webhook endpoint addresses. See [Security and privacy](#).

Next steps

1. [Connect your client](#).
2. Try the [environment inventory recipe](#).
3. Review [Troubleshooting](#) if a query is rejected or returns unexpected results.

DII MCP concepts and data model

Understanding these concepts helps an AI assistant select the right tools and construct valid queries.



The DII MCP is a [Preview](#) feature and is therefore subject to change.

Tenant scope

Every DII MCP request runs within the authenticated user's DII tenant and permissions. A tool can only return data that exists in that tenant and is visible to that identity.

Acquisition units

An acquisition unit (AU) is a remotely installed agent that polls data sources and reports telemetry to DII. AU records include connection status, reporting freshness, operating system, version, and the number of assigned data sources.

Connection and reporting are different signals. For example, an AU can accept a connection while having no successful reporting history. Assess both status and `lastReported`.

Data sources

A data source, also called a collector, is a configured connection to an infrastructure system such as a storage array, switch, hypervisor, or cloud service.

Use the list operation for inventory and status. Retrieve a specific data source when you need configuration details such as collection packages, poll intervals, or recent activity.

Supported data source types describe the broader DII collector catalog. They are not the same as collectors configured in the current tenant.

Objects

Objects are normalized infrastructure entities such as storage systems, pools, volumes, hosts, virtual machines, Kubernetes workloads, and product-specific resources.

Object types are dynamic. Always:

1. List object types.

2. Select the exact type.
3. Retrieve metadata for that type.
4. Query only attributes and metrics declared by the metadata.

Attributes and metrics

- **Attributes** describe identity, configuration, ownership, labels, or relationships.
- **Metrics** are time-varying measurements such as capacity, latency, throughput, or utilization.
- **Groupable attributes** can segment an aggregate query.

Metric filters are evaluated against raw metric values before requested time-window aggregation is applied.

Logs

Log types represent event streams such as ONTAP EMS, Kubernetes events, StorageGRID events, VMware events, and DII platform events.

Log querying is also metadata-first:

1. List available log types.
2. Retrieve metadata for the selected type.
3. Request only valid attributes.
4. Search events or build an event-count histogram.

Monitors and alerts

A monitor defines a condition DII evaluates. An alert is a specific monitor result for one or more related objects.

- Use monitor tools to inspect intent, status, object type, and corrective guidance.
- Use alert metadata before querying alerts.
- Use an alert name such as `AL-123456` or its UUID to retrieve one alert.

Alert status and severity are separate dimensions. Ask for both when prioritizing results.

Maintenance windows

A maintenance window suppresses alerts that match an expression during a defined period. An empty successful result means that no matching windows exist; it is not evidence of missing data.

Notification rules and targets

Notification rules route matching alerts. Targets can include Slack, Microsoft Teams, PagerDuty, or custom HTTP integrations.

Webhook addresses can contain secret material. For reporting, request only the target name, engine type, and status. Never reproduce the complete address.

Metadata-first query flow

```
Discover types
  → retrieve type metadata
    → select valid attributes and metrics
      → query a bounded time range
        → paginate, group, or retrieve time series
```

Filters

DII query tools support conditions and nested logical operators:

- Comparisons: EQ, NE, GT, LT, GE, LE
- Pattern matching: WILDCARD
- Presence: EXISTS
- Logic: and, or, not

Use the exact field name and compatible value type declared in metadata. For EXISTS, omit a comparison value when supported by the connected server version.

Time ranges and pagination

- Use ISO-8601 timestamps with explicit offsets or Z.
- Supply an IANA timezone such as America/New_York or UTC when required.
- Keep query windows to 30 days or less unless a tool documents a smaller limit.
- Object and alert queries generally use `limit` and `offset`.
- Log event queries use a returned cursor for the next page.

For errors and recovery steps, see [Troubleshooting](#).

Set up the DII MCP Server

Use this guide to connect an MCP-compatible AI client to DII.



The DII MCP is a [Preview](#) feature and is therefore subject to change.

Before you begin

You need:

- Access to a DII tenant.
- A DII user or service identity authorized to read the resources you intend to query.
- The remote MCP endpoint supplied for your DII environment.
- An MCP client that supports remote HTTP servers and your organization's authentication method.

The examples use <DII_HOST_URL> as a placeholder. Do not paste access tokens into source-controlled files. For example, the URL used by Cursor is:

```
https://<DII_HOST_URL>/api/v1/mcp
```

See the specific code snippets for your connection.

Configure the MCP client

1. Navigate to Admin > API Access and select the "MCP Configuration" tab.
2. Select the client you wish to configure.
3. Provide a meaningful name and an expiration, and select *Generate Token*.
4. **Copy the code snippet** You will only be able to copy this during this step. The token is hidden in the snippet but will paste correctly into your client app.
5. In the client settings, enter the correct DII Host URL in the "url" field.
6. Save the configuration.



Do not place bearer tokens in shell history, prompts, screenshots, project files, or source control.

Verify access

After the client reports that the server is connected, you can start sending requests like the following:

```
List the DII object types available to me.  
List the DII log types available to me.  
List my acquisition units without displaying IP addresses or UUIDs.
```

Successful discovery confirms connectivity, but it does not guarantee access to every underlying resource. Tenant features, collectors, and permissions determine what data is returned.

Credential safety

- Prefer interactive OAuth or another short-lived credential flow where available.
- Use a dedicated service identity for unattended clients.
- Do not put tokens in prompts, screenshots, examples, issue descriptions, or source control.
- Do not ask the agent to display complete webhook addresses.

For additional guidance, see [Security and privacy](#).

Next steps

- [Understand the DII data model](#)
- [Browse the tool reference](#)
- [Run the environment inventory recipe](#)

- [Troubleshoot connection and query errors](#)

Security and privacy

The DII MCP Server gives an AI client access to tenant infrastructure context. Treat that context with the same care as the DII user interface and APIs.



The DII MCP is a [Preview](#) feature and is therefore subject to change.

Least-privilege access

- Use an identity dedicated to the person, application, or automation connecting to MCP.
- Grant only the DII read permissions needed for the intended workflows.
- Review access when a user's role or operational responsibility changes.
- Use a separate service identity for unattended automation.

The current DII product toolset documented here is read-only. Read-only access can still disclose sensitive names, topology, configuration, telemetry, and operational state.

Authentication

Prefer interactive OAuth or another short-lived authentication flow supported by your deployment. If non-interactive credentials are required:

- Store them in an approved secret manager.
- Scope and rotate them according to organizational policy.
- Never place them in project files, shell history, prompts, screenshots, or documentation.

Sensitive fields

Potentially sensitive DII data includes:

- IP addresses, hostnames, tenant-specific resource names, and UUIDs
- Infrastructure topology and configuration
- Alert descriptions and log payloads
- User, owner, label, and annotation values
- Webhook endpoint addresses

Webhook URLs commonly embed routing keys, signatures, tokens, or unique paths. Do not ask an agent to return a complete webhook address. Request only:

- Target name
- Integration or engine type
- Enabled or delivery status

Applications integrating the MCP server should redact webhook addresses before displaying, storing, or forwarding tool results.

Prompt and output hygiene

- Use fictional identifiers in reusable prompts and examples.
- Ask the agent to omit IP addresses, UUIDs, endpoint addresses, and raw log payloads unless they are necessary.
- Summarize findings instead of copying large raw responses.
- Review generated content before placing it in tickets, chat channels, or external documents.
- Follow organizational data-handling requirements for the selected AI client.

Tenant isolation

MCP results are tenant-scoped to the authenticated DII context. Do not combine results from multiple tenants unless the workflow and authorization explicitly permit it.

Audit and retention

Audit-event coverage and retention depend on the DII deployment and connected AI client. Before production use, administrators should confirm:

- Whether MCP authentication and tool calls are logged
- Which request arguments and result metadata are retained
- Who can review those records
- How long the AI client retains prompts and tool results

Do not assume that MCP-specific audit events or retention guarantees exist unless they are documented for your deployment.

Incident response

If a credential or credential-bearing endpoint appears in an MCP result, prompt, transcript, or published document:

1. Stop sharing the affected content.
2. Revoke or rotate the credential or integration secret.
3. Remove the value from stored artifacts where possible.
4. Review access and audit records.
5. Report the exposure through your organization's security process.

Related documentation

- [Set up the DII MCP Server](#)
- [Concepts and data model](#)
- [Notification posture recipe](#)
- [Troubleshooting](#)

Troubleshoot DII MCP



The DII MCP is a [Preview](#) feature and is therefore subject to change.

The server does not connect

Check:

1. The configured URL matches the endpoint supplied for your DII environment.
2. The client supports remote HTTP MCP servers and the required authentication flow.
3. Authentication completed successfully.
4. Network policy allows access to the endpoint.
5. The connected identity can access the DII tenant.

Restart or reload the MCP server after changing client configuration.

Authentication is required

Complete the client's login flow, then retry the original request. If a service identity is used, verify that its credential is valid and stored through an approved secret mechanism.

Do not repeatedly retry a confirmed authorization denial. Ask a DII administrator to review the user's tenant and resource permissions.

An object query rejects a field

Object attributes and metrics vary by type.

1. Call `ObjectService_getObjectTypes`.
2. Confirm the exact object type.
3. Call `ObjectService_getMetadataForObjectType`.
4. Copy field names exactly from the metadata.
5. Verify that the value type and aggregation function are compatible.

Only attributes marked as groupable can be used to group objects.

A log query rejects an attribute

1. Call `LogService_getLogTypes`.
2. Confirm the exact log type.
3. Call `LogService_getLogTypeMetadata`.
4. Use only attributes returned for that log type.

Log event queries use cursor-based pagination. Pass the cursor returned by the previous page rather than an offset.

An alert query rejects a field

Call `AlertsService_getMetadata` immediately before constructing the query. Alert metadata can be broad and tenant-dependent; use exact field names.

The time range is invalid

- Ensure `fromTime` is earlier than or equal to `toTime`.
- Use ISO-8601 timestamps with an offset or `Z`.
- Keep the requested window below 30 days.
- Split longer investigations into consecutive, non-overlapping windows and summarize the results.
- Supply a valid IANA timezone when the tool requires one.

Runtime enforcement is authoritative if it is stricter than a client-cached tool description.

A query returns too many results

- Add a valid filter.
- Reduce the time range.
- Sort by the most relevant metric.
- Lower the result limit.
- Use grouping or a histogram when individual records are unnecessary.
- Continue with `offset` or a log cursor only when more detail is needed.

A successful call returns no rows

An empty result is often a valid answer. It can mean:

- No resources match the filter.
- No events occurred in the selected time range.
- No maintenance windows or notification rules are configured.
- The selected data type is not populated by the tenant's collectors.

Do not describe a successful empty result as missing data unless other evidence indicates a collection problem.

An acquisition unit is connected but has never reported

Connection state and successful reporting are separate. Check:

- `lastReported`
- Assigned data-source count
- Data-source acquisition status
- AU host connectivity and version
- Poll activity for an assigned data source

A collector is failed

Use `CollectorService_getDataSourceById` for the affected collector. Review recent poll activity, active patch information, package status, configuration, and the last successful acquisition time. Avoid displaying credentials or private configuration values.

Webhook results include complete addresses

Do not repeat the values. Summarize only target names, integration types, counts, and status. Follow [Security and privacy](#) if secret material was displayed or shared.

The AI assistant selected the wrong tool

Use a more explicit prompt:

First list the available object types. Retrieve metadata for the selected type. Then query only valid fields and explain which DII tools you used.

For common investigations, start from an [operational recipe](#).

DII MCP tool reference

The DII MCP Server currently exposes 26 read-only product tools across nine domains. A separate `mcp_auth` helper may appear when the connected client needs to authenticate.



The DII MCP is a [Preview](#) feature and is therefore subject to change.

Tool availability and results depend on the authenticated tenant, enabled DII features, configured collectors, and user permissions.

Query conventions

- Discover types and metadata before querying dynamic fields.
- Use exact, case-sensitive attribute and metric names returned by metadata.
- Use ISO-8601 timestamps and IANA timezone names where required.
- Keep query windows to 30 days or less.
- Use `limit` and `offset` for object and alert pagination.
- Use the returned cursor for log-event pagination.
- Do not display complete webhook endpoint addresses.

Acquisition units

List acquisition units

`AcquisitionUnitsService_getAcquisitionUnits`

Returns all acquisition units with connection status, reporting freshness, operating system, version, IP address,

and assigned data-source count.

Example:

List acquisition units. Summarize connected, failed, and never-reported units without displaying IP addresses or UUIDs.

Get an acquisition unit

AcquisitionUnitsService_getAcquisitionUnitByUuid

Retrieves full details for one acquisition unit. If the user supplies a name, list acquisition units first to resolve its UUID.

Required input: `uuid`

Alerts

Get alert metadata

AlertsService_getMetadata

Returns valid alert field names and data types. Call this before querying alerts.

Query alerts

AlertsService_queryForAlerts

Queries active or inactive alerts with selected attributes, filters, sorting, time bounds, and offset pagination.

Required inputs:

- `active`
- `attributeNames`
- `fromTime`
- `toTime`
- `timezone`

Optional inputs include `filter`, `sortField`, `sortAscending`, `limit`, and `offset`. `limit` is 1-500; `offset` is 0-1000. Keep the time window below 30 days.

Example:

Retrieve critical active storage alerts from the last seven days. Return alert ID, monitor name, severity, status, and related storage name.

Get an alert by name

AlertsService_getAlertByName

Retrieves one alert using an alert ID in the AL-123456 format.

Required input: `name`

Get an alert by UUID

AlertsService_getAlertByUUID

Retrieves one alert using its UUID.

Required input: `uuid`

Data collectors

List data sources

CollectorService_getDataSources

Lists configured collectors with name, status, vendor, model, acquisition unit, version, and last successful acquisition. Optionally filter by an AU UUID.

Example:

Group configured data sources by status and vendor. Highlight failures and stale successful acquisitions.

Get a data source

CollectorService_getDataSourceById

Retrieves one collector with configuration, collection-package status, poll activity, discovered devices, patch information, and acquisition-unit context.

Required input: `id`

Use this tool for poll intervals, configured metrics, collection settings, or failure investigation.

List supported data source types

CollectorService_getDataSourceTypes

Lists product types supported by the DII collector catalog. This is not a list of collectors configured in the tenant.

Logs

List log types

LogService_getLogTypes

Lists log types currently available in the tenant.

Get log type metadata

LogService_getLogTypeMetadata

Returns valid attributes for a log type. Call it before searching or grouping events.

Required input: `type`

Query log events

LogService_queryLogEvents

Returns log events in descending timestamp order.

Required inputs:

- `type`
- `fromTime`
- `toTime`
- `timezone`

Optional inputs include `attributes`, `filter`, `limit`, and `cursor`. `limit` is 1-200. The event window cannot exceed 30 days. Pass the previous response's `cursor` to retrieve the next page.

Count log events over time

LogService_queryLogEventsCountHistogram

Returns fixed-size time buckets of event counts. Optionally group each bucket by a valid groupable attribute; grouped results return the top segments and an `__other__` segment.

Required inputs:

- `type`
- `fromTime`
- `toTime`
- `timezone`

Optional inputs: `filter`, `groupBy`

Maintenance windows

List maintenance windows

MaintenanceWindowService_getMaintenanceWindows

Lists active, non-expired alert-suppression windows. Set `includeExpired` to `true` to include expired windows.

A successful empty response means no matching maintenance windows exist.

Get a maintenance window

MaintenanceWindowService_getMaintenanceWindowById

Retrieves one maintenance window.

Required input: `id`

Monitors

List monitors

MonitorsService_queryMonitors

Lists DII monitors. Use `isSystem` to return only system monitors, exclude system monitors, or return all monitors.

Get a monitor

MonitorsService_getMonitorById

Retrieves one monitor, including its description and corrective guidance when available.

Required input: `id`

Notifications

List notification rules

NotificationService_getNotificationRules

Lists rules that route matching alerts to notification targets. A successful empty response means no matching rules exist.

Get a notification rule

NotificationService_getNotificationRuleById

Retrieves one routing rule.

Required input: `id`

List webhook targets

NotificationService_getWebhooks

Lists webhook target metadata for Slack, Microsoft Teams, Discord, PagerDuty, and custom HTTP integrations. Optionally filter by comma-separated engine types.

Security requirement: complete endpoint addresses can contain credentials. Applications and agents must omit or redact the address field from displayed or forwarded results.

Safe example:

Count webhook targets by engine type and list only target names and status. Do not return endpoint addresses.

Objects and metrics

List object types

`ObjectService_getObjectTypes`

Lists object types currently available in the tenant.

Get object metadata

`ObjectService_getMetadataForObjectType`

Returns valid attributes, metrics, types, and grouping capabilities for one object type.

Required input: `objectType`

Query objects

`ObjectService_query`

Retrieves objects, selected attributes, and optional metrics. Filters referencing metrics apply to raw values before time-window aggregation.

Required inputs:

- `objectType`
- `attributeNames`

Optional inputs include `metricNames`, `filter`, `sortFieldName`, `isSortAscending`, `fromTime`, `toTime`, `aggregateFunctions`, `limit`, and `offset`.

Supported query aggregations: MAX, MIN, AVG, SUM, LAST. `limit` is 1-500. If time bounds are omitted, the tool defaults to a recent three-hour window.

Group objects

`ObjectService_groupObjects`

Groups objects by one valid groupable attribute and aggregates selected metrics. Filters apply to raw values before aggregation.

Required inputs:

- `objectType`
- `groupByAttribute`

- `metricNames`

Optional inputs include `aggregateFunctions`, `filter`, `sorting`, `time bounds`, `limit`, and `offset`.

Supported group aggregations: `MAX`, `MIN`, `AVG`, `SUM`, `FIRST`, `CHANGE`, `CHANGE_RATIO`.

Query performance time series

`ObjectService_queryPerformanceMetricsForObjects`

Returns bucketed time-series data for one metric and matching objects.

Required inputs:

- `objectType`
- `metricName`
- `fromTime`
- `toTime`
- `timezone`

Optional inputs include `filter`, `aggregationFunction`, `limit`, and `isSortAscending`. `limit` controls the number of returned series and must be 1-50.

Supported bucket aggregations: `MAX`, `MIN`, `AVG`, `SUM`, `FIRST`, `CHANGE`, `CHANGE_RATIO`.

Kubernetes collectors

List Kubernetes collectors

`OperatorsManagement_getKubernetesCollectors`

Returns cluster name, overall currency status, operator version, Network Performance and Map component version, Change Analysis component version, and last-reported timestamp.

Example:

Identify outdated Kubernetes collectors and summarize which components need attention.

Filter operators

Alert, log, and object queries support nested conditions:

- `EQ`, `NE`
- `GT`, `LT`, `GE`, `LE`
- `WILDCARD` using `*` and `?`
- `EXISTS`
- Nested `and`, `or`, and `not`

The connected server validates fields against metadata. If a query is rejected, refresh the relevant metadata and reconstruct the filter.

Related documentation

- [DII MCP overview](#)
- [Concepts and data model](#)
- [Operational recipes](#)
- [Troubleshooting](#)

Use Cases

DII MCP operational recipes

Recipes combine multiple read-only DII tools into repeatable infrastructure workflows. Use them as starting points and replace fictional resource names with values from your tenant.



The DII MCP is a [Preview](#) feature and is therefore subject to change.

Choose a workflow

Goal	Recipe	Primary DII surfaces
Validate collection coverage and freshness	Inventory environment and collection health	Acquisition units, data sources
Prioritize active infrastructure issues	Assess storage health	Alerts, monitors, logs, objects
Find pools approaching capacity pressure	Identify capacity risk	Object metadata, grouping, time series
Check Kubernetes collection component versions	Check Kubernetes collector currency	Kubernetes collectors
Investigate a slow volume or workload	Troubleshoot performance	Object metrics, alerts, logs
Review routing and suppression safely	Review notification posture	Rules, targets, maintenance windows

Standard investigation pattern

1. Clarify the resource scope and time range.
2. Discover available types.
3. Retrieve metadata before using dynamic fields.
4. Query a bounded time window.
5. Correlate independent signals.
6. Distinguish observed facts from inferences.

7. Report limitations and recommended follow-up.

Safety

- These recipes use read-only tools.
- Keep time windows to 30 days or less.
- Omit credentials, private addresses, UUIDs, raw webhook endpoints, and unnecessary log payloads.
- Confirm high-impact operational actions outside the MCP workflow.

Related documentation

- [DII MCP overview](#)
- [Concepts and data model](#)
- [Tool reference](#)
- [Security and privacy](#)

Identify capacity risk

Use this recipe to identify storage pools or equivalent capacity domains approaching pressure and to find potential rebalance candidates.



The DII MCP is a [Preview](#) feature and is therefore subject to change.

Prerequisites

- A storage scope or permission to review all visible storage pools.
- A lookback window of 30 days or less.
- Agreement on organizational warning and critical thresholds.

Tools used

- `ObjectService_getObjectTypes`
- `ObjectService_getMetadataForObjectType`
- `ObjectService_query`
- `ObjectService_groupObjects`
- `ObjectService_queryPerformanceMetricsForObjects`
- `AlertsService_getMetadata`
- `AlertsService_queryForAlerts`

Starter prompt

Assess storage capacity risk over the last 30 days. Discover the correct storage-pool object type and metadata, then rank pools using available used-capacity ratio, free capacity, growth, and time-to-full signals. Identify critical, warning, and watch-list pools, and suggest evidence-based rebalance candidates. State the thresholds and aggregation functions used.

Agent workflow

1. List object types and select the appropriate normalized or product-specific pool type.
2. Retrieve metadata.
3. Identify valid:
 - Pool and storage-system names
 - Total, used, and free capacity metrics
 - Used-ratio metric
 - Time-to-full or forecast metric
 - Groupable storage, tier, location, or service-level attributes
4. Query a current ranked snapshot using compatible aggregations.
5. Retrieve time series for growth-sensitive metrics over the requested lookback.
6. Where supported, group capacity by storage system, tier, or location.
7. Query active capacity-related alerts for corroboration.
8. Classify risk using the agreed thresholds.
9. Identify potential rebalance sources and destinations based on both capacity and operational constraints.

Example risk model

Use tenant policy when available. Otherwise, label the following as an illustrative model:

- **Critical:** used ratio at or above 90%, or time to full under 30 days
- **Warning:** used ratio at or above 80%, or time to full under 90 days
- **Watch:** sustained positive growth with time to full under 180 days
- **Lower risk:** none of the above conditions

Do not infer time to full from only two samples when DII supplies a forecast metric. If no forecast is available, describe the trend without presenting a precise exhaustion date.

Rebalance guidance

A destination is not suitable solely because it has free capacity. Consider:

- Storage system and protocol compatibility
- Service level and performance headroom
- Data-protection and replication relationships
- Workload placement constraints
- Thin provisioning and overcommitment
- Operational change windows

Suggested output

1. Capacity-risk summary
2. Critical and warning pools

3. Growth narrative
4. Active capacity alerts
5. Candidate source and destination pairs
6. Assumptions and missing data

Limits and privacy

- Object windows cannot exceed 30 days per call. Use consecutive non-overlapping windows for longer trends.
- Filters apply to raw metric values before aggregation.
- Only group by attributes marked groupable in metadata.
- Capacity values can use different units; normalize before comparing.

Follow-up prompts

Show the 30-day used-capacity trend for the five highest-risk pools.

Group pool capacity by storage system and identify systems with both a critical pool and a viable lower-risk destination.

Correlate the highest-risk pools with active capacity monitors and alerts.

[Back to recipes](#)

Inventory environment and collection health

Use this recipe to understand whether DII's acquisition units and configured data sources are connected, reporting, and collecting successfully.



The DII MCP is a [Preview](#) feature and is therefore subject to change.

Prerequisites

- Access to acquisition-unit and collector tools.
- An optional acquisition-unit or collector name to narrow the investigation.

Tools used

- AcquisitionUnitsService_getAcquisitionUnits
- AcquisitionUnitsService_getAcquisitionUnitByUuid
- CollectorService_getDataSources
- CollectorService_getDataSourceById

- `CollectorService_getDataSourceTypes`

Starter prompt

Review my DII collection environment. Summarize acquisition units by connection status and reporting freshness, then summarize configured data sources by acquisition status and vendor. Highlight acquisition units that never reported, units with no assigned collectors, failed collectors, and stale successful acquisitions. Do not display IP addresses, UUIDs, or credentials.

Agent workflow

1. List all acquisition units.
2. Classify each AU using both connection status and `lastReported`.
3. Count assigned data sources per AU.
4. List all configured data sources.
5. Group collectors by status, vendor, model, and AU.
6. Identify:
 - Disconnected or heartbeat-failed AUs
 - Connected AUs that never reported
 - AUs with zero data sources
 - Failed collectors
 - Collectors with no successful acquisition
 - Successful collectors whose last acquisition is unexpectedly old
7. Retrieve full AU or collector details only for the exceptions.
8. Return a prioritized summary and clearly state the freshness threshold used.

Interpretation

Use these distinctions:

- **Connected and reporting:** the AU is connected and has a recent `lastReported` value.
- **Connected but never reported:** connectivity exists, but successful telemetry reporting is not established.
- **Heartbeat failed:** investigate the AU host, network path, and service.
- **Collector failed:** inspect the collector's recent poll activity, package status, configuration, and last successful acquisition.
- **No assigned collectors:** this can be intentional; verify before treating it as a fault.

The supported data source type catalog describes products DII can collect. It does not prove that those products are configured in the tenant.

Suggested output

1. Executive summary
2. Acquisition-unit status counts

3. Collector status counts
4. Exceptions ranked by operational impact
5. Recommended checks
6. Data limitations

Limits and privacy

- Avoid returning AU UUIDs, IP addresses, private hostnames, or collector credentials.
- A status value alone does not establish data freshness.
- Retrieve detailed configuration only for named exceptions.

Follow-up prompts

Inspect the failed collector named `example-array`. Summarize recent poll activity and collection-package failures without showing credentials or private addresses.

Which acquisition units have not reported within the last 30 minutes, and which collectors depend on them?

Compare configured collector vendors with the DII supported data source type catalog.

[Back to recipes](#)

Check Kubernetes collector currency

Use this recipe to identify Kubernetes clusters whose DII operator or collection components are outdated, disabled, or stale.



The DII MCP is a [Preview](#) feature and is therefore subject to change.

Prerequisites

- Kubernetes collection is configured in the DII tenant.
- Access to Kubernetes collector information.

Tools used

- `OperatorsManagement_getKubernetesCollectors`
- Optional: `AcquisitionUnitsService_getAcquisitionUnits`
- Optional: `CollectorService_getDataSources`

Starter prompt

Review DII Kubernetes collector currency. Group clusters by overall status, identify outdated or disabled operators and components, and flag stale last-reported timestamps. Do not display internal addresses or UUIDs.

Agent workflow

1. List Kubernetes collectors.
2. Normalize each cluster into:
 - Up to date
 - Outdated
 - Disabled
 - Unknown
3. Compare:
 - Operator version
 - Network Performance and Map component version
 - Change Analysis component version
4. Convert the last-reported timestamp into the requested timezone.
5. Flag clusters that have not reported within the agreed freshness threshold.
6. Distinguish N/A from an outdated version:
 - N/A can mean the component is not installed, not enabled, or not applicable.
7. If reporting is stale, inspect related acquisition-unit and collector status where available.
8. Return a prioritized upgrade and investigation list.

Interpretation

- **Outdated:** at least one installed component is behind the DII-expected version.
- **Disabled:** collection is intentionally or administratively disabled.
- **Stale reporting:** version data may not reflect the cluster's current state.
- **N/A:** requires context; do not automatically classify it as a failure.

The tool reports versions and status. It does not perform upgrades.

Suggested output

1. Cluster counts by status
2. Outdated clusters ranked by reporting freshness
3. Component-level version differences
4. Disabled or unavailable components
5. Recommended verification steps

Limits and privacy

- Cluster names can be sensitive; use approved handling when sharing results.
- Do not infer vulnerability exposure from version currency alone.
- Validate upgrade compatibility and process outside this read-only workflow.

Follow-up prompts

Which clusters have an outdated operator and have not reported in the last hour?

Separate clusters with an outdated Change Analysis component from clusters where that component is unavailable.

For stale clusters, check whether the associated DII collection infrastructure is healthy.

[Back to recipes](#)

Review notification posture

Use this recipe to understand how DII alerts are routed and suppressed without exposing webhook endpoint addresses.



The DII MCP is a [Preview](#) feature and is therefore subject to change.

Prerequisites

- Access to notification and maintenance-window tools.
- An optional monitor, resource, or integration type to narrow the review.

Tools used

- NotificationService_getNotificationRules
- NotificationService_getNotificationRuleById
- NotificationService_getWebhooks
- MaintenanceWindowService_getMaintenanceWindows
- MaintenanceWindowService_getMaintenanceWindowById
- Optional: MonitorsService_queryMonitors

Starter prompt

Review DII notification posture. Summarize active and inactive notification rules, the monitor or resource scope of each rule, webhook target counts by integration type, and current maintenance windows. Return

only target names, engine types, and status—never endpoint addresses.

Agent workflow

1. List notification rules.
2. Classify rules as active, inactive, expiring, or without expiry.
3. Summarize the expression scope of each rule:
 - Monitor name or group
 - Related object
 - Broad wildcard scope
4. Retrieve a specific rule only when additional detail is needed.
5. List webhook targets and immediately discard the address field from the working result.
6. Group targets by engine type and status.
7. List active maintenance windows.
8. Identify suppression windows that overlap important resources or unusually long periods.
9. Optionally compare rule expressions with available monitors.
10. Report routing coverage, broad rules, inactive rules, and suppression risks.

Security requirement

Complete webhook addresses can contain:

- Slack or Teams signing paths
- PagerDuty routing keys
- API tokens
- Unique workflow signatures
- Custom integration credentials

Never display, quote, summarize, persist, or forward the address field. A masked suffix is also unnecessary for this workflow.

If a complete endpoint appears in a transcript or shared artifact, follow the incident steps in [Security and privacy](#).

Interpretation

- An inactive rule may be intentional; confirm ownership before recommending removal.
- A broad wildcard expression can create notification noise or unintended routing.
- A target status may be null or unknown; do not report it as enabled.
- A successful empty maintenance-window result means no matching windows exist.
- A long-lived window can hide real issues even when it is technically active and valid.

Suggested output

1. Rule counts by status
2. Rule scope and destination type
3. Target counts by engine
4. Active maintenance windows
5. Broad, inactive, or ambiguous configurations
6. Security and data-quality caveats

Limits and privacy

- Do not return addresses, tokens, UUIDs, or private resource identifiers.
- This workflow reviews configuration but does not test message delivery.
- The current tools do not create, update, disable, or delete rules and targets.

Follow-up prompts

Which active notification rules have broad wildcard expressions, and which monitor groups can they match?

List current and upcoming maintenance windows and explain which alert scopes they suppress.

Count notification targets by integration type and identify targets with unknown status. Do not display addresses.

[Back to recipes](#)

Troubleshoot performance

Use this recipe to investigate a slow volume, datastore, virtual machine, Kubernetes workload, or other DII object by correlating performance metrics, alerts, and platform logs.



The DII MCP is a [Preview](#) feature and is therefore subject to change.

Prerequisites

- The affected resource name and approximate incident time.
- A time range of 30 days or less.
- A comparison period or known healthy baseline when possible.

Tools used

- `ObjectService_getObjectTypes`

- ObjectService_getMetadataForObjectTypes
- ObjectService_query
- ObjectService_queryPerformanceMetricsForObjects
- ObjectService_groupObjects
- AlertsService_getMetadata
- AlertsService_queryForAlerts
- LogService_getLogTypes
- LogService_getLogTypeMetadata
- LogService_queryLogEvents

Starter prompt

Investigate why the volume `example-volume` was slow between 09:00 and 11:00 UTC. Discover the correct object type and valid metrics, compare latency, IOPS, and throughput over the incident window, inspect related active alerts and platform logs, and compare with a healthy period. Separate evidence from hypotheses and identify the next measurement needed.

Agent workflow

1. Resolve the resource to an exact object type and identity.
2. Retrieve metadata for that object type.
3. Identify valid latency, operations, throughput, utilization, and capacity metrics.
4. Query current object attributes to establish placement and relationships.
5. Retrieve time series over:
 - The incident window
 - A comparable healthy window
6. Align timestamps and aggregation functions across metrics.
7. Retrieve alerts related to the resource, containing system, host, node, or storage pool.
8. Select relevant log streams, retrieve metadata, and search a narrow interval around metric changes.
9. Test plausible hypotheses:
 - Demand spike
 - Resource saturation
 - Storage or fabric errors
 - Capacity pressure
 - Host or VM contention
 - Collection gap
10. Report the most supported explanation and unresolved alternatives.

Evidence standards

Label findings:

- **Observed:** directly returned by a tool.
- **Correlated:** independent signals changed in the same interval.
- **Inferred:** a plausible explanation not yet proven.
- **Unknown:** required evidence is unavailable.

Correlation is not causation. A simultaneous alert should be treated as supporting evidence, not proof, unless the mechanism is established.

Metric guidance

- Use `AVG` for broad behavior, but inspect `MAX` when short spikes matter.
- Use `SUM` only for additive metrics.
- Use `CHANGE` or `CHANGE_RATIO` for counters or growth where metadata supports it.
- Avoid comparing series with different bucket boundaries or units.
- Filters apply before time-bucket aggregation.

Suggested output

1. Incident scope and timeline
2. Metric evidence
3. Alert and log evidence
4. Healthy-period comparison
5. Most likely explanation
6. Alternative hypotheses
7. Confidence and missing evidence
8. Recommended next check

Limits and privacy

- Time-series windows cannot exceed 30 days.
- The performance tool returns a limited number of series; narrow the filter before increasing the limit.
- Omit sensitive raw log fields that do not contribute to the diagnosis.
- Do not propose disruptive remediation without validating it through normal operational controls.

Follow-up prompts

Compare maximum and average latency during the incident to the same hour on the previous healthy day.

Group related workloads by host or storage pool and identify whether the slowdown was isolated or

shared.

Show the platform events within 15 minutes of the first latency increase.

[Back to recipes](#)

Assess storage health

Use this recipe to prioritize storage issues by combining active alerts, monitor intent, corrective guidance, relevant platform logs, and object context.



The DII MCP is a [Preview](#) feature and is therefore subject to change.

Prerequisites

- A time range of 30 days or less.
- Optional storage-system, cluster, pool, or volume scope.
- Access to alert, monitor, log, and object tools.

Tools used

- AlertsService_getMetadata
- AlertsService_queryForAlerts
- AlertsService_getAlertByName
- MonitorsService_getMonitorById
- LogService_getLogTypes
- LogService_getLogTypeMetadata
- LogService_queryLogEvents
- ObjectService_getObjectTypes
- ObjectService_getMetadataForObjectType
- ObjectService_query

Starter prompt

Assess storage health for the last 24 hours. Identify critical and warning active alerts for storage systems, nodes, pools, and volumes. For the highest-impact alerts, explain the monitor condition and corrective guidance, then inspect relevant EMS or storage-platform logs around the trigger time. Separate confirmed evidence from likely causes.

Agent workflow

1. Get alert metadata.
2. Identify valid fields for severity, status, monitor, related object, and trigger time.

3. Query active storage-related alerts in the requested window.
4. Rank alerts using:
 - Severity
 - Affected object type
 - Number of affected objects
 - Recency and recurrence
 - Availability, data-protection, performance, or capacity impact
5. Retrieve full details for the highest-priority alerts.
6. Retrieve their monitor definitions and corrective guidance.
7. List log types and select the platform-native stream relevant to each alert, such as ONTAP EMS or StorageGRID events.
8. Retrieve log metadata, then query a narrow interval around the alert.
9. Retrieve object attributes or metrics needed to confirm current state.
10. Produce a health summary with evidence, likely cause, impact, and next check.

Interpretation

Do not treat one signal as a complete health assessment:

- **Alerts** show detected conditions.
- **Monitors** explain the detection rule and intended corrective action.
- **Logs** provide platform-native event context.
- **Objects and metrics** show current inventory or measured state.

An active alert can remain open after the underlying condition changes. Use fresh object and metric data where possible.

Suggested output

For each priority issue:

- Affected resource
- Severity and alert state
- Trigger time
- Monitor and condition
- Corroborating logs or metrics
- Confirmed evidence
- Likely cause, labeled as an inference
- Corrective guidance
- Confidence and missing evidence

Limits and privacy

- Keep alert and log windows below 30 days.
- Use narrow log windows to reduce unrelated events.
- Omit unnecessary UUIDs, addresses, annotations, and raw payloads.
- Corrective guidance is informational; validate commands against the relevant product documentation and change process.

Follow-up prompts

For alert `AL-123456`, show the monitor guidance and related EMS events within 30 minutes before and after the trigger.

Group active storage alerts by affected system and impact area. Which system needs attention first, and why?

Check whether the highest-priority alert is still supported by current object metrics.

[Back to recipes](#)

Copyright information

Copyright © 2026 NetApp, Inc. All Rights Reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

LIMITED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data -Noncommercial Items at DFARS 252.227-7013 (FEB 2014) and FAR 52.227-19 (DEC 2007).

Data contained herein pertains to a commercial product and/or commercial service (as defined in FAR 2.101) and is proprietary to NetApp, Inc. All NetApp technical data and computer software provided under this Agreement is commercial in nature and developed solely at private expense. The U.S. Government has a non-exclusive, non-transferrable, nonsublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b) (FEB 2014).

Trademark information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.