Astra REST implementation

Astra Automation

NetApp
August 11, 2025

Table of Contents

Astra REST implementation

Core design

REST web services

Representational State Transfer (REST) is a style for creating distributed web applications. When applied to the design of a web services API, it establishes a set of mainstream technologies and best practices for exposing server-based resources and managing their states. While REST provides a consistent foundation for application development, the details of each API can vary based on the specific design choices. You should be aware of the characteristics of the Astra Control REST API before using it with a live deployment.

Resources and state representation

Resources are the basic components of a web-based system. When creating a REST web services application, early design tasks include:

- Identification of system or server-based resources

  Every system uses and maintains resources. A resource can be a file, business transaction, process, or administrative entity. One of the first tasks in designing an application based on REST web services is to identify the resources.

- Definition of resource states and associated state operations

  Resources are always in one of a finite number of states. The states, as well as the associated operations used to affect the state changes, must be clearly defined.

URI endpoints

Every REST resource must be defined and made available using a well-defined addressing scheme. The endpoints where the resources are located and identified use a Uniform Resource Identifier (URI). The URI provides a general framework for creating a unique name for each resource in the network. The Uniform Resource Locator (URL) is a type of URI used with web services to identify and access resources. Resources are typically exposed in a hierarchical structure similar to a file directory.

HTTP messages

Hypertext Transfer Protocol (HTTP) is the protocol used by the web services client and server to exchange request and response messages about the resources. As part of designing a web services application, HTTP methods are mapped to the resources and corresponding state management actions. HTTP is stateless. Therefore, to associate a set of related requests and responses as part of one transaction, additional information must be included in the HTTP headers carried with the request and response data flows.

JSON formatting

While information can be structured and transferred between a web services client and server in several ways, the most popular option is JavaScript Object Notation (JSON). JSON is an industry standard for representing simple data structures in plain text and is used to transfer state information describing the resources. The Astra

Control REST API uses JSON to format the data carried in the body of each HTTP request and response.

Resources and collections

The Astra Control REST API provides access to resource instances and collections of resource instances.

Conceptually a REST **resource** is similar to an **object** as defined with the object-oriented programming (OOP) languages and systems. Sometimes these terms are used interchangeably. But in general, "resource" is preferred when used in the context of the external REST API while "object" is used for the corresponding stateful instance data stored at the server.

Attributes of the Astra resources

The Astra Control REST API conforms to RESTful design principles. Each Astra resource instance is created based on a well-defined resource type. A set of resource instances of the same type is referred to as a **collection**. The API calls act on individual resources or collections of resources.

Resource types

The resource types included with the Astra Control REST API have the following characteristics:

- Every resource type is defined using a schema (typically in JSON)
- Every resource schema includes the resource type and version
- Resource types are globally unique

Resource instances

Resource instances available through the Astra Control REST API have the following characteristics:

- Resource instances are created based on a single resource type
- The resource type is indicated using the Media Type value
- Instances are composed of stateful data which is maintained by the Astra service
- Each instance is accessible through a unique and long-lived URL
- In cases where a resource instance can have more than one representation, different media types can be used to request the desired representation

Resource collections

Resource collections available through the Astra Control REST API have the following characteristics:

- The set of resource instances of a single resource type is known as a collection
- Collections of resources have a unique and long-lived URL

Instance identifiers

Every resource instance is assigned an identifier when it is created. This identifier is a 128-bit UUIDv4 value. The assigned UUIDv4 values are globally unique and immutable. After issuing an API call that creates a new instance, a URL with the associated id is returned to the caller in a `Location` header of the HTTP response. You can extract the identifier and use it on subsequent calls when referring to the resource instance.

The resource identifier is the primary key used for collections.

Common structure for Astra resources

Every Astra Control resource is defined using a common structure.

Common data

Every Astra resource contains the key-values shown in the following table.

Key	Description
type	A globally unique resource type which is known as the **resource type**.
version	A version identifier which is known as the **resource version**.
id	A globally unique identifier which is known as the **resource identifier**.
metadata	A JSON object containing various information, including user and system labels.

Metadata object

The metadata JSON object included with each Astra resource contains the key-values shown in the following table.

Key	Description
labels	JSON array of client-specified labels associated with the resource.
creationTimestamp	JSON string containing a timestamp indicating when the resource was created.
modificationTimestamp	JSON string containing an ISO-8601 formatted timestamp indicating when the resource was last altered.
createdBy	JSON string containing the UUIDv4 identifier of the user id that created the resource. If the resource was created by an internal system component and there is no UUID associated with the creating entity, the **null** UUID is used.

Resource state

Selected resources a `state` value which is used to orchestrate lifecycle transitions and control access.

HTTP details

The Astra Control REST API uses HTTP and related parameters to act on the resource instances and collections. Details of the HTTP implementation are presented below.

API transactions and the CRUD model

The Astra Control REST API implements a transactional model with well-defined operations and state transitions.

Request and response API transaction

Every REST API call is performed as an HTTP request to the Astra service. Each request generates an associated response back to the client. This request-response pair can be considered an API transaction.

Support for CRUD operational model

Each of the resource instances and collections available through the Astra Control REST API is accessed based on the **CRUD** model. There are four operations, each of which maps to a single HTTP method. The

operations include:

- Create
- Read
- Update
- Delete

For some of the Astra resources, only a subset of these operations is supported. You should review the Online API reference for more information about a specific API call.

HTTP methods

The HTTP methods or verbs supported by the API are presented in the table below.

Method	CRUD	Description
GET	Read	Retrieves object properties for a resource instance or collection. This is considered a **list** operation when used with a collection.
POST	Create	Creates a new resource instance based on the input parameters. The long-term URL is returned in a `Location` response header.
PUT	Update	Updates an entire resource instance with the supplied JSON request body. Key values that are not user modifiable are preserved.
DELETE	Delete	Deletes an existing resource instance.

Request and response headers

The following table summaries the HTTP headers used with the Astra Control REST API.

See RFC 7232 and RFC 7233 for more information.

Header	Type	Usage notes
Accept	Request	If the value is "**/**" or is not provided, `application/json` is returned in Content-Type response header. If the value is set to the Astra resource Media Type, the same Media Type is returned in the Content-Type header.
Authorization	Request	Bearer token with the API key for the user.
Content-Type	Response	Returned based on the `Accept` request header.
Etag	Response	Included with a successful as defined with RFC 7232. The value is a hexadecimal representation of the MD5 value for the entire JSON resource.
If-Match	Request	A precondition request header implemented as described in section 3.1 RFC 7232 and support for **PUT** requests.
If-Modified-Since	Request	A precondition request header implemented as described in section 3.4 RFC 7232 and support for **PUT** requests.
If-Unmodified-Since	Request	A precondition request header implemented as described in section 3.4 RFC 7232 and support for **PUT** requests.

Header	Type	Usage notes
Location	Response	Contains the full URL of the newly created resource.

Query parameters

The following query parameters are available for use with resource collections. See Work with collections for more information.

Query parameter	Description
include	Contains the fields that should be returned when reading a collection.
filter	Indicates the fields that must match for a resource to be returned when reading a collection.
orderBy	Determines the sort order of resources returned when reading a collection.
limit	Limits the maximum number of resources returned when reading a collection.
skip	Sets the number of resources to pass over and skip when reading a collection.
count	Indicates if the total number of resources should be returned in the metadata object.

HTTP status codes

The HTTP status codes used by the Astra Control REST API are described below.

The Astra Control REST API also uses the **Problem Details for HTTP APIs** standard. See Diagnostics and support for more information.

Code	Meaning	Description
200	OK	Indicates success for calls that do not create a new resource instance.
201	Created	An object is successfully created and the location response header includes the unique identifier for the object.
204	No content	The request was successful although no content was returned.
400	Bad request	The request input is not recognized or is inappropriate.
401	Unauthorized	The user is not authorized and must authentiate.
403	Forbidden	Access is denied due to an authorization error.
404	Not found	The resource referred to in the request does not exist.
409	Conflict	An attempt to create an object failed because the object already exists.
500	Internal error	A general internal error occurred at the server.
503	Service unavailable	The service is not ready to handle the request for some reason.

URL format

The general structure of the URL used to access a resource instance or collection through the REST API is composed of several values. This structure reflects the

underlying object model and system design.

Account as the root

The root of the resource path to every REST endpoint is the Astra account. And so all paths in the URL begin with `/account/{account_id}` where `account_id` is the unique UUIDv4 value for the account. Internally structure this reflects a design where all resource access is based on a specific account.

Endpoint resource category

The Astra resource endpoints fall into three different categories:

- Core (`/core`)
- Managed application (`/k8s`)
- Topology (`/topology`)

See Resources for more information.

Category version

Each of the three resource categories has a global version that controls the version of the resources accessed. By convention and definition, moving to a new major version of a resource category (such as, from `/v1` to `/v2`) will introduce breaking changes in the API.

Resource instance or collection

A combination of resource types and identifiers can be used in the path, based on whether a resource instance or collection is accessed.

Example

- Resource path

  Based on the structure presented above, a typical path to an endpoint is: `/accounts/{account_id}/core/v1/users`.

- Complete URL

  The full URL for the corresponding endpoint is: `https://astra.netapp.io/accounts/{account_id}/core/v1/users`.

Resources and endpoints

You can access the resources provided through the Astra Control REST API to automate an Astra deployment. Each resource is available through one or more endpoints. An introduction to the REST resources you can use as part of an automation deployment is provided below.

The format of the path and full URL used to access the Astra Control resources is based on several values. See URL format for more information. Also see Online API reference for more details about using the Astra resources and endpoints.

Summary of Astra Control REST resources

The primary resource endpoints provided in the Astra Control REST API are organized in three categories. Each resource can be accessed with the full set of CRUD operations (create, read, update, delete) except where noted.

The **Release** column indicates the Astra release when the resource was first introduced. This field is bolded for the resources most recently added to the REST API.

Core resources

The core resource endpoints provide the foundational services needed to establish and maintain the Astra runtime environment.

Resource	Release	Description
Account	21.12	The account resources allow you to manage the isolated tenants within the multitenant Astra Control deployment environment.
ASUP	21.08	The ASUP resources represent the AutoSupport bundles forwarded to NetApp support.
Certificate	22.08	The certificate resources represent the installed certificates used for strong authentication for outgoing connections.
Credential	21.04	The credential resources contain security related information which can be used with Astra users, clusters, buckets, and storage backends.
Entitlement	21.08	The entitlement resources represent the features and capacities available for an account based on the active licenses and subscriptions.
Event	21.04	The event resources represent all the events occurring in the system, including the subset classified as notifications.
Execution hook	21.12	The execution hook resources represent custom scripts that you can run either before or after a snapshot of a managed app is performed.
Feature	21.08	The feature resources represent selected Astra features that you can query to determine if they are enabled or disabled in the system. Access is limited to read-only.
Group	22.08	The group resources represent the Astra groups and associated resources. Only LDAP groups are supported in the current release.
Hook source	21.12	The hook source resources represent the actual source code used with an execution hook. Separating the source code from the execution control has several benefits such as allowing the scripts to be shared.
LDAP group	22.1	You can list the groups within the configured LDAP server. Access to the LDAP groups is read-only.
LDAP user	22.11	You can list the users within the configured LDAP server. Access to the LDAP users is read-only.
License	21.08	The license resources represent the licenses available for an Astra account.
Notification	21.04	The notification resources represent Astra events that have a notification destination. Access is provided on a per-user basis.

Resource	Release	Description
Package	22.04	The package resources provide registration of and access to package definitions. Software packages consist of various components including files, images, and other artifacts.
Permission	23.06	The permission resources represent permissions related to operations within the system. The API provides read-only access to the permissions.
Role	23.06	The role resources represent roles available in the system. The API provides read-only access to the roles.
Role binding	21.04	The role binding resources represent the relationships between specific pairs of users and accounts. In addition to the linkage between the two, a set of permissions is specified for each through a specific role.
Setting	21.08	The setting resources represent a collection of key-value pairs which describe a feature for a specific Astra account.
Subscription	21.08	The subscription resources represent the active subscriptions for an Astra account.
Task	22.11	The task resources provide read-only access to managed task and can be used to display the status of the internal long-running tasks.
Token	21.04	The token resources represent the tokens available to programmatically access the Astra Control REST API.
Unread notification	21.04	The unread notification resources represent notifications assigned to a specific user but not yet read.
Upgrade	22.04	The upgrade resources provide access to software components and the ability to initiate upgrades.
User	21.04	The user resources represent Astra users able to access the system based on their defined role.

Managed application resources

The managed application resource endpoints provide access to the managed Kubernetes applications.

Resource	Release	Description
Application asset	21.04	The application asset resources represent internal collections of state information needed to manage the Astra applications.
Application backup	21.04	The application backup resources represent backups of the managed applications.
Application snapshot	21.04	The application snapshot resources represent snapshots of the managed applications.
Execution hook override	21.12	The execution hook override resources allow you to disable the preloaded NetApp default execution hooks for specific applications as needed.
Schedule	21.04	The schedule resources represent data protection operations that are scheduled for the managed applications as part of a data protection policy.

Topology resources

The topology resource endpoints provide access to the unmanaged applications and storage resources.

Resource	Release	Description
API resource	22.11	The API resource endpoints provide read-only access to the Kubernetes resources in a specific managed cluster.
App	21.04	The app resources represent all of the Kubernetes applications, including those unmanaged by Astra.
AppMirror	22.08	The AppMirror resources represent the AppMirror resources to provide for the management of application mirroring relationships.
Bucket	21.08	The bucket resources represent the S3 cloud buckets used to store backups of the applications managed by Astra.
Cloud	21.08	The cloud resources represent clouds that Astra clients can connect to in order to manage clusters and applications.
Cluster	21.08	The cluster resources represent the Kubernetes clusters not managed by Kubernetes.
Cluster node	21.12	The cluster node resources provide additional resolution by allowing you to access the individual nodes within a Kubernetes cluster.
Managed cluster	21.08	The managed cluster resources represent the Kubernetes clusters currently managed by Kubernetes.
Namespace	21.12	The namespace resources provide access to the namespaces used within a Kubernetes cluster.
Storage backend	21.08	The storage backend resources represent providers of storage services that can be used by the Astra managed clusters and applications.
Storage class	21.08	The storage class resources represent different classes or types of storage discovered and available to a specific managed cluster.
Volume	21.04	The volume resources represent the Kubernetes storage volumes associated with the managed applications.

Additional resources and endpoints

There are several additional resources and endpoints that you can use to support an Astra deployment.

These resources and endpoints are not currently included with the Astra Control REST API reference documentation.

OpenAPI

The OpenAPI endpoints provide access to the current OpenAPI JSON document and other related resources.

OpenMetrics

The OpenMetrics endpoints provide access to the account metrics through the OpenMetrics resource. Support is available with the Astra Control Center deployment model.

Additional considerations

RBAC security

The Astra REST API supports role-based access control (RBAC) to grant and restrict access to the system functions.

Astra roles

Every Astra user is assigned to a single role which determines the actions that can be performed. The roles are arranged in a hierarchy as described in the table below.

Role	Description
Owner	Has all the permissions of the Admin role and can also delete Astra accounts.
Admin	Has all the permissions of the Member role and can also invite users to join an account.
Member	Can fully manage the Astra application and compute resources.
Viewer	Restricted to only viewing resources.

Enhanced RBAC with namespace granularity

This feature was introduced with the 22.04 release of the Astra REST API.

When a role binding is established for a specific user, a constraint can be applied to limit the namespaces the user has access to. There are several ways this constraint can be defined as described in the table below. See the parameter `roleContraints` in the Role Binding API for more information.

Namespaces	Description
All	The user can access all the namespaces through the wildcard parameter "*"". This is the default value to maintain backwards compatibility.
None	The constraint list is specified although it is empty. This indicates the user cannot access any namespace.
Namespace list	The UUID of a namespace is included which restricts the user to the single namespace. A comma separated list can also be used to allow access to multiple namespaces.
Label	A label is specified and access is allowed to all the matching namespaces.

Work with collections

The Astra Control REST API provides several different ways to access resource collections through the defined query parameters.

Selecting values

You can specify which key-value pairs should be returned for each resource instance using the `include` parameter. All of the instances are returned in the response body.

Filtering

Collection resource filtering allows an API user to specify conditions which determine if a resource is returned

in the response body. The `filter` parameter is used to indicate the filtering condition.

Sorting

Collection resource sorting allows an API user to specify the order in which resources are returned in the response body. The `orderBy` parameter is used to indicate the filtering condition.

Pagination

You can enforce pagination by restricting the number of resource instances returned on a request using the `limit` parameter.

Count

If you include the Boolean parameter `count` set to `true`, the number of resources in the returned array for a given response is provided in the metadata section.

Diagnostics and support

There are several support features available with the Astra Control REST API that can be used for diagnostics and debugging.

API resources

There are several Astra features exposed through API resources that provide diagnostic information and support.

Type	Description
Event	System activities that are recorded as part of Astra processing.
Notification	A subset of the Events that are considered important enough to be presented to the user.
Unread notification	The notifications that have yet to be read or retrieved by the user.

Revoke an API token

You can revoke an API token at the Astra web interface when it is no longer needed.

Before you begin

You need credentials to sign in to the Astra web user interface for your deployment. You should also identify the tokens you want to revoke.

About this task

After a token is revoked, it is immediately and permanently unusable.

Steps

1. Sign in to Astra using your account credentials as follows:
   a. Astra Control Service: https://astra.netapp.io
   b. Astra Control Center: Use the URL for your local environment as established during installation
2. Click the figure icon at the top right of the page and select **API access**.
3. Select the token or tokens you want to revoke.

4. Under the **Actions** drop-down box, click **Revoke tokens**.

Copyright information

Trademark information