Astra RESTの実装

Astra Automation

NetApp
March 07, 2024

目次

Astra RESTの実装

コア設計

REST Web サービス

Representational State Transfer （ REST ）は、分散 Web アプリケーションの作成に使用される形式です。Web サービス API の設計に適用されることで、サーバベースのリソースを公開し、その状態を管理するための一連の主流テクノロジとベストプラクティスが確立されます。REST はアプリケーション開発のための一貫した基盤を提供しますが、各 API の詳細は設計内容に応じて異なる場合があります。ライブ環境で使用する前に、 Astra Control REST API の特性を理解しておく必要があります。

リソースと状態の表示

リソースは、 Web ベースシステムの基本コンポーネントです。REST Web サービスアプリケーションを作成する場合、設計の早い段階で次の作業を行います。

- システムまたはサーバベースのリソースの識別

  すべてのシステムは、リソースを使用および管理します。リソースには、ファイル、ビジネストランザクション、プロセス、または管理エンティティがあります。REST Web サービスに基づいてアプリケーションを設計する際に行う最初の作業の 1 つは、リソースを識別することです。

- リソースの状態および関連する状態操作の定義

  リソースの状態の数は有限で、リソースは必ずそのいずれかの状態にあります。状態、および状態の変化に影響する関連操作を明確に定義する必要があります。

URI エンドポイント

すべての REST リソースは、明確に定義されたアドレス指定方式を使用して定義および使用可能にする必要があります。 リソースが置かれているエンドポイントは、 Uniform Resource Identifier （ URI ）で識別されます。URI は、ネットワーク内の各リソースに一意の名前を作成するための一般的なフレームワークです。Uniform Resource Locator （ URL ）は、リソースを識別してアクセスするために Web サービスで使用される URI の一種です。リソースは通常、ファイルディレクトリに似た階層構造で公開されます。

HTTP メッセージ

Hypertext Transfer Protocol （ HTTP ）は、 Web サービスのクライアントとサーバがリソースに関する要求と応答のメッセージを交換する際に使用するプロトコルです。Web サービスアプリケーションの設計の一環として、 HTTP メソッドはリソースおよび対応する状態管理アクションにマッピングされます。HTTP はステートレスです。したがって、関連する一連の要求と応答を 1 つのトランザクションの一部として関連付けるには、要求と応答のデータフローで伝送される HTTP ヘッダーに追加情報を含める必要があります。

JSON 形式

Web サービスのクライアントとサーバの間で情報を構造化して転送する方法は複数ありますが、最も広く使用されているのは JavaScript Object Notation （ JSON ）です。JSON は、単純なデータ構造をプレーンテキ

ストで表すための業界標準であり、リソースについての状態情報の転送に使用されます。Astra Control REST API では、 JSON を使用して、各 HTTP 要求と応答の本文で伝送されるデータをフォーマットします。

リソースとコレクション

Astra Control REST API を使用すると、リソースインスタンスとリソースインスタンスのコレクションにアクセスできます。

> 概念的には REST * リソース * は、オブジェクト指向プログラミング（ OOP ）言語およびシステムで定義される * オブジェクト * に似ています。これらの用語が同じ意味で使用されることもあります。ただし一般には、外部 REST API のコンテキストで使用する場合は「リソース」が推奨され、サーバに格納される対応するステートフルインスタンスデータには「オブジェクト」が使用されます。

Astra リソースの属性

Astra Control REST API は、 RESTful 設計の原則に準拠しています。各アストラリソースインスタンスは、明確に定義されたリソースタイプに基づいて作成されます。同じタイプのリソースインスタンスのセットを * 集合 * と呼びます。API 呼び出しは、個々のリソースまたはリソースの集合に対して機能します。

リソースタイプ

Astra Control REST API に含まれるリソースタイプには、次の特徴があります。

- すべてのリソースタイプはスキーマを使用して定義されます（通常は JSON 形式）。
- すべてのリソーススキーマには、リソースタイプとバージョンが含まれています
- リソースタイプはグローバルに一意です

リソースインスタンス

Astra Control REST API から使用できるリソースインスタンスには、次のような特徴があります。

- リソースインスタンスは、単一のリソースタイプに基づいて作成されます
- リソースタイプは、 ［ メディアタイプ ］ の値を使用して示されます
- インスタンスは、アストラサービスによって管理されるステートフルデータで構成されます
- 各インスタンスには、一意で長く存続する URL を使用してアクセスできます
- リソースインスタンスが複数のリプレゼンテーションを持つことができる場合は、異なるメディアタイプを使用して必要なリプレゼンテーションを要求できます

リソースコレクション

Astra Control REST API で使用できるリソース収集には、次のような特徴があります。

- 単一のリソースタイプの一連のリソースインスタンスをコレクションと呼びます
- 一連のリソースには、一意で一時的な URL があります

インスタンス **ID**

すべてのリソースインスタンスには、作成時に識別子が割り当てられます。この識別子は 128 ビットの UUIDv4 値です。割り当てられた UUIDv4 の値は、グローバルに一意で変更できません。新しいインスタンス

を作成するAPI呼び出しを発行すると、関連付けられたIDを含むURLがの呼び出し元に返されます `Location` HTTP応答のヘッダー。リソースインスタンスを以降の呼び出しで参照する際には、この識別子を抽出して使用できます。

リソース識別子は、コレクションで使用される主キーです。

Astra のリソースに共通の構造

すべてのアストラ制御リソースは、共通の構造を使用して定義されます。

共通のデータ

すべてのアストラリソースには、次の表に示すキーと値が含まれています。

キーを押します	説明
を入力します	グローバルに一意なリソースタイプ。 * リソースタイプ * と呼ばれます。
バージョン	* リソースバージョン * と呼ばれるバージョン識別子。
ID	グローバル一意識別子。 * リソース識別子 * と呼ばれます。
メタデータ	ユーザラベルやシステムラベルなどのさまざまな情報を含む JSON オブジェクト。

メタデータオブジェクト

各アストラリソースに含まれるメタデータ JSON オブジェクトには、次の表に示すキーと値が含まれています。

キーを押します	説明
ラベル	リソースに関連付けられているクライアント指定のラベルの JSON 配列。
作成タイムスタンプ	リソースが作成されたときのタイムスタンプを含む JSON 文字列。
modificationTimestamp	リソースの最終変更日時を示す ISO-8601 形式のタイムスタンプを含む JSON 文字列。
作成者	リソースを作成したユーザ ID の UUIDv4 識別子を含む JSON 文字列。リソースが内部システムコンポーネントによって作成され、作成するエンティティに関連付けられている UUID がない場合は、「 * null * UUID 」が使用されます。

リソースの状態

選択したリソースA `state` ライフサイクル移行のオーケストレーションとアクセスの制御に使用する値。

HTTP の詳細

Astra Control REST APIは、HTTPおよび関連パラメータを使用して、リソースのインスタンスおよびコレクションに対して動作します。HTTP 実装の詳細を以下に示します。

API トランザクションと CRUD モデル

Astra Control REST API は、明確に定義された操作と状態の移行を伴うトランザクションモデルを実装します。

要求と応答の API トランザクション

すべての REST API 呼び出しは、 Astra サービスへの HTTP 要求として実行されます。要求ごとに、関連付けられた応答がクライアントに生成されます。この要求と応答のペアで API トランザクションを使用できます。

CRUD 操作モデルのサポート

Astra Control REST API で使用できる各リソースインスタンスとコレクションには、 *CRUD * モデルに基づいてアクセスします。4 つの操作があり、それぞれが単一の HTTP メソッドにマッピングされます。処理は次のとおりです。

- 作成
- 読み取り
- 更新
- 削除

一部の Astra リソースでは、これらの操作の一部のみがサポートされています。を確認しておきます "オンラインのAPIリファレンス" 特定の API 呼び出しに関する詳細については、を参照してください。

HTTP メソッド

次の表に、 API でサポートされる HTTP メソッドまたは動詞を示します。

メソッド	CRUD	説明
取得	読み取り	リソースインスタンスまたはコレクションのオブジェクトプロパティを取得します。これは、コレクションとともに使用される場合、 * list * 演算と見なされます。
投稿（ Post ）	作成	入力パラメータに基づいて新しいリソースインスタンスを作成します。長期URLは、で返されます `Location` 応答ヘッダー。
PUT	更新	指定した JSON 要求の本文でリソースインスタンス全体を更新します。ユーザが変更できないキー値は保持されます。
削除	削除	既存のリソースインスタンスを削除します。

要求と応答のヘッダー

次の表に、 Astra Control REST API で使用される HTTP ヘッダーを示します。

を参照してください "RFC 7232" および "RFC 7233" を参照してください。

ヘッダー	を入力します	使用上の注意
同意します	リクエスト	値が「/」または指定されていない場合、 `application/json` Content-Type応答ヘッダーで返されます。値が Astra リソースのメディアタイプに設定されている場合、同じメディアタイプが Content-Type ヘッダーに返されます。
承認	リクエスト	ユーザーの API キーを使用した Bearer トークン。
コンテンツタイプ	応答	に基づいて返されます `Accept` 要求ヘッダー。
ETag	応答	RFC 7232 で定義されたとおりに、成功したに含まれます。値は、 JSON リソース全体の MD5 値を 16 進数で表したものです。
if-match	リクエスト	セクション 3.1 「 RFC 7232 」で説明されているように、 * PUT * 要求をサポートする前提条件の要求ヘッダーが実装されています。
If-Modified-Since の略	リクエスト	セクション 3.4 「 RFC 7232 」に記載されている前提条件の要求ヘッダーと、 *PUT * 要求のサポート。
If - unmodified- since	リクエスト	セクション 3.4 「 RFC 7232 」に記載されている前提条件の要求ヘッダーと、 *PUT * 要求のサポート。
場所	応答	新しく作成されたリソースの完全な URL が含まれます。

クエリパラメータ

リソースコレクションでは、次のクエリパラメータを使用できます。を参照してください "コレクションを操作する" を参照してください。

クエリパラメータ	説明
含める	コレクションを読み取るときに返すフィールドが含まれています。
フィルタ	コレクションの読み取り時にリソースが返されるために一致しなければならないフィールドを示します。
OrderBy	コレクションの読み取り時に返されるリソースのソート順序を指定します。
制限（ Limit ）	コレクションの読み取り時に返されるリソースの最大数を制限します。
スキップします	コレクションの読み取り時に、パスオーバーおよびスキップするリソースの数を設定します。
カウント	メタデータオブジェクトで返されるリソースの総数。

HTTP ステータスコード

Astra Control REST API で使用される HTTP ステータスコードを次に示します。

Astra Control REST API は、 HTTP API * 標準の * 問題の詳細も使用します。を参照してください "診断とサポート" を参照してください。

コード	意味	説明
200	わかりました	新しいリソースインスタンスを作成しない呼び出しが成功したことを示します。
201	作成済み	オブジェクトが作成され、場所の応答ヘッダーにオブジェクトの一意の識別子が含まれている。
204	コンテンツがありません	要求は成功しましたが、コンテンツが返されませんでした。
400	無効な要求です	要求の入力が認識されないか不適切です。
401	権限がありません	ユーザは認証されていないため、認証が必要です。
403です	禁止されている	認証エラーによりアクセスが拒否されました。
404です	が見つかりません	要求で参照されているリソースが存在しません。
409だ	競合しています	オブジェクトがすでに存在するため、オブジェクトの作成に失敗しました。
500ドル	内部エラー	サーバで一般的な内部エラーが発生しました。
503	サービスを利用できません	サービスは何らかの理由で要求を処理する準備ができていません。

URL 形式

REST API を使用したリソースインスタンスまたはコレクションへのアクセスに使用される URL の一般的な構造は、いくつかの値で構成されます。この構造は、基礎となるオブジェクトモデルとシステム設計を反映しています。

root としてアカウントを作成します

すべての REST エンドポイントへのリソースパスのルートは Astra アカウントです。そのため、URL内のすべてのパスはで始まります `/account/{account_id}` ここで、 `account_id` は、アカウントの一意のUUIDv4値です。内部構造：すべてのリソースアクセスが特定のアカウントに基づいている設計を反映します。

エンドポイントリソースカテゴリ

Astra リソースエンドポイントは、次の 3 つのカテゴリに分類されます。

- コア (`/core`)
- マネージドアプリケーション (`/k8s`)
- トポロジ (`/topology`)

を参照してください "リソース" を参照してください。

カテゴリバージョン

3 つのリソースカテゴリのそれぞれに、アクセスされるリソースのバージョンを制御するグローバルバージョンがあります。規則および定義により、リソースカテゴリの新しいメジャーバージョン（例えば、から）に移動します `/v1` 終了： `/v2`）を使用すると、APIの変更が破棄されます。

リソースインスタンスまたはコレクション

リソースインスタンスまたはコレクションにアクセスするかどうかに基づいて、パスでリソースタイプと識別子の組み合わせを使用できます。

例

- リソースパス

  上記の構造に基づいて、エンドポイントへの一般的なパスは次のとおりです。
  `/accounts/{account_id}/core/v1/users`。

- 完全な URL

  対応するエンドポイントの完全なURLは次のとおりです。
  `https://astra.netapp.io/accounts/{account_id}/core/v1/users`。

リソースとエンドポイント

Astra Control REST APIを使用して提供されるリソースにアクセスすることで、Astraの導入を自動化できます。各リソースは、1つ以上のエンドポイントから使用できます。以下に、自動化導入の一環として使用できるRESTリソースの概要を示します。

Astra Control リソースへのアクセスに使用されるパスと完全な URL の形式は、いくつかの値に基づいています。を参照してください "URL 形式" を参照してください。も参照してください "オンラインのAPIリファレンス" Astra のリソースとエンドポイントの使用方法の詳細については、を参照してください。

Astra Control REST リソースの要約

Astra Control REST API に用意されているプライマリリソースエンドポイントは、 3 つのカテゴリに分類されています。各リソースは、特定の場所を除いて、すべての CRUD 操作（作成、読み取り、更新、削除）を使用してアクセスできます。

[* リリース *] 列は、リソースが最初に導入されたときのアストラリリースを示します。このフィールドは、REST APIに最後に追加されたリソースに対して太字で表示されます。

コアリソース

コアリソースエンドポイントは、 Astra ランタイム環境の確立と保守に必要な基盤サービスを提供します。

リソース	リリース。	説明
アカウント：	21.12	アカウントリソースを使用すると、マルチテナント Astra Control 導入環境内の分離されたテナントを管理できます。
ASUP	21.08	ASUP リソースには、ネットアップサポートに転送される AutoSupport バンドルが表示されます。
証明書	22.08	証明書リソースは、発信接続の強力な認証に使用されるインストール済み証明書を表します。

リソース	リリース。	説明
クレデンシャル	21.04	クレデンシャルリソースには、 Astra ユーザ、クラスタ、バケット、ストレージバックエンドで使用できるセキュリティ関連の情報が含まれています。
使用権	21.08	使用権リソースは、アクティブなライセンスとサブスクリプションに基づいて、アカウントで使用可能な機能と容量を表します。
イベント	21.04	イベントリソースは、通知として分類されたサブセットも含めて、システムで発生するすべてのイベントを表します。
実行フック	21.12	実行フックリソースは、管理対象アプリケーションのスナップショットの実行前または実行後に実行できるカスタムスクリプトを表します。
フィーチャー（ Feature ）	21.08	機能リソースは、選択した Astra 機能を表します。この機能を照会することで、システムで有効になっているか無効になっているかを確認できます。アクセスは読み取り専用に制限されます。
グループ	22.08	グループリソースは、アストラグループと関連するリソースを表します。現在のリリースでは、LDAPグループのみがサポートされています。
フックソース	21.12	フックソースリソースは、実行フックで使用される実際のソースコードを表します。ソースコードを実行コントロールから分離すると、スクリプトを共有できるようにするなど、いくつかの利点があります。
LDAPグループ	22.1	設定したLDAPサーバ内のグループをリストできます。LDAPグループへのアクセスは読み取り専用です。
LDAPユーザ	22.11	設定したLDAPサーバ内のユーザをリストできます。LDAPユーザへのアクセスは読み取り専用です。
使用許諾	21.08	ライセンスリソースは、 Astra アカウントで使用可能なライセンスを表します。
通知	21.04	通知リソースは、通知の送信先を持つ Astra イベントを表します。アクセスはユーザ単位で提供されます。
パッケージ	22.04	パッケージリソースは、パッケージ定義の登録とアクセスを提供します。ソフトウェアパッケージは、ファイル、イメージ、その他のアーティファクトなどのさまざまなコンポーネントで構成されます。
アクセス権	2006年3月23日	権限リソースは、システム内の処理に関連する権限を表します。このAPIは、権限への読み取り専用アクセスを提供します。
ロール	2006年3月23日	役割リソースは、システムで使用可能な役割を表します。APIでは、ロールへの読み取り専用アクセスが提供されます。
ロールのバインド	21.04	ロールバインドリソースは ' 特定のユーザーとアカウントのペア間の関係を表します2 つの間のリンクに加えて、特定のロールを通じて各に一連の権限が指定されます。
設定	21.08	設定リソースは、特定のアストラアカウントの機能を説明するキーと値のペアの集まりです。
サブスクリプション。	21.08	サブスクリプションリソースは、 Astra アカウントのアクティブなサブスクリプションです。
タスク	22.11	タスクリソースは、管理タスクへの読み取り専用アクセスを提供し、内部の長時間実行タスクのステータスを表示するために使用できます。

リソース	リリース。	説明
トークン	21.04	トークンリソースとは、プログラムによって Astra Control REST API にアクセスできるトークンのことです。
未読通知	21.04	未読通知リソースは、特定のユーザーに割り当てられているがまだ読み込まれていない通知を表します。
アップグレード	22.04	アップグレードリソースを使用して、ソフトウェアコンポーネントにアクセスしたり、アップグレードを開始したりできます。
ユーザ	21.04	ユーザリソースは、 Astra を使用するユーザで、定義したロールに基づいてシステムにアクセスできる。

アプリケーションリソースの管理

管理対象アプリケーションリソースエンドポイントは、管理対象の Kubernetes アプリケーションへのアクセスを提供します。

リソース	リリース。	説明
アプリケーションアセット	21.04	アプリケーションアセットリソースは、 Astra アプリケーションの管理に必要な状態情報の内部コレクションです。
アプリケーションのバックアップ	21.04	アプリケーションのバックアップリソースは、管理対象アプリケーションのバックアップを表します。
アプリケーションスナップショット	21.04	アプリケーションスナップショットリソースは、管理対象アプリケーションのスナップショットを表します。
実行フックのオーバーライド	21.12	実行フックの上書きリソースを使用すると、特定のアプリケーションにプリインストールされているネットアップのデフォルトの実行フックを必要に応じて無効にできます。
スケジュール	21.04	スケジュールリソースとは、データ保護ポリシーの一部として管理対象アプリケーションにスケジュールされているデータ保護処理のことです。

トポロジリソース

トポロジリソースエンドポイントは、管理対象外のアプリケーションとストレージリソースへのアクセスを提供します。

リソース	リリース。	説明
APIリソース	22.11	APIリソースエンドポイントは、特定の管理対象クラスタ内のKubernetesリソースへの読み取り専用アクセスを提供します。
アプリケーション	21.04	アプリケーションリソースは、 Astra が管理していないアプリケーションも含め、 Kubernetes のすべてのアプリケーションを表します。
AppMirror（アプリケーションミラー）	22.08	AppMirrorリソースは、アプリケーションのミラーリング関係を管理するためのAppMirrorリソースを表します。
バケット	21.08	バケットリソースは、 Astra が管理するアプリケーションのバックアップを保存するために使用する S3 クラウドバケットです。

リソース	リリース。	説明
クラウド	21.08	クラウドリソースとは、アストラクライアントから接続してクラスタやアプリケーションを管理できるクラウドのことです。
クラスタ	21.08	クラスタリソースは Kubernetes で管理されない Kubernetes クラスタを表します。
クラスタノード	21.12	クラスタノードリソースは、 Kubernetes クラスタ内の個々のノードにアクセスできるようにすることで、解決策を提供します。
管理対象クラスタ	21.08	管理対象クラスタリソースは、 Kubernetes で現在管理されている Kubernetes クラスタを表します。
ネームスペース	21.12	ネームスペースリソースは、 Kubernetes クラスタ内で使用されるネームスペースへのアクセスを提供します。
ストレージバックエンド	21.08	ストレージバックエンドリソースは、 Astra が管理するクラスタとアプリケーションで使用できるストレージサービスのプロバイダです。
ストレージクラス	21.08	ストレージクラスのリソースは、さまざまなクラスやタイプのストレージを表しており、特定の管理対象クラスタで使用できます。
ボリューム	21.04	ボリュームリソースは、管理対象アプリケーションに関連付けられた Kubernetes ストレージボリュームを表します。

その他のリソースとエンドポイント

Astra の導入をサポートするために使用できる追加のリソースとエンドポイントがいくつかあります。

これらのリソースとエンドポイントは、現在のところ、 Astra Control REST API リファレンスドキュメントに含まれていません。

OpenAPI

OpenAPI エンドポイントは、現在の OpenAPI JSON ドキュメントおよびその他の関連リソースへのアクセスを提供します。

OpenMetrics

OpenMetrics エンドポイントは、 OpenMetrics リソースを介してアカウントメトリックへのアクセスを提供します。サポートは、 Astra Control Center 導入モデルで利用できます。

その他の考慮事項については

RBACセキュリティ

Astra REST APIは、ロールベースアクセス制御（RBAC）をサポートしており、システム機能へのアクセスを許可および制限します。

Astra の役割

すべての Astra ユーザには、実行可能なアクションを決定する 1 つのロールが割り当てられます。役割は、次の表に示すように階層構造になっています。

ロール	説明
オーナー	admin ロールのすべての権限が割り当てられており、 Astra アカウントを削除することもできます。
管理	メンバーの役割のすべての権限を持ち、ユーザーをアカウントに招待することもできます。
メンバー	Astra アプリケーションとコンピューティングのリソースを完全に管理できる。
ビューアー（ Viewer ）	リソースの表示のみに制限されます。

ネームスペース単位で強化された **RBAC**

この機能は、 Astra REST API の 22.04 リリースで導入されました。

特定のユーザに対してロールバインドが確立されている場合は、制約を適用して、ユーザがアクセスできるネームスペースを制限できます。次の表に示すように、この制約を定義する方法はいくつかあります。パラメータを参照してください `roleContraints` 詳細については、Role Binding APIを参照してください。

ネームスペース	説明
すべて	ユーザは、ワイルドカードパラメータ「 * 」を使用してすべてのネームスペースにアクセスできます。これは、下位互換性を維持するためのデフォルト値です。
なし	拘束リストは指定されますが、空です。これは、ユーザがどのネームスペースにもアクセスできないことを示します。
ネームスペースリスト	ユーザを 1 つのネームスペースに制限するネームスペースの UUID が含まれています。カンマで区切ったリストを使用して、複数のネームスペースへのアクセスを許可することもできます。
ラベル	ラベルが指定され、一致するすべてのネームスペースへのアクセスが許可されます。

コレクションを操作する

Astra Control REST API には、定義されたクエリパラメータを使用してリソースコレクションにアクセスするためのさまざまな方法があります。

値の選択

を使用して、各リソースインスタンスに対して返すキーと値のペアを指定できます `include` パラメータすべてのインスタンスが応答の本文で返されます。

フィルタリング

収集リソースのフィルタリングを使用すると、 API ユーザは、応答の本文でリソースが返されるかどうかを決定する条件を指定できます。。 `filter` パラメータは、フィルタリング条件を示すために使用されます。

並べ替え

収集リソースのソートを使用すると、 API ユーザは応答の本文でリソースが返される順序を指定できます。。 `orderBy` パラメータは、フィルタリング条件を示すために使用されます。

ページ付け

ページ付けを適用するには、を使用して要求に対して返されるリソースインスタンスの数を制限します

`limit` パラメータ

カウント

Booleanパラメータを含める場合 `count` をに設定します `true`の場合、返される特定の応答について返されるアレイ内のリソースの数がメタデータセクションに表示されます。

診断とサポート

診断とデバッグに使用できる Astra Control REST API には、いくつかのサポート機能が用意されています。

API リソース

API リソースからは、診断情報とサポート情報を提供する Astra 機能がいくつか提供されています。

を入力します	説明
イベント	アストラ処理の一部として記録されるシステムアクティビティ。
通知	ユーザに提供するのに十分な重要性があると見なされるイベントのサブセット。
未読通知	ユーザがまだ読み取りまたは取得していない通知。

API トークンを取り消します

不要になった API トークンは、 Astra Web インターフェイスで取り消すことができます。

作業を開始する前に

導入時にAstra Webユーザインターフェイスにサインインするには、クレデンシャルが必要です。 また、取り消すトークンを特定する必要があります。

このタスクについて

トークンが取り消されると、そのトークンはただちに永続的に使用できなくなります。

手順

1. 次のアカウントクレデンシャルを使用してAstraにサインインします。
   a. Astra Controlサービス： "https://astra.netapp.io"
   b. Astra Control Center：インストール時に設定したローカル環境のURLを使用
2. ページの右上にある図のアイコンをクリックし、 * API access * を選択します。
3. 取り消すトークンまたはトークンを選択します。
4. [* アクション * （ * Actions * ）] ドロップダウンボックスで、 [トークンの無効化 * （ * Revoke tokens * ）] をクリック

著作権に関する情報

商標に関する情報