# Einzelheiten zur REST-Implementierung

ONTAP Automation

NetApp
July 25, 2024

# Inhalt

# Einzelheiten zur REST-Implementierung

## REST-Web-Services-Grundlage

Representational State Transfer (REST) ist ein Stil für die Erstellung von verteilten Web-Anwendungen. Bei der Anwendung auf das Design einer Web-Services-API stellt sie eine Reihe von Technologien her, mit denen Server-basierte Ressourcen offengelegt und deren Status verwaltet werden können. Die flexible Grundlage für die Administration von ONTAP Clustern bildet mit Mainstream-Protokollen und -Standards.

IM RUHEZUSTAND werden einheitliche Technologien und Best Practices festgelegt, jedoch können die Details jeder API je nach den während der Entwicklung getroffenen Entscheidungen variieren. Vor der Verwendung mit einer Live-Implementierung sollten Sie sich mit den Designeigenschaften der ONTAP REST API bewusst sein.

### Ressourcen- und Zustandsdarstellung

Ressourcen sind die Grundkomponenten eines webbasierten Systems. Beim Erstellen einer ANWENDUNG FÜR REST-Webservices umfassen die frühen Designaufgaben Folgendes:

- Identifizierung von System- oder serverbasierten Ressourcen

  Jedes System nutzt und verwaltet Ressourcen. Eine Ressource kann eine Datei-, Geschäftstransaktion-, Prozess- oder Verwaltungseinheit sein. Eine der ersten Aufgaben bei der Entwicklung einer auf REST-Webservices basierenden Applikation ist die Identifizierung der Ressourcen.

- Definition von Ressourcenstatus und zugehörigen Statusoperationen

  Die Ressourcen befinden sich immer in einer endlichen Anzahl von Staaten. Die Zustände sowie die damit verbundenen Operationen, die zur Auswirkung der Statusänderungen verwendet werden, müssen klar definiert werden.

### URI-Endpunkte

Jede REST-Ressource muss definiert und über ein gut definiertes Adressierungssystem verfügbar gemacht werden. Die Endpunkte, in denen die Ressourcen gefunden und identifiziert werden, verwenden einen einheitlichen Resource Identifier (URI). Der URI bietet ein allgemeines Framework zum Erstellen eines eindeutigen Namens für jede Ressource im Netzwerk. Der Uniform Resource Locator (URL) ist ein URI-Typ, der mit Webservices zur Identifizierung und zum Zugriff von Ressourcen verwendet wird. Ressourcen werden in der Regel in einer hierarchischen Struktur ausgesetzt, die einem Dateiverzeichnis ähnelt.

### HTTP-Meldungen

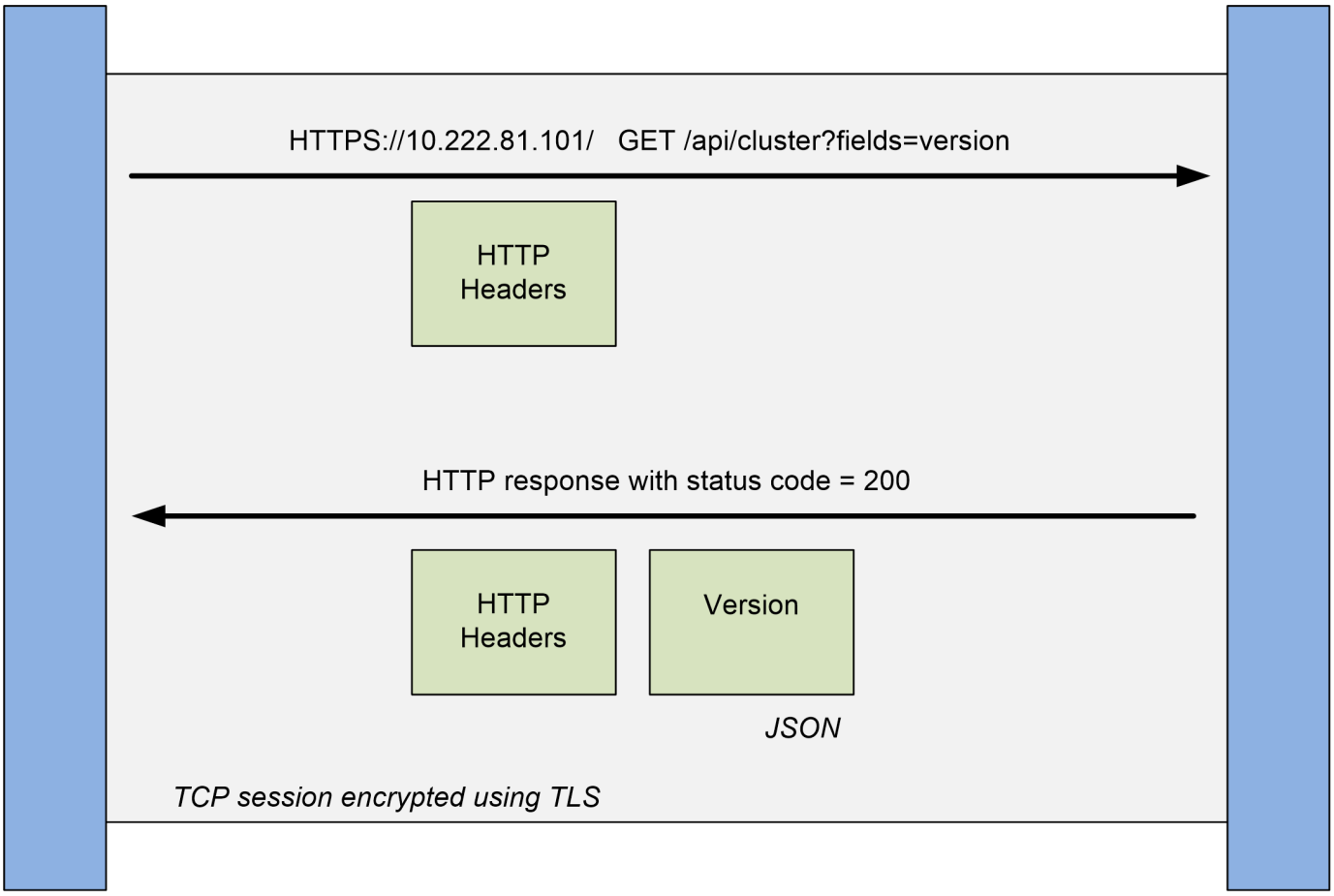
Hypertext Transfer Protocol (HTTP) ist das Protokoll, das vom Webservice-Client und -Server zum Austausch von Anforderungs- und Antwortmeldungen zu den Ressourcen verwendet wird. Im Rahmen der Entwicklung einer Web-Services-Anwendung werden HTTP-Methoden den Ressourcen und entsprechenden Statusmanagement-Aktionen zugeordnet. HTTP ist statusfrei. Um im Rahmen einer Transaktion eine Reihe verwandter Anforderungen und Antworten zuzuordnen, müssen daher zusätzliche Informationen in die HTTP-Header enthalten sein, die mit den Anforderungs- und Antwortdatenströmen verwendet werden.

## JSON-Formatierung

Obwohl Informationen auf verschiedene Weise zwischen einem Webservice-Client und Server strukturiert und übertragen werden können, ist die beliebteste Option JavaScript Object Notation (JSON). JSON ist ein Branchenstandard für die Darstellung einfacher Datenstrukturen im Klartext und wird zur Übertragung von Zustandsdaten zur Beschreibung der Ressourcen verwendet. Die ONTAP REST API verwendet JSON, um die Daten zu formatieren, die im Körper jeder HTTP-Anfrage und Antwort verwendet werden.

## Typische REST API-Transaktion

Jede API-Transaktion besteht aus einer HTTP-Anfrage und der zugehörigen Antwort. Diese Abbildung zeigt, wie Sie die Version der vom Cluster verwendeten ONTAP Software abrufen können.




### HTTP-Anforderung

Die vom Client an den Server gesendete Anforderung besteht aus folgenden Komponenten:

- Verb
- URL-Pfad für das Cluster
- Abfrageparameter (Felder)
- Kopfzeilen für Anfragen, einschließlich Autorisierung

### HTTP-Antwort

Die Antwort, die vom Server an den Client gesendet wird, besteht aus folgenden Komponenten:

- Statuscode 200
- Antwortkopfzeilen
- Response Body mit der Cluster-Softwareversion

# Grundlegende betriebliche Eigenschaften

IM RUHEZUSTAND werden einheitliche Technologien und Best Practices erstellt, jedoch können die Details jeder API je nach dem verfügbaren Design variieren.

## API-Transaktion bei Anfrage und Reaktion

Jeder REST-API-Aufruf wird als HTTP-Anfrage an das ONTAP-System durchgeführt, was eine damit verbundene Antwort an den Client generiert. Dieses Anforderungs-/Antwortpaar wird als API-Transaktion betrachtet. Bevor Sie die API verwenden, sollten Sie mit den verfügbaren Eingabevariablen zur Steuerung einer Anfrage und dem Inhalt der Antwortausgabe vertraut sein.

## Unterstützung von CRUD-Vorgängen

Auf alle über das ONTAP REST API verfügbaren Ressourcen kann basierend auf dem CRUD-Modell zugegriffen werden:

- Erstellen
- Lesen
- Aktualisierung
- Löschen

Für einige der Ressourcen wird nur ein Teil der Vorgänge unterstützt. Sie sollten die ONTAP-API-Dokumentationsseite im ONTAP Cluster überprüfen, um weitere Informationen zu jeder Ressource zu erhalten.

## Objektkennungen

Jeder Ressourceninstanz oder jedem Objekt wird eine eindeutige Kennung zugewiesen, wenn sie erstellt wird. In den meisten Fällen ist die Kennung eine 128-Bit-UUID. Diese Kennungen sind innerhalb eines bestimmten ONTAP Clusters global eindeutig. Nachdem ein API-Aufruf ausgegeben wurde, der eine neue Objektinstanz erstellt, wird eine URL mit dem zugeordneten id-Wert an den Anrufer im Standortkopf der HTTP-Antwort zurückgegeben. Sie können die Kennung extrahieren und bei nachfolgenden Anrufen verwenden, wenn Sie sich auf die Ressourceninstanz beziehen.

Der Inhalt und die interne Struktur der Objektkennungen können jederzeit geändert werden. Wenn Sie auf die zugeordneten Objekte verweisen, sollten Sie die Kennungen für die entsprechenden API-Aufrufe nur nach Bedarf verwenden.

## Objektinstanzen und -Sammlungen

Je nach Ressourcenpfad und HTTP-Methode kann ein API-Aufruf auf eine bestimmte Objektinstanz oder eine Sammlung von Objekten angewendet werden.

## Synchroner und asynchroner Betrieb

Es gibt zwei Möglichkeiten, wie ONTAP eine von einem Client empfangene HTTP-Anfrage durchführt.

### Synchrone Verarbeitung

ONTAP führt die Anfrage sofort aus und antwortet mit einem HTTP-Statuscode von 200 oder 201, wenn er erfolgreich ist.

Jede Anfrage mit den Methoden GET, HEAD und OPTIONEN wird immer synchron ausgeführt. Darüber hinaus werden Anfragen, die POST, PATCH und LÖSCHEN verwenden, synchron ausgeführt, wenn sie voraussichtlich in weniger als zwei Sekunden abgeschlossen werden.

### Asynchrone Verarbeitung

Wenn eine asynchrone Anforderung gültig ist, erstellt ONTAP eine Hintergrundaufgabe zur Verarbeitung der Anforderung und ein Jobobjekt zum Anker der Aufgabe. Der HTTP-Status 202 wird zusammen mit dem Jobobjekt an den Anrufer zurückgegeben. Um den endgültigen Erfolg oder Fehlschlag zu bestimmen, müssen Sie den Status des Jobs abrufen.

Anfragen, die die Methoden POST, PATCH und LÖSCHUNG verwenden, werden asynchron ausgeführt, wenn diese voraussichtlich mehr als zwei Sekunden dauern.

Der `return_timeout` Der Abfrageparameter ist mit asynchronen API-Aufrufen verfügbar und kann einen asynchronen Anruf synchron in den Abschluss konvertieren. Siehe "Asynchrone Verarbeitung mit dem Job-Objekt" Finden Sie weitere Informationen.

## Sicherheit

Die Sicherheit der REST-API basiert in erster Linie auf den vorhandenen Sicherheitsfunktionen von ONTAP. Die folgende Sicherheit wird von der API verwendet:

### Sicherheit In Transportschicht

Der gesamte über das Netzwerk zwischen dem Client und der logischen Schnittstelle von ONTAP gesendete Datenverkehr wird basierend auf den ONTAP Konfigurationseinstellungen in der Regel mit TLS verschlüsselt.

### Client-Authentifizierung

Die gleichen Authentifizierungsoptionen wie bei ONTAP System Manager und dem Network Manageability SDK können auch mit der ONTAP REST API verwendet werden.

### HTTP-Authentifizierung

Auf HTTP-Ebene gibt es beispielsweise beim direkten Zugriff auf die ONTAP-REST-API zwei Authentifizierungsoptionen wie unten beschrieben. In jedem Fall müssen Sie einen HTTP-Autorisierungsheader erstellen und diesen bei jeder Anforderung einschließen.

| Option | Beschreibung |
| --- | --- |
| HTTP-Basisauthentifizierung | Der ONTAP-Benutzername und das Passwort sind mit einem Doppelpunkt verbunden. Der String wird in base64 konvertiert und in den Request Header aufgenommen. |
| OAuth 2.0 | Ab ONTAP 9.14 können Sie ein Zugriffstoken von einem externen Autorisierungsserver anfordern und es als Inhabertoken in den Anforderungsheader aufnehmen. |

Weitere Informationen über OAuth 2.0 und die Implementierung in ONTAP finden Sie unter "Überblick über die Implementierung von ONTAP OAuth 2.0". Siehe auch "Die Nutzung der Workflows wird vorbereitet" Unten auf dieser Website.

**ONTAP-Autorisierung**

ONTAP implementiert ein rollenbasiertes Autorisierungsmodell. Das Konto, das Sie für den Zugriff auf die ONTAP REST-API- oder API-Dokumentationsseite verwenden, sollte über die entsprechende Berechtigung verfügen.

# Eingabevariablen, die eine API-Anforderung steuern

Sie können steuern, wie ein API-Aufruf über Parameter und Variablen verarbeitet wird, die in der HTTP-Anforderung festgelegt sind.

## HTTP-Methoden

Die von der ONTAP REST API unterstützte HTTP-Methoden sind in der folgenden Tabelle aufgeführt.

Nicht alle HTTP-Methoden sind an jedem REST-Endpunkt verfügbar. AUSSERDEM KÖNNEN PATCH und DELETE für eine Sammlung verwendet werden. Weitere Informationen finden Sie unter *Objektreferenzen und Zugang*.

| HTTP-Methode | Beschreibung |
|---|---|
| GET | Ruft Objekteigenschaften auf einer Ressourceninstanz oder -Sammlung ab. |
| POST | Erstellt eine neue Ressourceninstanz basierend auf der angegebenen Eingabe. |
| PATCH | Aktualisiert eine vorhandene Ressourceninstanz basierend auf den eingegebenen Eingaben. |
| Löschen | Löscht eine vorhandene Ressourceninstanz. |
| KOPF | Gibt eine GET-Anfrage effektiv aus, gibt aber nur die HTTP-Header zurück. |
| OPTIONEN | Legen Sie fest, welche HTTP-Methoden an einem bestimmten Endpunkt unterstützt werden. |

## Pfadvariablen

Der bei jedem REST-API-Aufruf verwendete Endpunktpfad kann verschiedene Kennungen enthalten. Jede ID entspricht einer bestimmten Ressourceninstanz. Beispiele sind Cluster-IDs und SVM-IDs.

## Anfragekopfzeilen

Sie müssen mehrere Header in die HTTP-Anfrage aufnehmen.

**Inhaltstyp**

Wenn der Anforderungsentext JSON enthält, muss dieser Header auf festgelegt werden `application/json`.

**Akzeptieren**

Diese Kopfzeile sollte auf gesetzt werden `application/hal+json`. Wenn sie stattdessen auf eingestellt

ist `application/json` Keiner der HAL-Links wird zurückgegeben, außer ein Link, der zum Abrufen des nächsten Stapels von Datensätzen benötigt wird. Wenn der Header etwas anderes außer diesen beiden Werten ist, ist der Standardwert des `content-type` Die Kopfzeile in der Antwort ist `application/hal+json`.

**Autorisierung**

Die grundlegende Authentifizierung muss mit dem Benutzernamen und dem Passwort als base64-Zeichenfolge codiert sein. Beispiel:

```
Authorization: Basic YWRtaW46cGV0ZXJzb24=.
```

## Text anfordern

Der Inhalt der Anfraentext variiert je nach Anruf. Der HTTP-Request-Text besteht aus einem der folgenden Elemente:

- JSON-Objekt mit Eingabevariablen
- Leeres JSON-Objekt

## Objekte filtern

Wenn Sie einen API-Aufruf mit der GET-Methode ausgeben, können Sie die zurückgegebenen Objekte anhand eines beliebigen Attributs mithilfe eines Abfrageparameters einschränken oder filtern.

**Analyse und Interpretation von Abfrageparametern**

Ein Satz von einem oder mehreren Parametern kann an die URL-Zeichenfolge angehängt werden, die nach dem beginnt `?` Zeichen. Wenn mehrere Parameter angegeben werden, werden die Abfrageparameter auf Basis des aufgeteilt `&` Zeichen. Jede Taste und jeder Wert im Parameter werden am geteilt = Zeichen.

Sie können beispielsweise einen exakten Wert angeben, der mit dem Gleichheitszeichen übereinstimmt:

```
<field>=<value>
```

Für eine komplexere Abfrage wird der zusätzliche Operator nach dem Gleichheitszeichen gesetzt. Um z. B. den Satz von Objekten auf der Grundlage eines bestimmten Felds auszuwählen, der größer oder gleich einem Wert ist, würde die Abfrage folgendermaßen lauten:

```
<field>=>=<value>
```

**Filteroperatoren**

Zusätzlich zu den oben genannten Beispielen stehen weitere Operatoren zur Verfügung, um Objekte über einen Wertebereich zurückzugeben. Eine Zusammenfassung der von der ONTAP-REST-API unterstützten Filteroperatoren ist in der folgenden Tabelle aufgeführt.

Nicht festgelegte Felder werden in der Regel von übereinstimmenden Abfragen ausgeschlossen.

| Operator | Beschreibung |
|---|---|
| = | Gleich |
| < | Kleiner als |

| | |
|---|---|
| > | Größer als |
| <= | Kleiner oder gleich |
| >= | Größer oder gleich |
| ! | Nicht gleich |
| * | Gierige Wildcard |

Sie können auch eine Sammlung von Objekten zurückgeben, basierend darauf, ob ein bestimmtes Feld über die festgelegt wurde oder nicht `null` Stichwort oder Negation `!null` Als Teil der Abfrage.

**Workflow-Beispiele**

Einige Beispiele aus den REST-API-Workflows auf dieser Site sind unten aufgeführt.

- "Festplatten auflisten"

  Filter basierend auf dem `state` Variable zur Auswahl der Ersatzfestplatten.

## Es werden bestimmte Objektfelder angefordert

Standardmäßig gibt die Ausgabe eines API-Aufrufs mit GET nur die Attribute zurück, die das Objekt oder die Objekte eindeutig identifizieren, zusammen mit einer HAL-Selbstverknüpfung. Dieser minimale Feldsatz dient als Schlüssel für jedes Objekt und variiert je nach Objekttyp. Sie können zusätzliche Objekteigenschaften mithilfe der auswählen `fields` Abfrageparameter auf folgende Weise:

- Allgemeine oder Standardfelder

  Angeben `fields=*` ` Zum Abrufen der am häufigsten verwendeten Objektfelder. Diese Felder werden normalerweise im lokalen Serverspeicher verwaltet oder erfordern nur wenig Verarbeitung für den Zugriff. Dies sind die gleichen Eigenschaften, die für ein Objekt zurückgegeben werden, nachdem GET mit einem URL-Pfadschlüssel (UUID) verwendet wurde.

- Alle Felder

  Angeben `fields=**` Zum Abrufen aller Objektfelder, einschließlich solcher, die für den Zugriff zusätzliche Serververarbeitung erforderlich sind.

- Benutzerdefinierte Feldauswahl

  Nutzung `fields=<field_name>` Um das genaue Feld anzugeben, das Sie wünschen. Wenn Sie mehrere Felder anfordern, müssen die Werte durch Kommas ohne Leerzeichen getrennt werden.

Als Best Practice sollten Sie immer die gewünschten Felder identifizieren. Sie sollten nur die gemeinsamen Felder oder alle Felder abrufen, wenn Sie dies benötigen. Welche Felder werden als allgemein klassifiziert und mit zurückgegeben `fields=*`, Wird von NetApp basierend auf interner Performance-Analyse ermittelt. Die Klassifizierung eines Felds kann sich in zukünftigen Releases ändern.

## Sortieren von Objekten im Ausgabungsset

Die Datensätze in einer Ressourcensammlung werden in der vom Objekt definierten Standardreihenfolge

zurückgegeben. Sie können die Bestellung über ändern `order_by` Abfrage-Parameter mit Feldname und Sortierrichtung wie folgt:

```
order_by=<field name> asc|desc
```

Sie können beispielsweise das Typfeld in absteigender Reihenfolge, gefolgt von id in aufsteigender Reihenfolge sortieren:

```
order_by=type desc, id asc
```

Beachten Sie Folgendes:

- Wenn Sie ein Sortierfeld angeben, aber keine Richtung angeben, werden die Werte in aufsteigender Reihenfolge sortiert.
- Wenn Sie mehrere Parameter eingeben, müssen Sie die Felder mit einem Komma trennen.

## Paginierung beim Abrufen von Objekten in einer Sammlung

Wenn ein API-Aufruf über GET auf eine Sammlung von Objekten desselben Typs zugreifen soll, versucht ONTAP, auf der Grundlage von zwei Einschränkungen so viele Objekte wie möglich zurückzugeben. Mit zusätzlichen Abfrageparametern auf der Anforderung können Sie jede dieser Einschränkungen steuern. Die erste Bedingung, die für eine bestimmte GET-Anforderung erreicht wurde, beendet die Anforderung und begrenzt damit die Anzahl der zurückgegebenen Datensätze.

Wenn eine Anfrage endet, bevor sie alle Objekte anführt, enthält die Antwort den Link, der zum Abrufen des nächsten Stapels von Datensätzen benötigt wird.

**Die Anzahl der Objekte wird begrenzt**

Standardmäßig gibt ONTAP maximal 10,000 Objekte für EINE GET-Anforderung aus. Sie können diese Begrenzung mit dem ändern `max_records` Abfrageparameter. Beispiel:

```
max_records=20
```

Die Anzahl der tatsächlich zurückgegebenen Objekte kann aufgrund der entsprechenden Zeitbeschränkung sowie der Gesamtanzahl der Objekte im System kleiner sein als die maximale Wirkung.

**Begrenzung der Zeit, die zum Abrufen der Objekte verwendet wird**

Standardmäßig gibt ONTAP so viele Objekte wie möglich innerhalb der für die GET-Anforderung zulässigen Zeit zurück. Die Standard-Zeitüberschreitung beträgt 15 Sekunden. Sie können diese Begrenzung mit dem ändern `return_timeout` Abfrageparameter. Beispiel:

```
return_timeout=5
```

Die Anzahl der tatsächlich zurückgegebenen Objekte kann aufgrund der damit verbundenen Beschränkung auf die Anzahl der Objekte sowie die Gesamtanzahl der Objekte im System kleiner sein als die maximal zulässige Anzahl.

**Verengung des Ergebnisset**

Bei Bedarf können Sie diese beiden Parameter mit zusätzlichen Abfrageparametern kombinieren, um den Ergebnissatz einzugrenzen. Im Folgenden werden z. B. bis zu 10 ems-Ereignisse zurückgegeben, die nach der angegebenen Zeit generiert wurden:

```
time=> 2018-04-04T15:41:29.140265Z&max_records=10
```

Sie können mehrere Anfragen zur Seite durch die Objekte ausgeben. Jeder nachfolgende API-Aufruf sollte einen neuen Zeitwert verwenden, der auf dem letzten Ereignis des letzten Ergebnisset basiert.

### Größeneigenschaften

Die bei einigen API-Aufrufen verwendeten Eingabewerte sowie bestimmte Abfrageparameter sind numerisch. Anstatt eine ganze Zahl in Byte bereitzustellen, können Sie optional ein Suffix wie in der folgenden Tabelle aufgeführt verwenden.

| Suffix | Beschreibung |
|---|---|
| KB | KB-Kilobyte (1024 Byte) oder Kibibyte |
| MB | MB Megabyte (KB x 1024 Byte) oder Mebibyte |
| GB | GB Gigabyte (MB x 1024 Byte) oder Gibibyte |
| TB | TB Terabyte (GB x 1024 Byte) oder Tebibyte |
| PB | PB (TB x 1024 Byte) oder Pebibyte |

**Verwandte Informationen**

- "Objektreferenzen und -Zugriff"

# Interpretation einer API-Antwort

Jede API-Anfrage generiert eine Antwort an den Client. Sie sollten die Antwort überprüfen, um festzustellen, ob sie erfolgreich war, und weitere Daten nach Bedarf abrufen.

### HTTP-Statuscode

Im Folgenden werden die von der ONTAP REST API verwendeten HTTP-Statuscodes beschrieben.

| Codieren | Grundsatz | Beschreibung |
|---|---|---|
| 200 | OK | Zeigt Erfolg für Anrufe an, die kein neues Objekt erstellen. |
| 201 | Erstellt | Ein Objekt wurde erfolgreich erstellt. Der Positionskopf in der Antwort enthält die eindeutige Kennung für das Objekt. |
| 202 | Akzeptiert | Ein Hintergrundjob wurde gestartet, um die Anforderung auszuführen, ist aber noch nicht abgeschlossen. |
| 400 | Schlechte Anfrage | Die Eingabe der Anfrage ist nicht erkannt oder nicht angemessen. |
| 401 | Nicht Autorisiert | Benutzerauthentifizierung fehlgeschlagen. |
| 403 | Verboten | Der Zugriff wird aufgrund eines Autorisierungsfehlers verweigert. |
| 404 | Nicht gefunden | Die Ressource, auf die in diesem Antrag verwiesen wird, ist nicht vorhanden. |

| Codieren | Grundsatz | Beschreibung |
|---|---|---|
| 405 | Methode nicht zulässig | Die HTTP-Methode in der Anforderung wird für die Ressource nicht unterstützt. |
| 409 | Konflikt | Der Versuch, ein Objekt zu erstellen, ist fehlgeschlagen, weil zunächst ein anderes Objekt erstellt werden muss oder das angeforderte Objekt bereits vorhanden ist. |
| 500 | Interner Fehler | Ein allgemeiner interner Fehler ist auf dem Server aufgetreten. |

## Antwortkopfzeilen

In der vom ONTAP erzeugten HTTP-Antwort sind mehrere Header enthalten.

**Standort**

Wenn ein Objekt erstellt wird, enthält die Standortkopfzeile die komplette URL zum neuen Objekt einschließlich der eindeutigen Kennung, die dem Objekt zugewiesen ist.

**Inhaltstyp**

Dies ist normalerweise der Fall `application/hal+json`.

## Antwortkörper

Der Inhalt des Antwortkörpers, der sich aus einer API-Anfrage ergibt, unterscheidet sich je nach Objekt, Verarbeitungstyp und Erfolg oder Misserfolg der Anforderung. Die Antwort wird immer in JSON gerendert.

- Einzelnes Objekt

  Je nach Anforderung kann ein einzelnes Objekt mit einer Reihe von Feldern zurückgegeben werden. Beispielsweise können Sie GET verwenden, um ausgewählte Eigenschaften eines Clusters mit der eindeutigen Kennung abzurufen.

- Mehrere Objekte

  Es können mehrere Objekte aus einer Ressourcensammlung zurückgegeben werden. In allen Fällen wird ein konsistentes Format verwendet, mit `num_records` Angabe der Anzahl der Datensätze und Datensätze, die ein Array der Objektinstanzen enthalten. Beispielsweise können Sie die in einem bestimmten Cluster definierten Nodes abrufen.

- Jobobjekt

  Wenn ein API-Aufruf asynchron verarbeitet wird, wird ein Job-Objekt zurückgegeben, das den Hintergrund-Task ankers. Beispielsweise wird die PATCH-Anfrage, die zum Aktualisieren der Cluster-Konfiguration verwendet wird, asynchron verarbeitet und ein Job-Objekt zurückgegeben.

- Fehlerobjekt

  Wenn ein Fehler auftritt, wird immer ein Fehlerobjekt zurückgegeben. Beispielsweise erhalten Sie einen Fehler beim Versuch, ein Feld zu ändern, das nicht für ein Cluster definiert ist.

- Leeres JSON-Objekt

  In bestimmten Fällen werden keine Daten zurückgegeben und der Antwortkörper enthält ein leeres JSON-Objekt.

## HAL-Verknüpfung

Die ONTAP-REST-API verwendet HAL als Mechanismus zur Unterstützung von Hypermedia als Engine of Application State (HATEOAS). Wenn ein Objekt oder Attribut zurückgegeben wird, das eine bestimmte Ressource identifiziert, wird auch ein HAL-codierter Link enthalten, mit dem Sie einfach weitere Details über die Ressource finden und ermitteln können.

## Fehler

Wenn ein Fehler auftritt, wird ein Fehlerobjekt im Antwortkörper zurückgegeben.

### Formatieren

Ein Fehlerobjekt hat das folgende Format:

```
"error": {
"message": "<string>",
"code": <integer>[,
"target": "<string>"]
}
```

Sie können den Codewert verwenden, um den allgemeinen Fehlertyp oder die allgemeine Fehlerkategorie zu bestimmen, und die Meldung, um den spezifischen Fehler zu ermitteln. Wenn verfügbar, enthält das Zielfeld die spezifische Benutzereingabe, die mit dem Fehler verknüpft ist.

### Allgemeine Fehlercodes

Die gängigen Fehlercodes werden in der folgenden Tabelle beschrieben. Spezifische API-Aufrufe können zusätzliche Fehlercodes enthalten.

| Codieren | | Beschreibung |
|---|---|---|
| 1 | 409 | Ein Objekt mit derselben Kennung ist bereits vorhanden. |
| 2 | 400 | Der Wert für ein Feld hat einen ungültigen Wert oder fehlt oder es wurde ein zusätzliches Feld angegeben. |
| 3 | 400 | Der Vorgang wird nicht unterstützt. |
| 4 | 405 | Ein Objekt mit der angegebenen Kennung wurde nicht gefunden. |
| 6 | 403 | Die Berechtigung zur Durchführung der Anforderung wird verweigert. |
| 8 | 409 | Die Ressource wird verwendet. |

# Asynchrone Verarbeitung mit dem Job-Objekt

Nachdem eine API-Anfrage ausgegeben wurde, die für die asynchrone Ausführung

ausgelegt ist, wird immer ein Jobobjekt erstellt und an den Anrufer zurückgegeben. Der Job beschreibt und Anker eine Hintergrundaufgabe, die die Anforderung verarbeitet. Abhängig vom HTTP-Statuscode müssen Sie den Status des Jobs abrufen, um festzustellen, ob die Anforderung erfolgreich war.

Siehe "API-Referenz" Ermitteln, welche API-Aufrufe asynchron ausgeführt werden sollen.

## Kontrolle der Verarbeitung einer Anfrage

Sie können das verwenden `return_timeout` Abfrageparameter zur Steuerung der Verarbeitung eines asynchronen API-Aufrufs. Bei Verwendung dieses Parameters sind zwei mögliche Ergebnisse möglich.

**Der Timer läuft ab, bevor der Antrag abgeschlossen ist**

Bei gültigen Anfragen gibt ONTAP zusammen mit dem Jobobjekt einen HTTP-Statuscode von 202 zurück. Sie müssen den Status des Jobs abrufen, um festzustellen, ob die Anforderung erfolgreich abgeschlossen wurde.

**Die Anforderung ist abgeschlossen, bevor der Timer abläuft**

Wenn die Anfrage gültig ist und erfolgreich abgeschlossen wird, bevor die Zeit abläuft, gibt ONTAP zusammen mit dem Jobobjekt einen HTTP-Statuscode 200 zurück. Da die Anforderung synchron abgeschlossen wird, wie vom 200 angegeben, müssen Sie den Job-Status nicht abrufen.

Der Standardwert für das `return_timeout` Der Parameter beträgt null Sekunden. Wenn Sie den Parameter nicht angeben, wird der HTTP-Statuscode 202 immer für eine gültige Anfrage zurückgegeben.

## Abfragen des mit einer API-Anforderung verknüpften Jobobjekts

Das in der HTTP-Antwort zurückgegebene Job-Objekt enthält mehrere Eigenschaften. Sie können die Statuseigenschaft in einem nachfolgenden API-Aufruf abfragen, um festzustellen, ob die Anforderung erfolgreich abgeschlossen wurde. Ein Job-Objekt befindet sich immer in einem der folgenden Zustände:

**Nicht-Terminal-Status**

- Warteschlange
- Wird Ausgeführt
- Angehalten

**Terminalzustände**

- Erfolg
- Ausfall

## Allgemeines Verfahren für die Ausgabe einer asynchronen Anfrage

Sie können den folgenden grundlegenden Vorgang verwenden, um einen asynchronen API-Aufruf abzuschließen. In diesem Beispiel wird vorausgesetzt, dass die `return_timeout` Parameter wird nicht verwendet oder die Zeit läuft ab, bevor der Hintergrundjob abgeschlossen ist.

1. Geben Sie einen API-Aufruf aus, der asynchron ausgeführt wird.
2. Sie erhalten eine HTTP-Antwort 202, die auf die Annahme einer gültigen Anfrage hinweist.

3. Extrahieren Sie die Kennung für das Job-Objekt aus dem Antwortkörper.
4. Führen Sie in einem zeitlich festgelegten Regelkreis in jedem Zyklus folgende Schritte aus:
   a. Abrufen des aktuellen Status des Jobs.
   b. Wenn sich der Job nicht im Terminalzustand befindet, führen Sie die Schleife erneut aus.
5. Beenden Sie, wenn der Job einen Terminalstatus erreicht (Erfolg, Fehler).

**Verwandte Informationen**

- "Cluster-Kontakt aktualisieren"
- "Job-Instanz abrufen"

# Objektreferenzen und -Zugriff

Auf die über die ONTAP REST-API offengelegten Ressourceninstanzen oder Objekte kann auf unterschiedliche Weise zugegriffen werden.

## Objektzugriffspfade

Auf hoher Ebene gibt es zwei Pfadtypen für den Zugriff auf ein Objekt:

- Primär

  Das Objekt ist das primäre oder direkte Ziel des API-Aufrufs.

- Im Ausland

  Das Objekt ist nicht die primäre Referenz des API-Aufrufs, sondern ist mit dem primären Objekt verknüpft. Es handelt sich daher um ein fremdes oder nachgeschalteter Objekt und wird durch ein Feld im primären Objekt referenziert.

## Zugriff auf ein Objekt mithilfe der UUID

Jedem Objekt wird eine eindeutige ID bei der Erstellung zugewiesen. Dies ist in den meisten Fällen eine 128-Bit-UUID. Die zugewiesenen UUID-Werte sind unveränderlich und werden innerhalb von ONTAP intern zum Zugriff und Management der Ressourcen verwendet. Aus diesem Grund bietet die UUID im Allgemeinen die schnellste und stabilste Art, auf Objekte zuzugreifen.

Für viele Ressourcentypen kann ein UUID-Wert als Teil des Pfadschlüssels in der URL bereitgestellt werden, um auf ein bestimmtes Objekt zuzugreifen. Beispielsweise können Sie Folgendes verwenden, um auf eine Node-Instanz zuzugreifen: `` `/cluster/nodes/{uuid}` ``

## Zugriff auf ein Objekt mithilfe einer Objekteigenschaft

Zusätzlich zu einer UUID können Sie auch mithilfe einer Objekteigenschaft auf ein Objekt zugreifen. In den meisten Fällen ist es bequem, die Namenseigenschaft zu verwenden. Sie können beispielsweise den folgenden Abfrageparameter in der URL-Zeichenfolge verwenden, um auf eine Node-Instanz mit ihrem Namen zuzugreifen: `/cluster/nodes?name=node_one`. Zusätzlich zu einem Abfrageparameter kann über eine Eigenschaft im primären Objekt auf ein fremdes Objekt zugegriffen werden.

Während Sie den Namen oder eine andere Eigenschaft für den Zugriff auf ein Objekt anstelle der UUID verwenden können, gibt es einige mögliche Nachteile:

- Das Namensfeld ist nicht unveränderlich und kann geändert werden. Wenn der Name eines Objekts vor dem Zugriff auf ein Objekt geändert wird, wird das falsche Objekt zurückgegeben oder ein Objektzugriffsfehler schlägt fehl.

  Dieses Problem kann mit EINER POST- oder PATCH-Methode auf einem fremden Objekt oder mit EINER GET-Methode auf einem primären Objekt auftreten.

- ONTAP muss das Namensfeld in die entsprechende UUID übersetzen. Diese Art von indirekten Zugriff kann zu einem Performance-Problem werden.

Insbesondere kann eine Performance-Verschlechterung erzielt werden, wenn eine oder mehrere der folgenden zutrifft:

- GET-Methode wird verwendet
- Auf eine große Sammlung von Objekten wird zugegriffen
- Es wird eine komplexe oder aufwändige Abfrage verwendet

### Der Kontext zwischen Cluster und SVM

Es gibt mehrere REST-Endpunkte, die sowohl ein Cluster als auch eine SVM unterstützen. Wenn Sie einen dieser Endpunkte verwenden, können Sie den Kontext des API-Aufrufs über das anzeigen `scope=[svm|cluster]` Wert: Beispiele für Endpunkte, die einen dualen Kontext unterstützen, sind IP-Schnittstellen und Sicherheitsrollen.

Der Scope-Wert hat einen Standardwert, der auf den Eigenschaften basiert, die für jeden API-Aufruf bereitgestellt werden.

### VERWENDEN VON PATCHES und LÖSCHEN einer Sammlung von Objekten

Jeder REST-Endpunkt, der PATCH oder LÖSCHUNG auf einer Ressourceninstanz unterstützt, unterstützt auch dieselbe Methode bei einer Objektsammlung. Die einzige Voraussetzung ist, dass mindestens ein Feld über einen Abfrageparameter im URL-String bereitgestellt werden muss. Bei der Ausgabe eines PATCHES oder BEIM LÖSCHEN einer Sammlung entspricht dies dem internen Verfahren:

- Abfrage-basierte ABRUFEN, um die Sammlung abzurufen
- Serielle Sequenz von PATCHES oder LÖSCHANRUFE für jedes Objekt in der Sammlung

Die Zeitdauer für den Vorgang kann von eingestellt werden `return_timeout` Standardmäßig 15 Sekunden. Wenn die Antwort vor dem Timeout nicht abgeschlossen wurde, enthält sie einen Link zum nächsten Objekt. Sie müssen dieselbe HTTP-Methode über den nächsten Link erneut ausgeben, um den Vorgang fortzusetzen.

# Performance-Metriken für Storage-Ressourcen

ONTAP sammelt Performance-Kennzahlen zu ausgewählten SVM-Storage-Objekten und -Protokollen und meldet diese Informationen über DIE REST-API. Sie können diese Daten für die Überwachung der Performance eines ONTAP Systems verwenden.

Für ein Storage-Objekt oder ein bestimmtes Protokoll fallen die Performance-Daten in drei Kategorien:

- IOPS

- Latenz
- Durchsatz

Innerhalb jeder Kategorie steht ein oder mehrere der folgenden Datentypen zur Verfügung:

- Lesen ®
- Schreiben (W)
- Sonstiges (O)
- Gesamt (T)

Die folgende Tabelle fasst die Performance-Daten zusammen, die über die ONTAP REST API verfügbar sind, einschließlich des Release, sobald sie hinzugefügt wurde. Weitere Informationen finden Sie auf der REST-API-Online-Dokumentationsseite Ihres ONTAP Systems.

| Storage-Objekt oder -Protokoll | IOPS | Latenz | Durchsatz | Version von ONTAP |
|---|---|---|---|---|
| Ethernet-Anschluss | Keine Angabe | Keine Angabe | RWT | 9.8 |
| FC-Port | RWOT | RWOT | RWT | 9.8 |
| IP-Schnittstelle | Keine Angabe | Keine Angabe | RWT | 9.8 |
| FC-Schnittstelle | RWOT | RWOT | RWT | 9.8 |
| NVMe-Namespace | RWOT | RWOT | RWOT | 9.8 |
| Qtree-Statistiken | RAW-RWOT | Keine Angabe | RAW-RWOT | 9.8 |
| Volume FlexCache | RWOT | RWOT | RWT | 9.8 |
| Node – Prozessnutzung | Prozessnutzung als numerischer Wert | Prozessnutzung als numerischer Wert | Prozessnutzung als numerischer Wert | 9.8 |
| Cloud Volume | RWOT | RWOT | Nicht applierbar | 9.7 |
| LUN | RWOT | RWOT | RWOT | 9.7 |
| Aggregat | RWOT | RWOT | RWOT | 9.7 |
| NFS-Protokoll der SVM | RWOT | RWOT | RWT | 9.7 |
| CIFS-Protokoll für SVM | RWOT | RWOT | RWT | 9.7 |
| FCP-Protokoll der SVM | RWOT | RWOT | RWT | 9.7 |
| ISCSI-Protokoll der SVM | RWOT | RWOT | RWT | 9.7 |
| NVMe-Protokoll der SVM | RWOT | RWOT | RWT | 9.7 |
| Cluster | RWOT | RWOT | RWOT | 9.6 |
| Volumes | RWOT | RWOT | RWOT | 9.6 |