HITACHI
Inspire the Next

Hitachi Content Platform

HCP Metadata Query API Reference

Hitachi Data Systems

MK-91ARC032-05

Contents

Preface

This book describes the **Hitachi Content Platform** (**HCP**) **metadata query API**. This RESTful HTTP API enables you to query namespaces programatically for objects that satisfy criteria you specify. The book explains how to construct and perform queries and describes query results. It also contains several examples, which you can use as models for your own queries.

Note: Throughout this book, the word *Unix* is used to represent all UNIX®-like operating systems (such as UNIX itself or Linux®).

Intended audience

This book is intended for people who want to search programmatically for objects in namespaces. It assumes you have a working knowledge of HCP concepts and the HTTP protocol.

Product version

This book applies to release 7.1 of HCP.

Syntax notation

The table below describes the conventions used for the syntax of commands, expressions, URLs, and object names in this book.

Notation	Meaning	Example
boldface	Type exactly as it appears in the syntax (if the context is case insensitive, you can vary the case of the letters you type)	This book shows: **http://admin.***hcp-domain-name*/**query** You enter: http://admin.hcp.example.com/query
italics	Replace with a value of the indicated type	
|	Vertical bar — Choose one of the elements on either side of the bar, but not both	This book shows: **<verbose>**(**true**|**false**)**</verbose>** You enter: <verbose>true</verbose> or: <verbose>false</verbose>
[ ]	Square brackets — Include none, one, or more of the elements between the brackets	This book shows: **http**[**s**]**://***tenant-name*. *hcp-domain-name*/**query** You enter: http://europe.hcp.example.com/query or: https://europe.hcp.example.com/query
( )	Parentheses — Include exactly one of the elements between the parentheses	This book shows: (**asc**|**desc**) You enter: asc or: desc
-path	Replace with a directory path with no file or object name	This book shows: **<directory>***directory-path***</directory>** You enter: <directory>/customers/widgetco/order</directory>
...	Ellipsis — Optionally, repeat the preceding parameter as many times as needed	This book shows: [*object-property-value* **+(asc**|**desc)**]**,...** You enter: size+asc, namespace+desc

Related documents

The following documents contain additional information about Hitachi Content Platform:

- *Administering HCP* — This book explains how to use an HCP system to monitor and manage a digital object repository. It discusses the capabilities of the system, as well as its hardware and software components. The book presents both the concepts and instructions

you need to configure the system, including creating the tenants that administer access to the repository. It also covers the processes that maintain the integrity and security of the repository contents.

- *Managing a Tenant and Its Namespaces* — This book contains complete information for managing the HCP tenants and namespaces created in an HCP system. It provides instructions for creating namespaces, setting up user accounts, configuring the protocols that allow access to namespaces, managing search and indexing, and downloading installation files for HCP Data Migrator. It also explains how to work with retention classes and the privileged delete functionality.

- *Managing the Default Tenant and Namespace* — This book contains complete information for managing the default tenant and namespace in an HCP system. It provides instructions for changing tenant and namespace settings, configuring the protocols that allow access to the namespace, managing search and indexing, and downloading installation files for HCP Data Migrator. It also explains how to work with retention classes and the privileged delete functionality.

- *Replicating Tenants and Namespaces* — This book covers all aspects of tenant and namespace replication. Replication is the process of keeping selected tenants and namespaces in two or more HCP systems in sync with each other to ensure data availability and enable disaster recovery. The book describes how replication works, contains instructions for working with replication links, and explains how to manage and monitor the replication process.

- *HCP Management API Reference* — This book contains the information you need to use the HCP management API. This RESTful HTTP API enables you to create and manage tenants and namespaces programmatically. The book explains how to use the API to access an HCP system, specify resources, and update and retrieve resource properties.

- *Using a Namespace* — This book describes the properties of objects in HCP namespaces. It provides instructions for accessing namespaces by using the HTTP, WebDAV, CIFS, and NFS protocols for the purpose of storing, retrieving, and deleting objects, as well as changing object metadata such as retention and shred settings. It also explains how to manage namespace content and view namespace information in the Namespace Browser.

- *Using the HCP HS3 API* — This book contains the information you need to use the HCP HS3 API. This S3™-compatible, RESTful, HTTP-based API enables you to work with buckets and objects in HCP. The book

introduces the HCP concepts you need to understand in order to use HS3 effectively and contains instructions and examples for each of the bucket and object operations you can perform with HS3.

- *Using the HCP OpenStack Swift API* — This book contains the information you need to use the HCP OpenStack Swift API. This S3™-compatible, RESTful, HTTP-based API enables you to work with containers and objects in HCP. The book introduces the HCP concepts you need to understand in order to use HSwift effectively and contains instructions and examples for each of the container and object operations you can perform with HSwift.

- *Using the Default Namespace* — This book describes the file system HCP uses to present the contents of the default namespace. It provides instructions for accessing the namespace by using the HCP-supported protocols for the purpose of storing, retrieving, and deleting objects, as well as changing object metadata such as retention and shred settings.

- *Searching Namespaces* — This book describes the HCP Search Console (also called the Metadata Query Engine Console). It explains how to use the Console to search namespaces for objects that satisfy criteria you specify. It also explains how to manage and manipulate queries and search results. The book contains many examples, which you can use as models for your own searches.

- *Using HCP Data Migrator* — This book contains the information you need to install and use HCP Data Migrator (HCP-DM), a utility that works with HCP. This utility enables you to copy data between local file systems, namespaces in HCP, and earlier HCAP archives. It also supports bulk delete operations and bulk operations to change object metadata. Additionally, it supports associating custom metadata and ACLs with individual objects. The book describes both the interactive window-based interface and the set of command-line tools included in HCP-DM.

- *Installing an HCP System* — This book provides the information you need to install the software for a new HCP system. It explains what you need to know to successfully configure the system and contains step-by-step instructions for the installation procedure.

- *Deploying an HCP-VM System* — This book contains all the information you need to install and configure an HCP-VM system. The book also includes requirements and guidelines for configuring the VMWare® environment in which the system is installed.

- *Third-Party Licenses and Copyrights* — This book contains copyright and license information for third-party software distributed with or embedded in HCP.

- *HCP-DM Third-Party Licenses and Copyrights* — This book contains copyright and license information for third-party software distributed with or embedded in HCP Data Migrator.

- *Installing an HCP SAIN System — Final On-site Setup* — This book contains instructions for deploying an assembled and configured single-rack HCP SAIN system at a customer site. It explains how to make the necessary physical connections and reconfigure the system for the customer computing environment. It also contains instructions for configuring Hi-Track® Monitor to monitor the nodes in an HCP system.

- *Installing an HCP RAIN System — Final On-site Setup* — This book contains instructions for deploying an assembled and configured HCP RAIN system at a customer site. It explains how to make the necessary physical connections and reconfigure the system for the customer computing environment. The book also provides instructions for assembling the components of an HCP RAIN system that was ordered without a rack and for configuring Hi-Track Monitor to monitor the nodes in an HCP system.

Getting help

The Hitachi Data Systems® customer support staff is available 24 hours a day, seven days a week. If you need technical support, call:

- United States: (800) 446-0744

- Outside the United States: (858) 547-4526

Note: If you purchased HCP from a third party, please contact your authorized service provider.

Comments

Please send us your comments on this document:

HCPDocumentationFeedback@hds.com

Include the document title, number, and revision, and refer to specific sections and paragraphs whenever possible. All comments become the property of Hitachi Data Systems.

Thank you!

1

Introduction to the HCP metadata query API

The HCP metadata query API is a RESTful HTTP API that lets you query HCP for objects that meet specific criteria. In response to a query, HCP returns metadata for the matching objects. With the metadata query API, you can query not only for objects currently in the repository but also for information about objects that have been deleted from the repository.

This chapter:

- Describes what you can do with the metadata query API
- Introduces the two types of queries you can run
- Describes query results
- Explains how to use paged queries to manage result sets
- Describes the effects of object indexing on query results

To learn about objects and object metadata, see *Using a Namespace* or *Using the Default Namespace*.

About the metadata query API

The HCP metadata query API lets you query namespaces for objects that match criteria you specify. Query criteria can be based on system metadata, custom metadata, ACLs, and operations performed on objects. The API does not support queries based on object content.

In response to a query, HCP returns metadata for objects that match query criteria. It does not return object data.

The metadata query API supports two types of queries, **object-based queries** and **operation-based queries**. For more information on these types of queries, see "Types of queries" on page 3.

A single query can return metadata for objects in multiple namespaces, including a combination of HCP namespaces and the default namespace. For HCP namespaces that support versioning, operation-based queries can return metadata for both current and old versions of objects.

To support object-based queries, HCP maintains an index of objects in the repository. For more information on this index, see "Object index" on page 6.

To access HCP through the metadata query API, you use the HTTP **POST** method. With this method, you specify query criteria in the request body. In the request body you also specify what information you want in the query results.

The API accepts query criteria in XML or JSON format and can return results in either format. For example, you could use XML to specify the query criteria and request that the response be JSON.

Note: This book uses the term *entry* to refer to an XML element and the equivalent JSON object and the term *property* for an XML attribute or the equivalent JSON name/value pair.

Because a large number of matching objects can result in a very large response, the metadata query API lets you limit the number of results returned for a single request. You can retrieve metadata for all the matching objects by using multiple requests. This process is called using a **paged query**. For more information on paged queries, see "Paged queries" on page 5.

Types of queries

The metadata query API supports two types of queries: object-based queries and operation-based queries. These query types have different request formats and return different information about objects in the result set. However, they have similar response formats.

Object-based queries

Object-based queries search for objects currently in the repository based on any combination of system metadata, object paths, custom metadata that's well-formed XML, ACLs, and **content properties**. (For information on content properties, see "Content properties" on page 7.) With object-based queries, you use a robust query language to construct query criteria.

In response to an object-based query, HCP returns a set of results, each of which identifies an object and contains metadata for the object. With object-based queries, you can specify sort criteria to manage the order in which results are returned. You can specify facet criteria to return summary information about object properties that appear in the result set.

Operation-based queries

Operation-based queries search for objects based on any combination of create, delete, and disposition operations and, for HCP namespaces that support versioning, purge and prune operations. Operation-based queries are useful for applications that need to track changes to namespace content.

In response to an operation-based query, HCP returns a set of **operation records**, each of which identifies an object and an operation on the object and contains additional metadata for the object. For more information on operation records, see "Operation-based query results" on page 4.

Query results

By default, for both types of queries, HCP returns only basic information about the objects that meet the query criteria. This information includes the object URL, the version ID, the operation type, and the change time.

If you specify a **verbose** entry with a value of **true** in the request body, HCP returns complete system metadata for the object or operation. If you aren't interested in the complete system metadata, you can specify the **objectProperties** entry with only the system metadata you want. For a description of all the system metadata you can request, see "Object properties" on page 52.

Object-based query results

Object-based queries return information about objects that currently exist in the repository. For objects with multiple versions, these queries return information only for the current version.

Object-based queries return information only about objects that have been indexed. For more information on the index that supports object-based queries, see "Object index" on page 6.

Operation-based query results

HCP maintains records of object creation, deletion, disposition, prune, and purge operations (also called **transactions**). These records can be retrieved through operation-based queries. The HCP system configuration determines how long HCP keeps deletion, disposition, prune, and purge records. HCP keeps creation records for as long as the object exists in the repository.

Each record has a change time. For creation records, this is the time the object was last modified. For deletion, disposition, prune, and purge records, the change time identifies the time of the operation.

Records returned while versioning is enabled

If versioning is enabled for an HCP namespace, the types of records that are returned by an operation-based query depend on the query request parameters. However, the following the rules determine which operation records can be returned:

- HCP returns a creation record for the current version of an object, as long as this version is not a deleted version.
- HCP returns creation records for old versions of an object.
- HCP returns creation records for versions of both deleted objects and disposed objects.
- HCP returns a single purge record for each purge operation. It does not return records for the individual versions of the purged object.

- HCP returns deletion, disposition, prune, and purge records until it removes them from the system.

Records returned while versioning is disabled

If you create and then delete an object while versioning is disabled, HCP keeps only the deletion record and not the creation record. Operation-based queries return the deletion record until HCP removes that record from the system.

If you create an object and then HCP disposes of that object while versioning is disabled, HCP keeps only the disposition record and not the creation record. Operation-based queries return the disposition record until HCP removes that record from the system.

If versioning was enabled at an earlier time but is no longer enabled, operation-based queries continue to return records of all operations performed during that time according to the rules listed in "Records returned while versioning is enabled" above. If you delete an object while versioning is disabled or if HCP disposes of an object while versioning is disabled, operation-based queries do not return any creation records for that object, regardless of whether versioning was enabled when it was created.

Paged queries

With paged queries, you issue multiple requests that each retrieve a limited number of results. You would use a paged query, for example, if:

- The size of the response to a single request would reduce the efficiency of the client. In this situation, you can use a paged query to prevent overloading the client. The client can process the results in each response before requesting additional data.

- The application issuing the query handles a limited number of objects at a time. For example, an application that lists a given number of objects at a time on a web page would use a paged query in which each request returned that number of results.

The criteria for paged queries differ between object-based queries and operation-based queries. For information on the criteria for paged queries:

- With object-based queries, see "Paged queries with object-based requests" on page 42

- With operation-based queries, see "Paged queries with operation-based requests" on page 51

Object index

To support object-based queries, HCP maintains an index of objects in the repository. This index is based on object paths, system metadata, custom metadata that's well formed XML, and ACLs.

Namespace indexing

Indexing is enabled on a per-namespace basis. If a namespace is not indexed, object-based queries do not return results for objects in the namespace.

HCP periodically checks indexable namespaces for new objects and for objects with metadata that has changed since the last check. When it finds new or changed information, it updates the index. The amount of time HCP takes to update the index depends on the amount of information to be indexed. New or changed information is not reflected in the results of object-based queries until the information is indexed.

Indexing of custom metadata can be configured in these ways:

- Specific content properties can be indexed. For information on content properties see "Content properties" below.
- Specific annotations can be excluded from being indexed. An annotation is a discrete unit of custom metadata
- Custom metadata contents can be optionally indexed for full-text searching.

If indexing of custom metadata is enabled for a namespace, these rules determine whether custom metadata is indexed for an object:

- The custom metadata must be well-formed XML
- The custom metadata must be smaller than one MB.
- The object must have an index setting of **true**. For more information on index settings, see "Object properties" on page 52.
- If custom metadata is not indexed for an object, object-based queries that are based on custom metadata do not return results for that object.

Content properties

A content property is a named construct used to extract an element or attribute value from custom metadata that's well-formed XML. Each content property has a data type that determines how the property values are treated when indexing and searching.

A content property is defined as either single-valued or multivalued. A multivalued property can extract the values of multiple occurrences of the same element or attribute from the XML.

The XML below shows XML elements with multiple occurrences of two elements, **date** and **rank** within the element **WeeklyRank**.

```
<record>
    <weeklyRank>
        <date> dd/MM/yyyy </date>
        <rank> (rank) </rank>
    </weeklyRank>
    <weeklyRank>
        <date> dd/MM/yyyy </date>
        <rank> (rank) </rank>
    </weeklyRank>
    <weeklyRank>
        <date> dd/MM/yyyy </date>
        <rank> (rank) </rank>
    </weeklyRank>
</record>
```

If the **WeekyRank** object property specifies the record/weeklyRank/rank entry in the XML, the property is multivalued.

2

Access and authentication

With the HCP metadata query API, each request you make must specify a URL that represents an HCP tenant, the default tenant, or all tenants to which system-level users have access. Each request must also include the credentials for the user account you're using to access namespaces through the metadata query API. Your user account determines which namespaces you can access.

This chapter describes request URLs and explains how to include account credentials in a metadata query API request.

The examples in this book use cURL and Python with PycURL, a Python interface that uses the libcurl library. cURL and PycURL are both freely available open-source software. You can download them from http://curl.haxx.se.

Request URL

The URL format in a metadata query API request depends on whether you use a hostname or IP address to connect to the HCP system and on the namespaces you want to query.

Connecting using a hostname

When connecting to HCP using a hostname, the URL format you use depends on the namespaces you are querying:

- To query one or more namespaces owned by an HCP tenant, use this format:

  http[**s**]**://***hcp-tenant-name*.*hcp-domain-name*/**query**

  For example:

  https://europe.hcp.example.com/query

  To use this format, you need either a tenant-level user account or, if the tenant has granted system-level users administrative access to itself, a system-level user account. In either case, the account must be configured to allow use of the metadata query API.

  When you use a tenant-level user account, HCP returns results only for objects in namespaces for which the tenant-level user has search permission.

  Unlike with requests to the `/rest` interface, you do not specify a namespace in this URL. For information on that interface, see *"Using a Namespace"*.

- To query only the default namespace, use this format:

  https://default.*hcp-domain-name*/**query**

  For example:

  https://default.hcp.example.com/query

  To use this format, you need a system-level user account that's configured to allow the user to use the metadata query API.

For this URL format, you need to use HTTP with SSL security (HTTPS). If the query specifies HTTP instead of HTTPS in the URL, HCP returns a 403 (Forbidden) error.

- To query the entire repository (that is, both the default namespace and all namespaces owned by each tenant that has granted system-level users administrative access to itself), use this format:

  https://admin.*hcp-domain-name***/query**

  For example:

  https://admin.hcp.example.com/query

  To use this format, you need a system-level user account that 's configured to allow use of the metadata query API.

  For this URL format, you need to use HTTP with SSL security (HTTPS). If the query specifies HTTP instead of HTTPS in the URL, HCP returns a 403 (Forbidden) error.

The following considerations apply to these URLs:

- The URL must specify **query**, in all lowercase, as the first element following the hostname in the URL.

- If the URL specifies HTTPS and the HCP system uses a self-signed SSL server certificate, the request must include an instruction not to perform SSL certificate verification. With cURL, you do this by including the **-k** option in the request command line. In Python with PycURL, you do this by setting the **SSL_VERIFYPEER** option to **false**.

Connecting using an IP address

The core hardware for an HCP system consists of servers, called **nodes**, that are networked together. When you access an HCP system, your point of access is an individual node. Typically, you let HCP choose the node on which to process a metadata query API request. You can, however, use an IP address in the URL to access the system on a specific node. To do this, you replace the fully qualified hostname in the URL with the IP address of the node you want:

https://*node-ip-address***/query**

With this URL format, you can provide an HTTP Host header that specifies a fully qualified hostname for a tenant or the entire repository. The hostname format you use depends on the namespaces you want to query:

- To query namespaces owned by an HCP tenant, use this format:

  hcp-tenant-name.hcp-domain-name

- To query only the default namespace, use this format:

  default.*hcp-domain-name*

- To query the entire repository, use this format:

  admin.*hcp-domain-name*

If you omit the Host header, the request queries the entire respository.

Note: The Host header is required when you are performing an operation-based query and the request body specifies a namespace.

With cURL, you use the **-H** option to provide the Host header. For example:

```
-H "Host: finance.hcp.example.com"
```

In Python with PycURL, you do this with the **HTTPHEADER** option. For example:

```
curl.setopt(pycurl.HTTPHEADER, ["HOST: default.hcp.example.com"])
```

When using an IP address in a URL, you need to use HTTP with SSL security.

For information on when to use an IP address for access to the HCP system, see *Using a Namespace*.

Note for tenant-level users: If you don't know the IP addresses for the HCP system, contact your HCP system administrator.

Connecting using a hosts file

All operating systems have a `hosts` file that contains mappings from hostnames to IP addresses. If the HCP system does not support DNS, you can use this file to enable access to tenants by hostname.

The location of the hosts file depends on the client operating system:

- On Windows®, by default: c:\windows\system32\drivers\etc\hosts
- On Unix: /etc/hosts
- On Mac OS® X: /private/etc/host

Hostname mappings

Each entry in a hosts file maps one or more fully qualified hostnames to a single IP address. For example, the entry below maps the hostname of the europe tenant in the HCP system named hcp.example.com to the IP address 192.168.210.16:

```
192.168.210.16      europe.hcp.example.com
```

The following considerations apply to hosts file entries:

- Each entry must appear on a separate line.
- Multiple hostnames in a single line must be separated by white space. With some versions of Windows, these must be single spaces.
- At the system-level, the fully qualified hostname includes *admin*.
- Each hostname can map to multiple IP addresses.

You can include comments in a hosts file either on separate lines or following a mapping on the same line. Each comment must start with a number sign (#). Blank lines are ignored.

Note for tenant-level users: If you don't know the IP addresses for the HCP system, contact your HCP system administrator.

Hostname mapping considerations

You can map a hostname to any number of IP addresses. The way multiple mappings are used depends on the client platform. For information on how your client handles multiple mappings in a `hosts` file, see your client documentation.

If any of the HCP nodes listed in the `hosts` file are unavailable, timeouts may occur when you use a `hosts` file to access the System Management Console.

Sample hosts file entries

Here's a sample `hosts` file that contains mappings for the repository as a whole and the europe tenant:

```
# HCP system-level mappings
192.168.210.16 admin.hcp.example.com
192.168.210.17 admin.hcp.example.com

# tenant-level mappings
192.168.210.16 europe.hcp.example.com
192.168.210.17 europe.hcp.example.com
```

Authentication

To use the metadata query management API, you need either a system-level or tenant-level user account that's defined in HCP. If HCP is configured to support Windows Active Directory® (AD), applications can also use an AD user account that HCP recognizes to access HCP through the metadata query API.

With each metadata query API request, you need to provide your account credentials in the form of a username and password. If you do not provide credentials or provide invalid credentials, HCP responds with a 403 (Forbidden) error message.

To provide credentials in a metadata query API request, you specify an **authentication token** in an HTTP Authorization request header.

HCP also accepts credentials provided in an **hcp-ns-auth** cookie. However, this method of providing credentials is being deprecated and should not be used in new applications.

Note: To use a recognized AD user account for access to HCP through the metadata query API, applications must use the SPNEGO protocol to negotiate the AD user authentication themselves. For more information on SPNEGO, see http://tools.ietf.org/html/rfc4559.

Authentication token

An authentication token consists of a username in Base64-encoded format and a password that's hashed using the MD5 hash algorithm, separated by a colon, like this:

base64-encoded-username:*md5-hashed-password*

For example, here's the token for the Base64-encoded username *myuser* and the MD5-hashed password *p2Ss#0rd*:

bXl1c2Vy:6ecaf581f6879c9a14ca6b76ff2a6b15

The GNU Core Utilities include the **base64** and **md5sum** commands, which convert text to Base64-encoded and MD5-hashed values, respectively. With these commands, a line such as this creates the required token:

```
echo `echo -n username | base64`:`echo -n password | md5sum` |
     awk '{print $1}'
```

The character before *echo*, before and after the colon, and following *md5sum* is a backtick (or grave accent). The **-n** option in the **echo** command prevents the command from appending a newline character to the output. This is required to ensure correct Base64 and MD5 values.

For more information on the GNU Core Utilities, see http://www.gnu.org/software/coreutils/.

Other tools that generate Base64-encoded and MD5-hashed values are available for download on the web. For security reasons, do not use interactive public web-based tools to generate these values.

Authorization header

You use the HTTP Authorization request header to provide the authentication token for a metadata query API request. The value of this header is **HCP** followed by the authentication token, in this format:

Authorization: HCP *authentication-token*

For example, here's the Authorization header for a user named *myuser* and password *p2Ss#0rd*:

Authorization: HCP bXl1c2Vy:6ecaf581f6879c9a14ca6b76ff2a6b15

Specifying the Authorization header with cURL

With cURL, you use the **-H** option to specify a header. So, for example, a query API request for objects in namespaces owned by the tenant europe might look like this:

```
curl -i -k "https://europe.hcp.example.com/query"
    -H "Authorization: HCP bGdyZWVu:2a9d119df47ff993b662a8ef36f9ea20"
    -H "Content-Type:application/xml -H "Accept: application/xml"
    -d @queryRequest.xml
```

For more information on the format for metadata query API requests, see "Request format" on page 18.

Specifying the authentication header in Python with PycURL

In Python with PycURL, you use the **HTTPHEADER** option to specify a header, as in this example:

```
curl.setopt(pycurl.HTTPHEADER, ["Authorization: HCP
    bXl1c2Vy:6ecaf581f6879c9a14ca6b76ff2a6b15"])
```

3

Query requests

This chapter describes how to construct both object-based and operation-based query requests.

Request format

You use the HTTP **POST** method to send a metadata query API request to HCP. The **POST** request for both object-based and operation-based queries has these elements:

- A request URL.

  For information on request URL formats, see "Request URL" on page 10.

- Optionally, if the URL starts with an IP address, an HTTP Host header.

  For information on specifying the HTTP Host header, see "Connecting using an IP address" on page 11.

- An Authorization header.

  For information on the Authorization header, see "Authentication" on page 14.

- An HTTP Content-Type header with one of these values:

  – If the request body is XML, application/xml

  – If the request body is JSON, application/json

- An HTTP Accept header to specify the response format: application/xml or application/json.

- Optionally, to send the query in gzip-compressed format:

  – An HTTP Content-Encoding header with a value of gzip

  – A chunked transfer encoding

Note: When using cURL to send the query in gzip-compressed format, the request must specify **--data-binary**. If the request specifies **-d** instead, HCP returns a 400 (Bad Request) error.

- Optionally, to request that HCP return the response in gzip-compressed format, an HTTP Accept-Encoding header containing the value gzip or *. The header can specify additional compression algorithms, but HCP uses only gzip.

- Optionally, to request that HCP format the returned XML or JSON in an easily readable format, a **prettyprint** URL query parameter. The **prettyprint** parameter increases the time it takes to process a request. Therefore, you should use it only for testing purposes and not in production applications.

- A request body containing the query criteria and specifications for the contents of the result body. The entries you can specify depends on whether the request body is for an object-based query or an operation-based query.

  For more information on the request body for object-based queries, see "Object-based query requests" below. For more information on the request body for operation-based queries, see "Operation-based query requests" on page 44.

Object-based query requests

The body of an object-based query request consists of entries in XML or JSON format.

XML request body for object-based queries

The XML request body for an object-based query must contain a top-level **queryRequest** entry, an **object** entry, and a **query** entry. All other entries are optional.

The XML request body for an object-based query has the format shown below. The entries under **object** can be specified in any order.

```
<queryRequest>
  <object>
    <query>query-expression</query>
    <contentProperties>(true|false)</contentProperties>
    <count>number-of-results</count>
    <facets>comma-separated-list-of-facets</facets>
    <objectProperties>comma-separated-list-of-properties
    </objectProperties>
    <offset>number-of-results-to-skip</offset>
    <sort>object-property[+(asc|desc)][,.object-property
      [+(asc|desc)]]....</sort>
    <verbose>(true|false)</verbose>
  </object>
</queryRequest>
```

For a complete list of object properties, see "Object properties" on page 52.

JSON request body for object-based queries

The JSON request body for an object-based query must contain an unnamed top-level entry, an **object** entry, and a **query** entry. All other entries are optional.

The JSON request body for an object-based query has the format shown below. The entries under **object** can be specified in any order.

```
{
  "object":{
    "query":"query-expression",
    "contentProperties":"(true|false)",
    "count":number-of-results,
    "facets":"comma-separated-list-of-facets",
    "objectProperties":"comma-separated-list-of-properties",
    "offset":number-of-results-to-skip,
    "sort":"object-property[+(asc|desc)][,.object-property
      [+(asc|desc)]]..."
    "verbose":"(true|false)"
  }
}
```

For a complete list of objectProperty values, see “Object properties” on page 52.

Request body contents

The following sections describe the entries in an object-based metadata query API request body.

Top-level entry

XML has a single top-level **queryRequest** entry. JSON has a corresponding unnamed top-level entry. All request bodies must contain this entry.

object entry

The **object** entry is required for object-based requests. It must contain the **query** entry and can contain any combination of the other entries listed in the table below.

Entry	Valid values	Description
query	A query expression	Specifies the query criteria. This entry is required. For more information on query expressions, see "Query expressions" on page 26.
content Properties	One of: • **true** — Return information for all content properties. • **false** — Do not return any information on content properties.	Returns information about the content properties available for use in queries. The default is **false**. To return only content properties, specify a **count** entry with a value of 0.
count	One of: • **-1**, to request all results • **0** to request a response that includes only the object count and, if requested, content properties and facets. • An integer between one and 10,000	Specifies the maximum number of results to return. If you omit this entry, HCP returns a maximum of one hundred results. HCP responds significantly faster to a request for all results when the request is for basic information only (that is, the value of the **verbose** entry is (or defaults to) **false** and the **objectProperties** entry is omitted). Additionally, a request for all results that includes the **verbose** entry with a value of **true** or that includes the **objectProperties** entry may not return all the expected results due to a connection timeout.
facets	A comma-separated list of zero or more of: • hold • namespace • retention • retentionClass • *content-property-name*	Requests summary information for the returned values of the specified object properties. The values for this entry are case sensitive. For more information on this entry, see "facets entry" on page 23.

(Continued)

Entry	Valid values	Description
object Properties	A comma-separated list of object properties	Requests specific object properties to return for each **object** entry in the query results. All object entries include the **operation**, **version**, **urlName**, and **changeTimeMilliseconds** properties, so you don't need to specify them in this property. If you specify this property, any **verbose** property is ignored. For a list of object properties, see "Object properties" on page 52.
offset	An integer between zero and 100,000	Skips the specified number of object entries in the complete result set. Specify this entry when you're performing a paged query. The default is zero. For information on performing paged queries, see "Paged queries with object-based requests" on page 42.
sort	A comma-separated list of object properties and content properties with optional sort-order indicators	Specifies the sort order for object entries in the result set. For more information on this entry, see "sort entry" on page 22.
verbose	One of: • **true** — Return all object properties. • **false** — Return only the object URL, version ID, operation, and change time.	Specifies whether to return complete metadata for each object in the result set (**true**) or only the object URL, version ID, operation type, and change time. The default is **false**. If the request body contains both this property and the **objectProperties** property, this property is ignored. For information on the returned properties, see "Object properties" on page 52.

sort entry

You use the **sort** entry to specify the order in which object-based query results are listed. The entry contains a comma-separated list of properties and a sort-order indicator, in this format:

object-property[**+**(**asc**|**desc**)][**,***.object-property*[**+**(**asc**|**desc**)]]...

asc means sort in ascending order. **desc** means sort in descending order. The default is **asc**.

For information on the object properties on which you can sort query results, see "Object properties" on page 52.

Sort order

Objects are sorted by properties in the order in which the properties are listed in the **sort** entry. For example, to sort query results in ascending order based on namespace name and descending order based on size within each namespace, specify this entry:

```
<sort>namespace+asc,size+desc</sort>
```

If you omit the **sort** entry, the query results are listed in order of relevance to the query criteria.

Sorting on content properties

You can sort only on single-valued content properties. You cannot sort on properties that can have multiple values.

facets entry

You use the **facets** entry to request summary information for the returned values of specified object properties. For each specified property, HCP returns a list of up to one hundred object property values that occur most frequently in the result set. Each entry in the list has the number of objects that have each of the object property value. For example, if you specify **retentionClass** in the **facets** entry, HCP returns a list of up to one hundred retention classes that occur with objects in the result set, along with the number of objects in each of those classes.

Facet object properties

The value of the **facets** entry is a comma-separated list of one or more of the object properties in the table below. Multiple properties can be specified in any order.

Object property	Description
hold	Returns the numbers of objects in the result set that are on hold and not on hold.
namespace	Returns the names of namespaces that contain objects in the result set and the number of objects in the result set in each of those namespaces.

(Continued)

Object property	Description
retention	For each of these retention values, returns the number of objects in the result set that have that value: • **initialUnspecified** — For objects with a retention setting of Initial Unspecified • **neverDeletable** — For objects with a retention setting of Deletion Prohibited • **expired** — For objects with a retention setting that is Deletion Allowed or a specific date in the past • **not expired** — For objects with a retention setting that is a specific date in the future
retentionClass	Returns the retention classes that are retention settings for objects in the result set and the number of objects in each retention class. The count of objects in a retention class can include objects from more than one namespace. This is because multiple namespaces can have retention classes with the same name. To get an accurate count of the objects in a namespace that are in a specific retention class, restrict the query to a single namespace.
content-property-name	For Boolean and string content properties, returns the number of objects with the specified property value. For numeric and date properties, returns the number of objects in ranges of values. You cannot use tokenized (full-text searchable) content properties with facets. For information on specifying ranges for numeric and date content properties, see "Content property facet ranges" below.

Content property facet ranges

For numeric and date content properties, you specify the minimum and maximum values (range) for which to return information. You also specify the size of the sub-ranges (the interval) into which to divide the range.

You use the following format to specify the range and interval for facets for content properties with a type of integer, floating point, or date:

(*start-value***;***end-value***;+***interval***)**

In this expression:

- *start-value* is inclusive, that is the range includes the specified value.
- Each entry in the response has an interval that is as close as possible to the specified interval, but not larger than it.
- *end-value* is inclusive.
- The last facet must have a full interval, even if *end-value* is less than the end of an interval.

For example, if the a **facets** entry includes a **salary** content property with a *start-value* of 10,000, an *end-value* of 99,999.99 and an *interval* of 10,000, the response will include ten entries for the property. The first entry will contain the number of employees with salaries of 10,000.00 through 19,999.99, the second will count salaries of 20,000.00 through 29,999.99, and the last will count salaries of 90,000 through 99,999.99.

However, if you specify an *end-value* of 100,000, the response will include 11 entries. The tenth entry will count salaries of 90,000.00 through 99,999.99, as before, but the response will include an additional entry that counts salaries of 100,000 through 109,999.99, even though you specified only 100,000

For dates:

- The start-value and end value must be either **NOW**, for the time when HCP processes the request, or a date-time value in this format:

  yyyy-*mm*-*dd*T*hh*:*mm*:*ss*Z

  The time must be in UTC (coordinated universal time, also known as Greenwich Mean Time), not the local time, and you must specify the letter **Z** at the end of the format. For example, to specify noon Eastern Standard Time on February 10, 2013, specify 2013-02-10T17:00:00Z

- Follow these rules when you specify the interval:

  – Specify the time using a number immediately followed by the calendar unit: **SECOND**, **MINUTE**, **HOUR**, **DAY**, **MONTH**, **YEAR**. You can use plurals of these values, for example 2MONTHS.

  – Precede the time interval with a plus sign (+).

  – You can combine intervals, such as +1YEAR+6MONTHS

This example requests facet information for three content properties, **salary**, **dateOfBirth**, and **zip**:

```
<facets>salary(0;999999.99;50000),
    dateOfBirth(1900-01-01T00:00:00Z;*;+10YEARS),zip</facets>
```

The example consists of these facets:

- The **salary** facet requests The number of objects with salaries in the range zero through 999,999.00, broken out into intervals of 50,000.
- The **dateOfBirth** facet requests the number of objects with birth dates in each ten-year interval from midnight, January 1, 1900 to now.
- The **zip** facet requests the number of objects with each zip code that occurs in the result set. In this example, the zip content property has a type of string, so you cannot specify a range or interval for it. This request returns facets only for zip codes that have at least one matching object.

Query expressions

With object-based queries, you specify a query expression in the **query** request entry. Query expressions have this format:

```
[+|-]criterion [[+|-]criterion]...
```

In this expression, [**+**|**-**] is an optional Boolean operator and *criterion* is one of:

- A single text-based or property-based criterion.
- One or more criteria in parentheses, in this format:

  ([**+**|**-**]*criterion* [[**+**|**-**]*criterion*]*...***)**

  In this expression *criterion* can be a single criterion or one or more criteria in parentheses.

For example, here is one possible query expression:

```
-(namespace:"finance.europe")  +(retention:0 index:1)
```

Query expressions can contain only valid UTF-8 characters.

Tip: You can use the Metadata Query Engine Console to generate query expressions. To do this, construct a query on the **Structured Query** page and then click on the **Show as advanced query** link. The resulting advanced query can be used as a query expression in an object-based query request.

For information on constructing structured queries with the Metadata Query Engine Console, see *Searching Namespaces*.

For information on the criteria you can specify in query expressions, see "Text-based criteria" below and "Property-based criteria" on page 30. For information using criteria to construct query expressions, see "Query expression considerations" on page 34. For more information on the query language used to construct query expressions, see the Apache Solr documentation at http://lucene.apache.org/solr/documentation.

Text-based criteria

Text-based criteria let you perform queries based on object paths and the full-text content of custom metadata. Queries that use text-based criteria find objects with matching custom metadata only in namespaces that are configured to support full-text searches of custom metadata.

To perform queries based on object paths only or on custom metadata content only, use property-based criteria. For information on property-based criteria, see "Property-based criteria" on page 30.

A single text-based criterion is a text string consisting of one or more UTF-8 characters. This string is interpreted as one or more search terms, where each search term is a sequence of either alphabetic or numeric characters. All other characters, except wildcards, are treated as term separators.

For example, the string *product123* contains two search terms — *product* and *123*. A query based on this string finds objects with paths or custom metadata that contains at least one of *product* and *123*.

Search terms match only complete alphabetic or numeric strings in paths or custom metadata. For example, the text strings *AnnualReport*, *2012*, and *AnnualReport_2012* match the object named AnnualReport_2012.pdf. A query expression with a text string such as *Annual* or *201* does not match this object.

Similarly, to query for objects with a path or custom metadata that contains the word *product*, you need to use the complete word *product* as the text string. A query expression with a text string such as *prod* does not match objects with a path or custom metadata containing *product*.

Search terms are not case sensitive. Therefore, the text strings *AnnualReport*, *Annualreport*, and *annualreport* are equivalent.

Common words such as *a* and *is* are valid search terms. For example, a query containing the text string *A3534* matches all objects with paths and custom metadata that contain the word *a*. To prevent such a match, use a phrase as described below.

To specify a negative number as a text-based criterion, enclose the criterion term in double quotation marks ("); for example, "-3121".

To specify a phrase as a criterion, put the text string in double quotation marks. A phrase matches paths and custom metadata that contain each of the alphabetic or numeric search terms within the quotation marks in the specified order, but any special characters or white space between the individual strings is ignored. For example, the phrase "product 123" matches custom metadata that contains any of these strings:

```
product 123
product123
product_123
```

Boolean operators in text-based criteria

You can precede a text-based criterion with one of these Boolean operators:

- Plus sign (+) — Objects in the result set must contain the search term following the plus sign.

- Minus sign (-) — Objects in the result set must not contain the search term following the minus sign.

For example, this query expression finds objects where the path and custom metadata do not contain the string *product*.

```
-product
```

If a value is in quotation marks, the Boolean operator comes before the opening quotation mark. For example, this query expression finds objects with paths or custom metadata that contains the phrase *wetland permit*:

```
+"wetland permit"
```

A plus sign in front of a string that is not all-alphabetic or all-numeric finds paths and custom metadata that match at least one of the search terms. For example, the following expression matches paths and custom metadata that contain either the string *product* or the number *456*:

```
+product456
```

A minus sign in front of a string that is not all-alphabetic or all-numeric finds paths that contain none of the search terms. For example, the following expression matches all paths and custom metadata that do not contain the string *product* or the number *456*:

```
-product456
```

Wildcard characters in text-based criteria

You can use these wildcard characters in or at the end of the text string for a text-based criterion:

- Question mark (?) — Represents a single character
- Asterisk (*) — Represents any number of consecutive printable characters, including none

These characters do not function as wildcards when included within double quotation marks (").

Wildcards are not valid at the beginning of a text string. For example, the query expression on the left is valid; the query expression on the right is not:

Valid: princ* **Invalid:** *cipal

You can use multiple wildcards in a criterion. Two asterisks next to each other are treated as a single asterisk. Asterisks with characters between them are treated as separate wildcards. For example, the criterion below matches the path /Conflicts.txt:

```
c**nflict*
```

Similarly, in an **all** query, the criterion below matches any path with at least two directories preceding the object in the path:

```
/*/*/**
```

Two question marks next to each other are treated as separate wild cards. For example, the criterion below does not match the path /Conflicts.txt:

```
c??nflict*
```

Wildcards between text that the metadata query engine considers to be separate search terms are not valid. For example, the search string below does not match the path test1.txt because the wildcard is between an alphabetic character and a numeric character:

```
tes*1
```

Property-based criteria

Property-based criteria let you query for objects based on specified object property values. The format for a simple property-based criterion is:

property:*value*

For example, this expression finds objects that are on hold:

```
hold:true
```

When querying for a value that's a negative number, enclose the value in double quotation marks ("). For example, this query expression finds objects with the retention setting -2:

```
retention:"-2"
```

The special property based criterion *:* matches all objects in all namespaces searchable by the user.

For information on:

- The object properties and values you can use in property-based criteria, see "Object properties" on page 52.
- Specifying the property value when you query for an object path, custom metadata content, or a content property with the tokenized data type, see "Text-based criteria" on page 27.
- The property you use to query for objects based on the content of ACLs, see "aclGrant property" on page 38.

- The property you use to query for objects based on the full-text content of custom metadata, see "customMetadataContent property" on page 35.

Boolean operators with property-based criteria

You can precede a criterion or an individual property value with one of these Boolean operators:

- Plus sign (+) — Objects in the result set must contain the criterion or value following the plus sign.

- Minus sign (-) — Objects in the result set must not contain the criterion or value following the minus sign.

For example, this query expression finds objects that are not on hold:

```
-hold:true
```

Multiple values for a single property

A property-based criterion can specify multiple values for a single property. To specify multiple values, use this format:

```
property:([+|-]value [[+|-]value]...)
```

In this format, the parentheses are required.

For example, this query expression finds objects in either the HlthReg-107 or HlthReg-224 retention class:

```
retentionClass:(HlthReg-107  HlthReg-224)
```

This query expression finds objects with custom metadata that contains the string *finance* but not the string *foreign*.

```
customMetadataContent:(+finance -foreign)
```

When you specify multiple values for a single property, you can combine values that are preceded by Boolean operators with values that do not have Boolean operators. In this case, objects that match the property values that are not preceded by Boolean operators may or may not appear in the result set, but objects that match the terms without Boolean operators are sorted higher in the query results than objects that don't match those terms.

For example, this query expression finds objects that have custom metadata that contains both the terms *quarterly report* and *accounting department* or only the term *quarterly report*:

```
customMetadataContent:(+"quarterly report" "accounting department")
```

Objects that contain both terms are sorted higher in the query results.

Value ranges

You can query based on ranges of values for properties with numeric, string, or date data types. These properties are **accessTime**, **accessTimeString**, **changeTimeString**, **dpl**, **hash**, **hashScheme**, **ingestTime**, **ingestTimeString**, **retention**, **retentionClass**, **retentionString**, **size**, **updateTime**, **updateTimeString**, and **utf8Name**. You can also query based on ranges for content properties with numeric, string or date data types.

Criteria that query for a range of values can have either of these formats:

- For a range that includes the start and end values:

  property`:[`*start-value* **TO** *end-value*`]`

  In this format, the square brackets are required.

  For example, this query expression finds objects that were ingested from 0800 through 0900 UTC on March 1, 2012, inclusive:

  ```
  ingestTimeString:[2012-03-01T08:00:00-0000 TO 2012-03-01T09:00:00-0000]
  ```

- For a range that does not include the start or end values:

  property`:{`*start-value* **TO** *end-value*`}`

  In this format, the curly braces are required.

  For example, this query expression finds objects that have names that occur alphabetically between *Brown_Lee.xls* and *Green_Chris.xls*, exclusive of those values:

  ```
  utf8Name:{Brown_Lee.xls TO Green_Chris.xls}
  ```

Note: utf8Name property values are case sensitive and are ordered according to the positions of characters in the UTF-8 character table.

You can mix square brackets and curly braces in an expression. For example, this query expression finds objects that were ingested from 0800 to 0900 UTC on March 1, 2012, including objects that were ingested at 0800 but excluding objects that were ingested at 0900:

```
ingestTimeString:[2012-03-01T08:00:00-0000 TO 2012-03-01T09:00:00-0000}
```

When querying for a range of property values, you can precede the whole criterion with a Boolean operator but you cannot precede an individual value with a Boolean operator. For example, the query expression on the first line is valid; the criterion on the second line is not:

Valid: +retentionString:[2013-07-01T00:00:00 TO 2013-07-31T00:00:00]

Invalid: retentionString:[+2013-07-01T00:00:00 TO 2013-07-31T00:00:00]

When querying for a range of values, you can replace a value with an asterisk (*) to specify an unlimited range. For example, this query expression finds objects with a size equal to or greater than two thousand bytes:

```
size:[2000 TO *]
```

This query expression finds objects with change times before 9:00 AM, March 1, 2012 in the local time zone of the HCP system:

```
changeTimeString:[*  TO  2012-03-01T09:00:00}
```

Wildcard characters in property-based searches

You can use the question mark (?) and asterisk (*) wildcard characters when specifying values for these object properties:

- **customMetadataContent**
- **hash**
- **hashScheme**
- **retentionClass**
- **objectPath**
- **utf8Name**
- content properties

For example, this query expression finds objects assigned to any retention class starting with HlthReg, such as HlthReg-107 or HlthReg-224:

```
retentionClass:HlthReg*
```

The question mark and asterisk characters do not function as wildcards when included within double quotation marks (").

Wildcards are not valid at the beginning of a property value. For example, the query expression on the left is valid; the query expression on the right is not:

Valid: utf8Name:princ* **Invalid:** utf8Name:*cipal

For information on using wildcards with **objectPath** and **customMetadataContent** properties and for content properties with the Tokenized data type, see "Wildcard characters in text-based criteria" on page 29.

Query expression considerations

These considerations apply to query expressions, whether they contain property-based criteria, text-based criteria, or a combination of both:

- If the query expression consists of a single criterion without a Boolean operator, objects in the result set must meet the criterion. For example, this query expression finds objects with custom metadata that contains the string *accounting*:

  ```
  customMetadataContent:accounting
  ```

  The expression above is equivalent to this expression that uses the plus sign (+):

  ```
  +customMetadataContent:accounting
  ```

- If a query expression consists of multiple criteria without Boolean operators, objects in the result set must meet at least one of the criteria. For example, this query expression finds objects that have a retention setting of Deletion Allowed or are on hold or will be shredded on deletion:

  ```
  retention:0 hold:true shred:true
  ```

- The greater the number of criteria an object meets, the higher the object is in the default sort order. For example, with this query expression, objects that match all three criteria are sorted higher than those that match only two, and those that match only two are sorted higher than those that match only one:

  ```
  retention:0 hold:true shred:true
  ```

- If a plus sign precedes some search criteria but not others, the criteria that are not preceded by a plus sign have no effect on which objects are returned. For example, this query expression finds objects that

have a utf8Name property with the value *Q1_2012.ppt*, regardless of whether they are in the finance namespace owned by the europe tenant:

```
+utf8Name:"Q1_2012.ppt"  namespace:"finance.europe"
```

Objects that match the namespace criterion are sorted higher in the result set than those that do not match it.

- If a minus sign precedes some search criteria but not others and no criteria have plus signs, the query expression finds objects that do not match the criteria preceded by the minus signs and do match at least one of the criteria without a Boolean operator. For example, this query expression finds objects that are not in the finance namespace owned by the europe tenant and can be deleted.

  ```
  -namespace:"finance.europe"  retention:0
  ```

  This query finds objects that are not in the finance namespace owned by the tenant europe and either can be deleted or can be indexed (or both):

  ```
  -namespace:"finance.europe"  retention:0  index:1
  ```

- If a Boolean operator precedes an opening parenthesis, that operator applies to the entire set of criteria inside the parentheses, not the individual criteria. For example, this query expression finds objects that are on hold or have a retention setting of Deletion Prohibited:

  ```
  +(hold:true  retention:"-1")
  ```

- These characters have special meaning when specified in query expressions:

  ```
  ? * + - ( ) [ ] { } " :
  ```

  To specify one of these characters in a query expression, precede the character with a backslash (\). To specify a backslash in a query expression, precede the backslash with another backslash.

customMetadataContent property

To search for objects based on the full-text content of custom metadata, you specify the **customMetadataContent** property in a query expression. Criteria that use this property find objects only in namespaces that have full-text indexing of custom metadata enabled.

When custom metadata is indexed for full-text searching, the XML is treated as text, not as a structured document. Similarly, the **customMetadataContent** property value is treated as text. Therefore, the rules described in "Text-based criteria" on page 27 apply to the property value.

Tip: If you frequently search for values of a particular element or attribute, use a content property that corresponds to that element or attribute, as content property searches are more efficient than **customMetadataContent** searches. If the required content property does not exist, ask your tenant administrator to create one.

To use the **customMetadataContent** property to query for any element name, attribute name, element value, or attribute value that matches a text string, use a query expression with this format:

customMetadataContent:*text-string*

If the text string consists of more than a single string of alphabetic or numeric characters, enclose the entire value in double quotation marks (").

To query for a combination of elements and attribute names and values, use a query expression with either of these formats:

customMetadataContent:"*element-name.*
*attribute-name.attribute-value...element-value.element-name***"**

<![CDATA[customMetadataContent:"<*element-name*
*attribute-name=attribute-value...>element-value</element-name>***"]]>**

The two formats are equivalent. The first format is simpler. The second format uses well-formed XML.

When using the second format, enclose both the property and text string in the square brackets that mark the CDATA content, and enclose the text string in double quotation marks ("). The outer square brackets ([ ]) are also required, as are the outside angle brackets and exclamation mark.

To query for the value of a specific element, specify every attribute and attribute value for the element, not just the element name and value.

To query for the value of a specific attribute, regardless of which element it applies to, use this format:

customMetadataContent:"*attribute-name.attribute-value***"**

You can use the asterisk (*) and question mark (?) wildcard characters when specifying **customMetadataContent** property values that are not in quotation marks. For information on specifying these wildcards, see "Text-based criteria" on page 27.

Here is some sample custom metadata that you might want to search:

```
<?xml version="1.0" ?>
<weather>
    <location>Boston</location>
    <date>20121130</date>
    <duration unit="secs">180</duration>
    <temp>
        <temp_high unit="deg_F">31</temp_high>
        <temp_low unit="deg_F">31</temp_low>
    </temp>
    <velocity>
        <velocity_high unit="mph">17</velocity_high>
        <velocity_low unit="mph">14</velocity_low>
    </velocity>
    <conditions>partly cloudy</conditions>
</weather>
```

Here are some examples of query expressions that use the **customMetadataContent** property to search the XML:

- This query expression finds objects that have custom metadata with an element name, element value, attribute name, or attribute value that contains *Boston*:

  ```
  customMetadataContent:Boston
  ```

- This query expression finds objects that have custom metadata that contains the **location** element with a value of *Boston*:

  ```
  customMetadataContent:"location.Boston.location"
  ```

- This query expression finds objects that have custom metadata that contains the **velocity_high** element with a value of *17* and the **unit** attribute with a value of *mph*:

  ```
  customMetadataContent:"velocity_high.unit.mph.17.velocity_high"
  ```

- This query expression returns objects that have custom metadata that contains the **conditions** element with a value of *partly cloudy*:

  ```
  customMetadataContent:"conditions.partly cloudy.conditions"
  ```

- This query expression finds objects that have custom metadata that contains the **date** element with a value of *20121130*:

  ```
  <![CDATA[customMetadataContent:"<date>20121130</date>"]]>
  ```

- This query expression finds objects that have custom metadata that contains the **temp_high** element with a value of *31* and the **unit** attribute with a value of *deg_F*:

  ```
  <![CDATA[customMetadataContent:"<temp_high unit=deg_F>31
      </temp_high>"]]>"
  ```

aclGrant property

To query for objects based on the content of ACLs, you specify the **aclGrant** property in a query expression. Valid values for this property have these formats:

"*permissions*"

"*permissions*,**USER**[,*location*,*username*]"

"permissions,**GROUP**,*location*,(*ad-group-name*|**all_users**|**authenticated**)"

In these formats:

- *permissions* is one or more of these with no space between them:
  - **R** — Read_ACL
  - **r** — Read
  - **W** — Write_ACL
  - **w** — Write
  - **d** — Delete

  If you specify only *permissions* as the **aclGrant** property value, the query expression finds objects with ACLs that grant you the specified permissions to any user or group. For information on specifying permissions, see "Specifying permissions" below.

- **USER** is required when querying for objects with ACLs that grant permissions to a specified user.

If the credentials you specify in the query request are for a tenant-level user account that's defined in HCP, you can find objects that have ACLs that grant the specified permissions to that user account by specifying only a value for *permissions* and **USER**.

- **GROUP** is required when querying for objects with ACLs that grant permissions to a specific group of users.

- *location* is the location in which the specified user or group is defined. Valid values are either:

  – The name of an HCP tenant

  – The name of an AD domain preceded by an at sign (@)

  If the value for the **aclGrant** property includes **all_users** or **authenticated**, *location* must be the name of an HCP tenant.

- *username* is the name of a user to which matching ACLs grant the specified permissions. Valid values are:

  – The username for a user account that's defined in HCP.

  – The username for an AD user account. This can be either the user principal name or the Security Accounts Manager (SAM) account name for the AD user account.

- *ad-group-name* is the name of an AD group to which the matching ACLs grant the specified permissions.

- **all_users** represents all users.

- **authenticated** represents all authenticated users.

Specifying permissions

The permissions in an **aclGrant** property value must be specified in this order:

R, r, W, w, d

For example, to find objects that have ACLs that grant write and write_ACL permissions, and only those permissions, to the user rsilver who is defined in the europe tenant, specify this query expression:

```
aclGrant:"Ww,USER,europe,rsilver"
```

You can replace one or more permissions with the asterisk (*) wildcard character. When you do this, you still need to specify permissions in the correct order.

When you specify both an asterisk and one or more permissions, the metadata query API finds objects with ACLs that grant only the permissions you explicitly specify or that grant the permissions you explicitly specify and any permissions represented by the asterisk. For example, this query expression finds objects with ACLs that grant read, read_ACL, write, and write_ACL permissions and may also grant delete permission:

```
aclGrant:"RrWw*"
```

A single asterisk represents all the missing permissions in the location where it appears. Therefore, you don't use consecutive asterisks. For example, in this query expression, the wildcard character represents any combination of write, write_ACL, and delete permissions:

```
aclGrant:"r*"
```

In this query expression, the wildcard character represents any combination of read and write_ACL permissions:

```
aclGrant:"R*w"
```

In this query expression, the wildcard character represents only read_ACL permission:

```
aclGrant:"*r"
```

You can specify multiple asterisks in a query expression. For example, this query expression finds objects with ACLs that grant read permission and any combination of other permissions to the AD group named managers that is defined in the corp.widgetco.com domain:

```
aclGrant:"*r*,GROUP,@corp.widgetco.com,managers"
```

By replacing all permission values with a single asterisk, you query for objects that have ACLs that grant any combination of permissions. For example, if you're accessing the metadata query API with a tenant-level user account, this query expression finds objects with ACLs that grant any combination of permissions to that user account:

```
aclGrant:"*,USER"
```

Note: Using **aclGrant** without specifying a user and tenant returns every object in the index that has the ACL are searching. For instance, aclGrant:"r" returns all objects that have the Read ACL set.

aclGrant considerations

These considerations apply when you specify the **aclGrant** property in a query expression:

- The entire value for this property must be enclosed in double quotation marks (" ").
- The locations and usernames you specify are not case sensitive.
- The group names you specify, except for **all_users** and **authenticated**, are case sensitive.
- The permission values you specify and the values **USER** and **GROUP** are case sensitive.

Query expression examples

Here are some examples of query expressions that use both text-based criteria and property-based criteria:

- This expression returns metadata for objects that have a retention setting of Deletion Allowed, are not on hold, and may or may not have a path or custom metadata that contains the term *report*:

  ```
  +(retention:0) -(hold:true) report
  ```

- This expression returns metadata for objects in the finance namespace under the `/Corporate/Employees` directory that were ingested after March 1, 2012:

  ```
  +(namespace:"finance.europe"  objectPath:"/Corporate/Employees"
  ingestTimeString:[2012-03-01T00:00:00 TO *])
  ```

Paged queries with object-based requests

To use a paged query with object-based requests:

- In the first request, use a **count** entry with a value of zero to get a response that does not include any object records but contains a **totalResults** value that specifies the total number of objects that meet the query criteria.
- In each request after the first, optionally specify a **count** entry. If you omit the **count** entry, the result set includes at most 100 objects.
- After each request, check the value of the **code** property of the **status** entry to determine whether the result set contains the last object that meets the criteria:
  - If the value is **INCOMPLETE**, more results remain. Request another page.
  - If the value is **COMPLETE**, the result set includes the last object that meets the query criteria.

For information on handling paged queries with 100,000 or fewer matching objects, see "Paged queries with 100,000 or fewer matching objects" below. For information on handling requests with very large numbers of matching objects, see "Paged queries with more than 100,000 matching objects" on page 42.

For an example of using a paged object-based query, see "Example 2: Using a paged query to retrieve a list of all objects in a namespace" on page 80.

Paged queries with 100,000 or fewer matching objects

If no more than 100,000 objects match the query criteria, use the **offset** entry to page through the result set. In each request after the first one with a **count** value greater than 0, include an **offset** entry that specifies the number of results to skip when returning the next page of results. For example, if you specified a **count** value of 50 for your first request, specify an **offset** value of 50 for your second request.

Paged queries with more than 100,000 matching objects

If a large number of objects match the query criteria, a paged query can consume a large amount of memory. If more than 100,000 objects match the query criteria, limit memory use by using multiple paged queries. Each

paged query should retrieve results for no more than 100,000 objects. To do this, use the **changeTimeMilliseconds** as the basis for generating the paged queries, as follows:

1. Issue a request with a **count** entry value of zero and a **changeTimeMilliseconds** criterion with a range from zero to some time in the past, such as this:

```
<queryRequest>
    <object>
        <query>+changeTimeMilliseconds:[0 TO 1262304000000.00]
            +retentionClass:hlthReg-107</query>
        <count>0</count>
    </object>
</queryRequest>
```

   If the **count** property in the response is greater than 100,000, repeat this step with an earlier **changeTimeMilliseconds** end time until the **count** property in the response is no more than 100,000.

2. Use a paged query with:

   – A **changeTimeMilliseconds** criterion that specifies the same time as you used in the last request in step 1

   – A **count** entry value that specifies the number of objects you want per page

   – An **offset** entry that you increment by the **count** value in each request

   For example, the request body for the third iteration of the paged query might look like this:

```
<queryRequest>
    <object>
        <query>+changeTimeMilliseconds:[0 TO 1150000000000.00]
            +retentionClass:hlthReg-107</query>
        <sort>changeTimeMilliseconds</sort>
        <count>50</count>
        <offset>150</offset>
    </object>
</queryRequest>
```

   Stop when the **code** property of the **status** entry in the response is **COMPLETE**.

3. Repeat step 1 above using a **changeTimeMilliseconds** entry that specifies a range with start value equal to the end value of the **changeTimeMilliseconds** range you used in step 2. Use a curly opening brace for the range so that the last entry in the previous result set is not included in the new results. For example, use a changeTimeMilliseconds value like this:

   changeTimeMilliseconds:{1150000000000.00 TO 1341000000000.00]

   Then repeat step 2 using the new query criteria.

4. Repeat step 3 until you retrieve the last matching object. Use a value of * (for an unlimited range) as the end of the **changeTimeMilliseconds** range in the last paged query to ensure that you retrieve all objects including those that were most recently added.

Operation-based query requests

The body of an operation-based query request consists of entries in XML or JSON format.

XML request body for operation-based queries

The XML request body for an operation-based query must contain a top-level **queryRequest** entry and, except when requesting all available information, an **operation** entry. All other entries are optional.

The XML request body has the format shown below. Entries at each hierarchical level can be specified in any order:

```
<queryRequest>
  <operation>
    <count>number-of-results</count>
    <lastResult>
      <urlName>object-url</urlName>
      <changeTimeMilliseconds>change-time-in-milliseconds.index
      </changeTimeMilliseconds>
      <version>version-id</version>
    </lastResult>
    <objectProperties>comma-separated-list-of-properties
    </objectProperties>
    <systemMetadata>
      <changeTime>
        <start>start-time-in-milliseconds</start>
        <end>end-time-in-milliseconds</end>
      </changeTime>
      <directories>
        <directory>directory-path</directory>
```

```
      ...
    </directories>
    <indexable>(true|false)</indexable>
    <namespaces>
      <namespace>namespace-name.tenant-name</namespace>
      ...
    </namespaces>
    <replicationCollision>(true|false)</replicationCollision>
    <transactions>
      <transaction>operation-type</transaction>
      ...
    </transactions>
  </systemMetadata>
  <verbose>(true|false)</verbose>
 </operation>
</queryRequest>
```

The XML body for an operation-based query that requests all available operation records contains only this line:

```
<queryRequest/>
```

JSON request body for operation-based queries

The JSON request body for an operation-based query must contain an unnamed top-level entry and, except when requesting all available information, the **operation** entry. All other entries are optional.

The JSON request body has the format shown below. Entries at each hierarchical level can be in any order:

```
{
  "operation": {
    "count":"number-of-results",
    "lastResult": {
      "urlName":"object-url",
      "changeTimeMilliseconds":"change-time-in-milliseconds.index",
      "version":version-id
    },
    "objectProperties":"comma-separated-list-of-properties",
    "systemMetadata": {
      "changeTime": {
        "start":start-time-in-milliseconds,
        "end":end-time-in-milliseconds
      },
      "directories": {
        "directory":["directory-path",...]
      },
      "indexable":"(true|false)",
      "namespaces": {
        "namespace":["namespace-name.tenant-name",...]
```

```
      },
      "replicationCollision":"(true|false)",
      "transactions": {
        "transaction":["operation-type",...]
      }
    },
    "verbose":"(true|false)"
  }
}
```

For the **namespace**, **directory**, and **transaction** entries, the square brackets shown in this format are required.

The JSON body for an operation-based query that requests all available operation records contains only this line:

```
{}
```

Request body contents

The following sections describe the entries in an operation-based metadata query API request body.

Top-level entry

An XML request body has a single top-level **queryRequest** entry. A JSON request body has a corresponding unnamed top-level entry. All requests must contain this entry.

operation entry

Except when requesting all available information, the **operation** entry is required for operation-based queries. It can contain any combination of the entries listed in the table below.

Entry	Valid values	Description
count	One of: • **-1**, to request all operation records that meet the query criteria • A positive integer	Specifies the maximum number of operation records to return per request. If you omit this entry, HCP returns up to ten thousand operation records per request.

(Continued)

Entry	Valid values	Description
lastResult	N/A	Specifies the last record returned by the previous query. Use this entry in paged queries to request additional results after an incomplete response. Omit this entry if you are not using a paged query or if this is the first request in a paged query. For descriptions of the child entries, see "lastResult entry" on page 48. For more information on paged queries, see "Paged queries with operation-based requests" on page 51.
objectProperties	A comma separated list of object properties.	Requests specific object property values for each object entry in the query results. If the request body contains both the **verbose** and **objectProperties** entries, HCP returns only the object URL, version ID, operation type, and change time and the information specified in the **objectProperties** entry. For a list of object properties, see "Object properties" on page 52.
systemMetadata	N/A	Specifies the properties to use as the query criteria. For descriptions of the child entries, see "systemMetadata entry" on page 48.
verbose	One of: • **true** — Return all object properties. • **false** — Return only the object URL, version ID, operation, and change time.	Specifies whether to return complete metadata for each operation record in the result set (**true**) or to return only the object URL, version ID, operation type, and change time (**false**). The default is **false**. If the query request body contains both the **verbose** and **objectProperties** entries, HCP returns only the object URL, version ID, operation type, and change time and the information specified in the **objectProperties** entry. For information on the returned properties, see "Object properties" on page 52.

lastResult entry

Use the **lastResult** entry only in the second through final requests of a paged query. This entry identifies the last record that was returned in the previous query so that HCP can retrieve the next set of records. The entry contains the child entries described in the table below.

Entry	Valid values	Description
urlName	A fully qualified object URL, for example: http://finance.europe.hcp.example.com/rest/Presentations/Q1_2012.ppt	Specifies the **urlName** value in the last operation record returned in response to the previous query.
changeTime Milliseconds	A timestamp in milliseconds since January 1, 1970, at 00:00:00 UTC, followed by a period and a two-digit suffix	Specifies the **changeTimeMilliseconds** value in the last operation record returned in response to the previous query. For more information on this entry, see"Object properties" on page 52.
version	A version ID	Specifies the **version** value in the last operation record returned in response to the previous query.

systemMetadata entry

The **systemMetadata** entry specifies the criteria that the returned operation records must match. The entry contains the child entries listed in the table below. Some of the subentries, such as **changeTime**, have children. In this table, the parent entries are immediately followed by their children.

Entry	Valid values	Description
changeTime	N/A	Specifies the range of change times of the objects for which to return operation records. This entry can contain neither, one, or both of the **start** and **end** child entries. If you omit this entry, HCP returns operation records for objects with change times between January 1, 1970, at 00:00:00 UTC and one minute before the time HCP received the request.

(Continued)

Entry	Valid values	Description
start *(child)*	One of: • Milliseconds since January 1, 1970, 00:00:00 UTC. • An ISO 8601 datetime value in this format: *yyyy*-*MM*-*dd*T*hh*:*mm*:*ssZ* *Z* represents the offset from UTC, in this format: (+|-)*hhmm* For example, 2011-11-16T14:27:20-0500 represents the start of the 20th second into 2:27 PM, November 16, 2011, EST.	Requests operation records for objects with change times on or after the specified date and time. This entry is a child entry of the **changeTime** entry. The default is zero (January 1, 1970, 00:00:00 UTC). In the ISO 8601 format, you cannot specify a millisecond value. The time corresponds to zero milliseconds into the specified second.
end *(child)*	One of: • Milliseconds since January 1, 1970, 00:00:00 UTC • An ISO 8601 datetime value in this format: *yyyy*-*MM*-*dd*T*hh*:*mm*:*ssZ*	Requests operation records for objects with change times before the specified date and time. This entry is a child entry of the **changeTime** entry. The default value is one minute before the time HCP received the request. In the ISO 8601 format, you cannot specify a millisecond value. The time corresponds to zero milliseconds into the specified second. If you specify a value that is less than one minute before the current time, ensure that all writes finished at least one minute ago so that you get results for the most recent operations.
directories	N/A	Specifies the directories to query. This entry contains zero or more **directory** entries. If you omit this entry, HCP returns operation records for objects in all directories in the specified namespaces.

(Continued)

Entry	Valid values	Description
directory *(child)*	The path to the directory containing the objects for which to retrieve operation records. Start the path with a forward slash (/) followed by the name of a directory immediately below `rest`, `data`, or `fcfs_data`. Do not include `rest`, `data`, or `fcfs_data` in the path.	Specifies a directory to query. This entry is a child of the **directories** entry. If you query multiple namespaces, HCP returns operation records for the directory contents in each namespace in which the directory occurs.
indexable	One of: • **true** — Return operation records only for objects with an index setting of **true**. • **false** — Return operation records only for objects with index setting of **false**.	Specifies whether to filter the returned operation records based on the object index setting. HCP returns deletion and purge records only for objects that had the specified setting at the time they were deleted or purged. If you omit this entry, HCP returns operation records for objects regardless of their index settings.
namespaces	N/A	Specifies the namespaces to query. This entry contains zero or more **namespace** entries. If the URL in the request starts with **default**, you can omit this entry. The URL itself limits the query to the default namespace. If you omit this entry and the URL starts with **admin**, HCP returns operation records for the default namespace and the namespaces owned by each tenant that has granted system-level users administrative access to itself. If you omit this entry and the URL starts with a tenant name, HCP returns operation records for the namespaces owned by the tenant that the user has rights to search.
namespace *(child)*	A namespace name along with the name of the owning tenant, in this format: *namespace-name*`.`*tenant-name*	Specifies a namespace to query. This entry is a child of the **namespaces** entry. For information on considerations that apply when you specify this entry, see "Querying specified namespaces" on page 105.

(Continued)

Entry	Valid values	Description
replication Collsion	One of: • **true** — Return operation records only for objects that are flagged as replication collisions. • **false** — Return operation records only for objects that are not flagged as replication collisions.	Specifies whether to filter the returned operation records based on whether the object is flagged as a replication collision. HCP returns deletion and purge records only for objects that were flagged as replication collisions at the time they were deleted or purged. If you omit this entry, HCP returns operation records for objects regardless of whether they are flagged as replication collisions.
transactions	N/A	Specifies the operation types for which to query. This entry contains up to five **transaction** entries, each specifying a different operation type. If you omit this entry, HCP returns records only for create, delete, and purge operations.
transaction *(child)*	One of: • **create** • **delete** • **dispose** • **prune** • **purge**	Specifies a type of operation for which to return records. This entry is a child entry of the **transactions** entry. HCP returns prune and disposition records only when you explicitly request them. Objects in the default namespace don't have prune or purge operation types. For more information on operation types, see "Operation-based query results" on page 4.

Paged queries with operation-based requests

To use a paged query with operation-based query requests:

- Optionally, specify a **count** entry in each request body. If you omit this entry, HCP returns ten thousand operation records per request.
- For each request after the first, specify a **lastResult** entry containing the values of the **urlName**, **changeTimeMilliseconds**, and **version** properties in the last record returned in response to the previous request.

- After each request, check the value of the **code** property of the **status** entry to determine whether the result set contains the last object that meets the criteria:
  - If the value is **INCOMPLETE**, more results remain. Request another page.
  - If the value is **COMPLETE**, the result set includes the last object that meets the query criteria.

For an example of using a paged operation-based query, see "Example 3: Using a paged query to retrieve a large number of records" on page 98.

Object properties

The table below describes the object properties that you can specify in these contexts:

- **objectProperties** entry
- **sort** entry
- Query entry

In the **sort** and **objectProperties** entries, you specify only the object property name. In query expressions, you specify both the property name and one or more values for the property.

The properties listed below are also returned in response bodies. The **verbose** and **objectProperties** request entries determine which properties are returned.

Object property	Data type	Description	Query expression example
accessTime	Long	The value of the POSIX **atime** attribute for the object, in seconds since January 1, 1970 at 00:00:00 UTC.	accessTime: [1312156800 TO 1312243200]

(Continued)

Object property	Data type	Description	Query expression example
accessTime String[1]	Datetime	The value of the POSIX **atime** attribute for the object, in ISO 8601 format: *YYYY-MM-DD*T*hh:mm:ssZ* *Z* represents the offset from UTC, in this format: (+|-)*hhmm* The UTC offset is optional. If you omit it, the time is in the zone of the HCP system. For example, 2011-11-16T14:27:20-0500 represents the 20th second into 2:27 PM, November 16, 2011, EST.	accessTimeString: [2012-03-01 T00:00:00 TO 2012-03-01 T23:59:59]
acl[2]	Boolean	An indication of whether the object has an ACL. Valid values are: • **true** — The object has an ACL. • **false** — The object does not have an ACL. This value is always **false** for objects in the default namespace.	acl:true
aclGrant	String	ACL content. This property can be used only in queries. It cannot be used in **sort** or **objectProperties** properties. For more information on this property, see "aclGrant property" on page 38.	aclGrant:"Ww,USER, europe,rsilver"
changeTime Milliseconds	String	The time at which the object last changed. For delete, dispose, prune, and purge records, this is the time when the operation was performed on the object. The value is the time in milliseconds since January 1, 1970, at 00:00:00 UTC, followed by a period and a two-digit suffix. The suffix ensures that the change time values for versions of an object are unique. This property is not returned for objects with the NOT_FOUND operation type. For more information on this operation type, see the description of the operation entry. This property corresponds to the POSIX **ctime** attribute for the object.	changeTimeMilliseconds: [1311206400000.00 TO 1311292800000.00]
changeTime String[1]	Datetime	The object change time in ISO 8601 format: *YYYY-MM-DD*T*hh:mm:ssZ* For more information on this format, see the description of the **accessTimeString** property. This property corresponds to the POSIX **ctime** attribute for the object.	changeTimeString: [2012-03-21 T00:00:00 TO 2012-03-21 T23:59:59]

(Continued)

Object property	Data type	Description	Query expression example
customMetadata[2]	Boolean	An indication of whether the object has custom metadata. Valid values are: • **true** — The object has custom metadata. • **false** — The object does not have custom metadata.	customMetadata:true
customMetadataAnnotation	String	One or more comma-delimited annotation names. Annotation names are case-sensitive.	customMetadataAnnotation:inventory
customMetadataContent	String	Custom metadata content. This property can be used only in queries. It cannot be used in **sort** or **objectProperties** properties. For more information on this property, see "customMetadataContent property" on page 35.	customMetadataContent:city.Bath.city
dpl	Integer	The DPL for the namespace that contains the object.	dpl:2
gid[3]	Integer	The POSIX group ID.	N/A
hash[4]	String	The cryptographic hash algorithm used to compute the hash value of the object, followed by a space and the hash value of the object. In query expressions, the values you specify for both the hash algorithm and the hash value are case sensitive. You need to use uppercase letters when specifying these values. When using wildcard characters with this object property, instead of a space, separate the hash algorithm and the hash value with a wildcard character. In this case, do not enclose the value for this property in quotation marks. If you do not specify wildcard characters in the value for this property, you need to enclose the entire value for this property in double quotation marks.	hash:"SHA-256 9B6D4..."
hashScheme[4]	String	The cryptographic hash algorithm the namespace uses. In query expressions, the values you specify for this property are case sensitive. Do not enclose these values in quotation marks.	hashScheme:SHA-256
hold[2]	Boolean	An indication of whether the object is currently on hold. Valid values are: • **true** — The object is on hold. • **false** — The object is not on hold.	hold:false

(Continued)

Object property	Data type	Description	Query expression example
index[2]	Boolean	An indication of which parts of the object are indexed. Valid values are: • **true** — All metadata, including any custom metadata and ACL, is indexed. • **false** — Only system metadata and ACLs are indexed.	index:true
ingestTime	Long	The time at which HCP stored the object, in seconds since January 1, 1970, at 00:00:00 UTC.	ingestTime:[130947840 TO 1312156800]
ingestTime String[1]	Datetime	The time at which HCP stored the object, in ISO 8601 format: *YYYY-MM-DDThh:mm:ssZ* For more information on this format, see the description of the **accessTimeString** property.	ingestTimeString: [2012-03-01 T00:00:00 TO 2012-03-01 T23:59:59]
namespace[2]	String	The name of the namespace that contains the object, in this format: *namespace-name.tenant-name* In query expressions, the values you specify for this property are not case sensitive. For considerations that apply when you specify this property in a query expression, see "Querying specified namespaces" on page 105.	namespace: finance.europe
objectPath[4]	String	The path to the object following `rest`, `data`, or `fcfs_data`, beginning with a forward slash (/). In query expressions, the values you specify for this property are not case sensitive and do not need to begin with a forward slash (/).	objectPath:"/Corporate/Employees/45_Jane_Doe.xls"

(Continued)

Object property	Data type	Description	Query expression example
operation[3]	String	The type of operation the result represents. Possible values in a response body are: • CREATED • DELETED • DISPOSED • PRUNED • PURGED • NOT_FOUND PRUNED and PURGED do not apply to objects in the default namespace. Results for object-based queries have either the CREATED or NOT_FOUND operation type. NOT_FOUND means that the object has been deleted from the repository but has not yet been removed from the index. The NOT_FOUND operation type is returned only for queries that specify **true** in the **verbose** entry.	N/A

(Continued)

Object property	Data type	Description	Query expression example
owner[2]	String	For objects in HCP namespaces, the user that owns the object. Valid values are: • For objects that have an owner: **USER,***location*,*username* • For objects with no owner: **GROUP,***location*,**all_users** • For objects that existed before the HCP system was upgraded from a pre-5.0 release and that have not subsequently been assigned an owner: **nobody** In these values: • *location* is the location in which the user account for the object owner is defined. This can be: - The name of an HCP tenant - The internal ID of an HCP tenant - An Active Directory domain preceded by an at sign (@) Internal IDs of HCP tenants are not returned in query results. For objects with no owner, *location* is the name of the tenant that owns the namespace in which the object is stored. • *username* is the name of the user that owns the object. This can be: - The username of a user account that's defined in HCP. - The username of an Active Directory user account. This can be either the user principal name or the Security Accounts Manager (SAM) account name for the user account. This property is not returned for objects in the default namespace. If the Authorization header or **hcp-ns-auth** cookie identifies a tenant-level user, you can specify this criterion in a query expression to find all objects owned by that user: owner:USER	owner:"USER,europe, pdgrey"

(Continued)

Object property	Data type	Description	Query expression example
owner[2] *(continued)*	String	These considerations apply when you specify the **owner** property in a query expression: • The entire value must be enclosed in double quotation marks. • **USER**, **GROUP**, and **nobody** are case sensitive. • The *location* values you specify are not case sensitive. • The *username* values you specify, except for **all_users**, are not case sensitive.	
permissions[3]	Integer	The octal value of the POSIX permissions for the object.	N/A
replicated[3]	Boolean	An indication of whether the object has been replicated. Possible values in a response body are: • **true** — The object, including the current version and all metadata, has been replicated. • **false** — The object has not been replicated.	N/A
replication Collision	Boolean	An indication of whether the object is flagged as a replication collision. Valid values are: • **true** — The object is flagged as a replication collision. • **false** — The object is not flagged as a replication collision.	replicationCollision:true
retention	Long	The end of the retention period for the object, in seconds since January 1, 1970, at 00:00:00 UTC. This value can also be: • **0** — Deletion Allowed • **-1** — Deletion Prohibited • **-2** — Initial Unspecified	retention:"-1"
retentionClass[4]	String	The name of the retention class assigned to the object. If the object is not assigned to a retention class, this value is an empty string in the query results. In query expressions, the values you specify for this property are case sensitive.	retentionClass:Reg-107

(Continued)

Object property	Data type	Description	Query expression example
retentionString[1]	String	The end of the retention period for this object in ISO 8601 format: *YYYY-MM-DD**T**hh:mm:ssZ* For more information on this format, see the description of the **accessTimeString** property. This value can also be one of these special values: • **Deletion Allowed** • **Deletion Prohibited** • **Initial Unspecified** In query expressions, these special values are case sensitive. In query results, this property also displays the retention class and retention offset, if applicable.	retentionString: "2015-03-02T 12:00:00-0500"
shred[2]	Boolean	An indication of whether the object will be shredded after it is deleted. Valid values are: • **true —** The object will be shredded. • **false —** The object will not be shredded.	shred:true
size	Long	The size of the object content, in bytes.	size:[2000 TO 3000]
type[3]	String	The object type. In a response body, this value is always **object**.	N/A
uid[3]	Integer	The POSIX user ID.	N/A
urlName[3]	String	The fully qualified object URL. For example: https://finance.europe.hcp.example.com/rest/Presentations/ Q1_2012.ppt	N/A
updateTime	Long	The value of the POSIX **mtime** attribute for the object, in seconds since January 1, 1970, at 00:00:00 UTC.	updateTime:[1309478400 TO 1312156800]
updateTimeString[1]	Datetime	The value of the POSIX **mtime** attribute for the object, in ISO 8601 format: *YYYY-MM-DD**T**hh:mm:ssZ* For more information on this format, see the description of the **accessTimeString** property.	updateTimeString: [2012-04-01 T00:00:00 TO 2012-04-30 T23:59:59]
utf8Name[4]	String	The UTF-8-encoded name of the object. In query expressions, the values you specify for this property are case sensitive.	utf8Name:23_John_ Doe.xls

(Continued)

Object property	Data type	Description	Query expression example
version	Unsigned long	The version ID of the object. All objects, including those in the default namespace, have version IDs. This property is not returned for objects with the NOT_FOUND operation type. For more information on this operation type, see the operation entry, above. When you specify the version ID of an old version in a query expression, HCP returns information about the current version of the object.	version:83920048912257
content-property-name[4]	Depends on property type	The value of a content property.	doctor_name: "John Smith"

1. HCP maintains the time for this property as a value that includes millisecond, but the property format uses seconds. As a result, specifying a single datetime value for this property in a query does not return all expected results. To retrieve all expected results, do one of these:

- Specify a range of values for this property.
- Specify a value for the corresponding long-type object property. For example, instead of specifying **ingestTimeString:2012-04-01T00:00:00**, specify **ingestTime:1333238400**.

2. You cannot specify a range of values for this property.

3. For object-based queries, you can specify this property only in the **objectProperties** entry. If you specify this property in either the **sort** or **query** entry, HCP returns a 400 (Bad Request) error.

4. You can use the asterisk (*) and question mark (?) wildcard characters when specifying values for this property.

4

Query responses

This chapter describes the response format for both object-based queries and operation-based queries.

Note: In some situations, when you specify one or more namespaces in a query request, the result may differ depending on whether the query is object-based or operation-based. For more information on these situations, see "Querying specified namespaces" on page 105.

Response body

The body of the response to a metadata query API request contains XML or JSON that lists the objects or operation records that match the request criteria. Object-based query responses list objects. Operation-based query responses return operation records.

Each object or operation record is specified by an **object** entry. The order in which **object** entries are listed in a response body depends on the type of query request:

- For object-based queries, **object** entries are listed in order according to the values specified in the **sort** request entry. If the request does not include the **sort** entry, **object** entries are listed by the number of search criteria they match. Objects that match the same number of criteria are not listed in any specific order.

- For operation-based queries, **object** entries are listed in ascending order based on change time.

All other entries in a response body are always listed in a fixed order.

XML response bodies

The format of an XML query response differs depending on the type of the query.

XML response body for object-based queries

An XML response for an object-based query has this format:

```
<?xml version='1.0' encoding='UTF-8'?>
<queryResult xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="/static/xsd/query-result-6.0.xsd">
    <query>
        <expression>query-request-entry</expression>
    </query>
    <resultSet>
        <object
            changeTimeMilliseconds="change-time-in-milliseconds.index"
            version="version-id"
            urlName="object-url"
            operation="operation-type"
            Additional properties if specified in the objectProperties request entry
                or if the verbose request entry specifies true
        />
        Additional object entries
```

```
</resultSet>
<status
    totalResults="total-object-count"
    results="returned-object-count"
    message=""
    code="COMPLETE|INCOMPLETE" />
The contentProperties entry below is included only if the request included a
contentProperties entry with a vlaue of true.
<contentProperties>
    <contentProperty>
        <expression>content-property-expression</expression>
        <name>content-property-name</name>
        <type>data-type</type>
        <multivalued>true|false</multivalued>
        <format>data-format</format>
    </contentProperty>
    Additional content properties
</contentProperties>
The facets entry below is included only if the request included a facets entry.
<facets>
    One or more of the following facet entries depending on the properties specified
    in the facets request entry
    <facet
        property="hold">
        <frequency
            count="object-count"
            value="true" />
        <frequency
            count="object-count"
            value="false" />
    </facet>
    <facet
        property="namespace">
        <frequency
            count="object-count"
            value="namespace-name.tenant-name" />
        Up to 99 additional frequency entries
    </facet>
    <facet
        property="retentionClass">
        <frequency
            count="object-count"
            value="retention-class-name" />
        Up to 99 additional frequency entries
    </facet>
    <facet
        property="retention">
        <frequency
            count="object-count"
```

```
                value="initialUnspecified" />
            <frequency
                count="object-count"
                value="neverDeletable" />
            <frequency
                count="object-count"
                value="expired" />
            <frequency
                count="object-count"
                value="not expired" />
        </facet>
        Zero or more of the following facet entries depending on the number of content
        properties in the facets request entry:
        <facet
            property="content-property-name">
            <frequency
                count="object-count"
                value="value-or-facet-range" />
            Up to 99 additional range frequency entries
        </facet>
    </facets>
</queryResult>
```

XML response body for operation-based queries

An XML response for an operation-based query has this format:

```
<?xml version='1.0' encoding='UTF-8'?>
<queryResult xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="/static/xsd/query-result-6.0.xsd">
    <query
        start="start-time-in-milliseconds"
        end="end-time-in-milliseconds" />
    <resultSet>
        <object
            changeTimeMilliseconds="change-time-in-milliseconds.index"
            version="version-id"
            urlName="object-url"
            operation="operation-type"
            Additional properties if specified in the objectProperties request entry
                or if the verbose request entry specifies true
        />
        Additional object entries
    </resultSet>
    <status
        results="returned-record-count"
        message=""
        code="COMPLETE|INCOMPLETE" />
    </queryResult>
```

JSON response bodies

The format of a JSON query response differs depending on the type of the query.

JSON response body for object-based queries

A JSON response for an object-based query has this format:

```
{
    "queryResult":{
        "query":{
            "expression":"query-request-entry"
        },
        "resultSet":[{
            {
                "urlName":"object-url",
                "operation":"operation-type",
                "changeTimeMilliseconds":"change-time-in-milliseconds.index",
                "version":version-id,
                Additional properties if specified in the objectProperties request entry or
                    if the verbose request entry specifies true
            },
            Additional object entries
        }],
        "status":{
            "totalResults":total-object-count,
            "results":returned-object-count,
            "message":"",
            "code":"COMPLETE|INCOMPLETE"
        },
        The contentProperties entry below is included only if the request included a
        contentProperties entry with a value of true
        "contentProperties":[{
            "contentProperty":{
                "expression":content-property-expression,
                "name":content-property-name,
                "type":data-type,
                "multivalued":true|false,
                "format":data-format,
            }
            Additional content properties
        }],
        The facets entry below is included only if the request included a facets entry.
```

```
"facets":{
    One or more of the following facet entries depending on the properties
    specified in the facets request entry:
    "facet":[{
        "property":"hold",
        "frequency":[{
            "value":"true",
            "count:object-count
        }, {
            "value":"false",
            "count":object-count
        }]
    }, {
        "property":"namespace",
        "frequency":[{
            "value":"namespace-name.tenant-name",
            "count":object-count
        Up to 99 additional value properties
        }]
    }, {
        "property":"retentionClass",
        "frequency":[{
            "value":"retention-class-name",
            "count":object-count
        Up to 99 additional value properties
        }]
    }, {
        "property":"retention",
        "frequency":[{
            "value":"initialUnspecified",
            "count":object-count
        },{
            "value":"neverDeletable",
            "count":object-count
        },{
            "value":"expired",
            "count":object-count
        },{
            "value":"not expired",
            "count":object-count
        }]
```

```
                    Zero or more of the following facet entries depending on whether the
                    number of defined content properties in the facets request entry.
                },{
                    property:"content--property-name",
                    frequency:[{
                        count:"object-count",
                        value:"value-or-facet-range"
                    },{
                        Up to 99 additional range frequency entries
                    }
                }]
            }]
        }
}
```

JSON response body for operation-based queries

A JSON response for an operation-based query has this format:

```
{
    "queryResult":{
        "query":{
            "end":end-time-in-milliseconds,
            "start":start-time-in-milliseconds
        },
        "resultSet":[{
            {
                "urlName":"object-url",
                "operation":"operation-type",
                "changeTimeMilliseconds":"change-time-in-milliseconds.index",
                "version":version-id,
                Additional properties if specified in the objectProperties request entry or
                    if the verbose request entry specifies true
            },
            Additional object entries
        }],
        "status":{
            "results":returned-record-count,
            "message":"",
            "code":"COMPLETE|INCOMPLETE"
        }
    }
}
```

Response body contents

Both XML and JSON have a single top-level **queryResult** entry. The **queryResult** entry contains one of each of the entries listed in the table below.

Entry	Description
query	For object-based queries, a container for the query expression. For operation-based queries, a specification of the time period that the query covers. The results include only operation records for objects with change times during this period. For more information on this entry, see "query entry" below.
resultSet	A container for the set of **object** entries representing the objects or operation records that match the query. For more information on this entry, see "resultSet entry" on page 69.
status	Information about the response, including the number of returned records and whether the response completes the query results. For more information on this entry, see "status entry" on page 70.
contentProperties *(object-based queries only)*	For queries that contained a **contentProperties** entry with a value of **true**, a list of the available content properties. For more information on this entry see "contentProperties entry" on page 71.
facets *(object-based queries only)*	Summary information about property values that appear in the result set. For more information on this entry, see "facets entry" on page 71. This entry is returned only if the query request included the **facets** entry.

query entry

The **query** entry contains the entry and properties described in the table below.

Entry/Property	Description
expression *(entry, object-based queries only)*	A container for value of the **query** request entry.

(Continued)

Entry/Property	Description
start *(property, operation-based queries only)*	The value of the **start** request property, in milliseconds since January 1, 1970, at 00:00:00 UTC. If you omitted the **start** entry in the request, this value is 0 (zero).
end *(property, operation-based queries only)*	The value of the **end** request property, in milliseconds since January 1, 1970, at 00:00:00 UTC. If you omitted the **end** entry in the request, this value is one minute before the time HCP received the request.

resultSet entry

The **resultSet** entry has one child **object** entry for each object or operation record that matches the query criteria.

Note: The metadata query API does not return results for open objects (that is, objects that are still being written or were never closed).

object entry

In XML, the **object** entries are child elements of the **resultSet** entry. In JSON, the **object** entries are unnamed objects in the **resultSet** entry.

The information that the **object** entry provides depends on the type of the query request:

- For object-based queries, each **object** entry provides information about an individual object.
- For operation-based queries, each **object** entry provides information about an individual create, delete, dispose, prune, or purge operation and the object affected by the operation.

The **object** entry always contains these object properties:

- changeTimeMilliseconds
- operation
- urlName
- version

The **object** entry can contain other object properties depending on the value of the **verbose** request entry or the value of the **objectProperties** request entry.

For more information the properties that this entry can contain, see "Object properties" on page 52.

status entry

The **status** entry has the properties listed in the table below.

Property	Description
code	An indication of whether all results have been returned: • **COMPLETE** — All results have been returned. This value is returned if the response includes all results or if the response includes the last result for a paged query. • **INCOMPLETE** — Not all results have been returned. This value is returned if any of these apply: - The **count** request entry is smaller than the number of objects or operation records that meet the query criteria. - For object-based queries, the **count** request entry is not specified and more than one hundred objects meet the query criteria. - For operation-based queries, the **count** request entry is not specified and more than ten thousand operation records meet the query criteria. - The response is incomplete due to an error encountered in executing the query. You can retrieve additional results by resubmitting the request with an **offset** entry (for object-based queries) or a **lastResult** entry (for operation-based queries). For more information on these techniques, see "Paged queries with object-based requests" on page 42 and "Paged queries with operation-based requests" on page 51.
message	Always an empty string.
results	The number of results returned.
totalResults	For object-based queries, the total number of indexed objects that meet the query criteria. This property is not returned for operation-based queries.

contentProperties entry

If the request included a **contentProperties** entry with a value of **true**, the result has a **contentProperties** entry containing zero or more **contentProperty** entries. Each **contentProperty** entry contains the entries listed in the table below.

Entry	Description
expression	The expression that specifies how HCP locates the property value in the custom metadata XML.
format	The pattern used to parse a number or date value in the XML custom metadata. For example, the format used for dollar values for a content property with a type of float might be $#,##0.00. This entry is included for integer, float, and date types only.
multivalued	An indication of whether the property can have multiple values. For an example of multivalued content properties, see "Sorting on content properties" on page 23.
name	The content property name.
type	The content property data type. One of: • BOOLEAN • DATE • FLOAT • INTEGER • TOKENIZED (full-text searchable string) • STRING

facets entry

The **facets** response entry has one or more child **facet** entries, as described in the table below.

Entry/Property	Description
facet	Child of the **facets** entry. This entry contains the **property** property and one or more **frequency** entries.

(Continued)

Entry/Property	Description
property *(property)*	Property of the **facet** entry. The value for this property is one of: • **hold** • **namespace** • **retention** • **retentionClass** • *content-property-name*
frequency *(child)*	Child of the **facet** entry. This entry contains the **count** and **value** properties. This entry is returned only for property values that appear in the result set. **frequency** entries are listed in descending order based on the value of the **count** property. A query response can contain a maximum of one hundred **frequency** entries.
count (*property*)	The number of objects in the result set with the property value identified by the **value** property.

(Continued)

Entry/Property	Description
value *(property)*	An object property value that applies to one or more objects in the result set. The value of this property depends on the **property** value of the parent facet entry. When the value of the parent **facet** entry **property** property is: • **hold**, this value is either **true** or **false**. • **retention**, this value is one of: - **initialUnspecified** — For objects with a retention setting of Initial Unspecified - **neverDeletable** — For objects with a retention setting of Deletion Prohibited - **expired** — For objects with a retention setting that is Deletion Allowed or a specific date in the past - **not expired** — For objects with a retention setting that is a specific date in the future • **retentionClass**, this value is the name of a retention class for an object in the result set. • **namespace**, this value identifies a namespace that contains an object in the result set. The value has this format: *namespace-name.tenant-name* • *content-property-name*, this value is a value of the named content property that occurs in the result set.

HTTP return codes

The table below describes the possible HTTP return codes for metadata query API requests.

Code	Meaning	Description
200	OK	HCP successfully processed the query.

(Continued)

Code	Meaning	Description
400	Bad Request	The request syntax is invalid. Possible reasons for this error include: • The query request contains an invalid URL query parameter. • The query request body contains invalid XML or JSON (for example, an invalid entry name). • The query request body contains an invalid entry value, such as a malformed version ID or invalid directory path. • One of the **sort**, **facet**, **query**, or **objectProperties** request entries contains an invalid object property. For information on object properties and the request entries in which they are supported, see "Object properties" on page 52. • The request contains a Content-Encoding header that specifies gzip, but the request body is not in gzip-compressed format. • The cURL **-d** option is specified instead of the **--data-binary** option with a request body in gzip-compressed format • For object-based queries, the **query** request entry specifies a query expression that is not in UTF-8 format. • For operation-based queries, the query request specifies a namespace that does not exist. • For object-based queries, HCP has insufficient memory to process and return query results. To avoid this error, do one or more of these: - Specify more precise query criteria to return fewer results. - Omit the **sort** request entry. - Omit the **facets** request entry. If more information about the error is available, the response includes the HCP-specific X-HCP-ErrorMessage HTTP header.

(Continued)

Code	Meaning	Description
403	Forbidden	One of: • The request does not include an Authorization header or **hcp-ns-auth** cookie. • The Authorization header or **hcp-ns-auth** cookie specifies invalid credentials. • The Authorization header or **hcp-ns-auth** cookie specifies credentials for a system-level user account that is not configured to allow use of the metadata query API. • The Authorization header or **hcp-ns-auth** cookie specifies credentials for a system-level user account, but the URL specifies an HCP tenant that has not granted administrative access to system-level users. • For operation-based queries, the Authorization header or **hcp-ns-auth** cookie specifies credentials for a tenant-level user, but the query specifies a namespace for which that user account does not have search permission. • For operation-based queries, the Authorization header or **hcp-ns-auth** cookie specifies credentials for a system-level user account that is configured to allow use of the metadata query API and the URL specifies **admin**, but the request body specifies a namespace in a tenant that has not granted administrative access to system-level users. • The tenant specified in the URL does not exist. If more information about the error is available, the response includes the HCP-specific X-HCP-ErrorMessage HTTP header.
406	Not Acceptable	One of: • The request does not have an Accept header, or the Accept header does not specify application/xml or application/json. • The request has an Accept-Encoding header that does not specify gzip or *.

(Continued)

Code	Meaning	Description
415	Unsupported Media Type	One of: • The request does not have a Content-Type header, or the Content-Type header does not specify application/xml or application/json. • The request has a Content-Encoding header with a value other than gzip.
500	Internal Server Error	An internal error occurred. Try the request again, gradually increasing the delay between each successive attempt. If this error happens repeatedly, contact your tenant administrator.
503	Service Unavailable	HCP is temporarily unable to handle the request, probably due to system overload, maintenance, or upgrade. Try the request again, gradually increasing the delay between each successive attempt.

HTTP response headers

The response to a valid query request includes a Transfer-Encoding header with a value of chunked and an Expires header with a value of Thu, 01 Jan 1970 00:00:00 GMT.

If the query request specifies a gzip-compressed response, the response includes a Content-Encoding header with a value of gzip.

If HCP can provide additional information about an invalid query request, the response has an X-HCP-ErrorMessage header describing the error.

5

Examples

This chapter contains examples of both object-based and operation-based queries. The examples show some of the ways you can use the metadata query API to get information about namespace content. Specifically, they show how to:

- With object-based queries:
  - Query for objects based on custom metadata content
  - Use a paged query to retrieve a list of all objects in a namespace
  - Use a faceted query to retrieve summary information about objects
  - Query for objects that are flagged as replication collisions
  - Retrieve a list of content properties
- With operation-based queries:
  - Retrieve all operation records for all existing and deleted objects in a directory
  - Retrieve basic metadata for objects that changed during a specific time period
  - Use a paged query to retrieve a large number of operation records
  - Check whether the namespaces owned by a tenant contain any objects that are flagged as replication collisions

Object-based query examples

This section contains examples of object-based queries.

Example 1: Querying for custom metadata content

Here's a sample metadata query API request that retrieves metadata for all objects that:

- Are in namespaces owned by the europe tenant
- Have custom metadata that contains an element named **department** with a value of Accounting

The query uses an XML request body and requests results in JSON format.

In addition to the basic information about the objects in the result set, this request returns the shred and retention settings for each object in the result set. The request also specifies that objects in the result set be listed in reverse chronological order based on change time.

Request body in the XML file named Accounting.xml

```
<queryRequest>
    <object>
        <query>customMetadataContent:
            "department.Accounting.department"
        </query>
        <objectProperties>shred,retention</objectProperties>
        <sort>changeTimeMilliseconds+desc</sort>
    </object>
</queryRequest>
```

Request with cURL command line

```
curl -k -H "Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d"
    -H "Content-Type: application/xml" -H "Accept: application/json"
    -d @Accounting.xml "https://europe.hcp.example.com/query?prettyprint"
```

Request in Python using PycURL

```
import pycurl
import os
curl = pycurl.Curl()

# Set the URL, command, and headers
curl.setopt(pycurl.URL, "https://europe.hcp.example.com/" +
```

```
        "query?prettyprint")
    curl.setopt(pycurl.SSL_VERIFYPEER, 0)
    curl.setopt(pycurl.SSL_VERIFYHOST, 0)
    curl.setopt(pycurl.POST, 1)
    curl.setopt(pycurl.HTTPHEADER,
        ["Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d",
        "Content-Type: application/xml", "Accept: application/json"])

    # Set the request body from an XML file
    filehandle = open("Accounting.xml", 'rb')
    curl.setopt(pycurl.UPLOAD, 1)
    curl.setopt(pycurl.CUSTOMREQUEST, "POST")
    curl.setopt(pycurl.INFILESIZE,
             os.path.getsize("Accounting.xml"))
    curl.setopt(pycurl.READFUNCTION, filehandle.read)

    curl.perform()
    print curl.getinfo(pycurl.RESPONSE_CODE)
    curl.close()
    filehandle.close()
```

Request headers

```
POST /query?prettyprint HTTP/1.1
Host: europe.hcp.example.com
Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d
Content-Type: application/xml
Accept: application/json
Content-Length: 192
```

Response headers

```
HTTP/1.1 200 OK
Server: HCP V7.0.0.16
Transfer-Encoding: chunked
```

JSON response body

To limit the example size, the JSON below shows only one object in the result set.

```
{"queryResult:
    {"query":
        {"expression":"customMetadataContent:
            "department.Accounting.department""},
    "resultSet":[
        {"version":84689494804123,
        "operation":"CREATED",
        "urlName":"https://finance.europe.hcp.example.com/rest/presentations/
            Q1_2012.ppt",
```

```
            "changeTimeMilliseconds":"1334244924615.00",
            "retention":0,
            "shred":false},
        .
        .
        .
        ],
        "status":{
            "message":"",
            "results":12,
            "code":"COMPLETE"}
        }
    }
```

Custom metadata file for the Q1_2012.ppt object

```
<?xml version="1.0">
<presentation>
    <presentedBy>Lee Green</presentedBy>
    <department>Accounting</department>
    <slides>23</slides>
    <date>04-01-2012</date>
</presentation>
```

Example 2: Using a paged query to retrieve a list of all objects in a namespace

The Java® example below implements a paged query that uses multiple requests to retrieve all objects in a namespace. The example returns metadata for fifty objects per request and also returns information about the size and ingest time of each object in the result set.

This example uses the **com.hds.hcp.apihelpers.query** Java class infrastructure, which uses the Jackson JSON processor to produce a JSON query request body and consume a JSON query response. To limit the example size, the example does not include the source code for this infrastructure.

The Jackson JSON processor serializes and deserializes JSON formatted content with Java Objects. For more information on the Jackson JSON processor, see http://jackson.codehaus.org.

```
package com.hds.hcp.examples;

import java.util.List;
import java.io.BufferedReader;
import java.io.InputStreamReader;
```

```
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.HttpResponseException;
import org.apache.http.client.methods.*;
import org.apache.http.entity.StringEntity;
import org.apache.http.util.EntityUtils;

/* General purpose helper routines for samples */
import com.hds.hcp.apihelpers.HCPUtils;

/* Provide for helper routines to encapsulate the queryRequest and queryResults. */
import com.hds.hcp.apihelpers.query.request.Object;
import com.hds.hcp.apihelpers.query.request.QueryRequest;
import com.hds.hcp.apihelpers.query.result.Status;
import com.hds.hcp.apihelpers.query.result.QueryResult;
import com.hds.hcp.apihelpers.query.result.ResultSetRecord;

public class PagedObjectQuery {

  // Local member variables
  private Boolean bIsInitialized = false;
  private String sQueryTenant;
  private String sQueryNamespace;
  private String sEncodedUserName, sEncodedPassword;
  private String sHTTPQueryURL;

  private HttpClient mHttpClient;

  /**
  * Initialize the object by setting up internal data and establishing the HTTP client
  * connection.
  *
  * This routine is called by the ReadFromHCP and WriteToHCP routines, so calling it
  * by the consumer of this class is unnecessary.
  */
  void initialize(String inNamespace, String inUsername, String inPassword) throws
  Exception {

    if (! bIsInitialized)  // Initialize only if we haven't already
    {
      // Break up the namespace specification to get the namespace and tenant parts.
      String parts[] = inNamespace.split("\\.");

      sQueryNamespace = parts[0];
      sQueryTenant = parts[1];

      // Now extract just the tenant part of the URL and use it to create the
      // HTTPQueryURL.
      parts = inNamespace.split(sQueryNamespace + "\\.");

      sHTTPQueryURL = "https://" + parts[1] + "/query";
```

```
      // Encode both the username and password for the authentication string.
      sEncodedUserName = HCPUtils.toBase64Encoding(inUsername);
      sEncodedPassword = HCPUtils.toMD5Digest(inPassword);

      // Set up an HTTP client for sample usage.
      mHttpClient = HCPUtils.initHttpClient();

      bIsInitialized = true;
    }
  }

  /**
   * This method performs an orderly shutdown of the HTTP connection manager.
   */
  void shutdown() throws Exception {
    // Clean up open connections by shutting down the connection manager.
    mHttpClient.getConnectionManager().shutdown();
  }

  /**
   *  This routine issues a query to an HCP namespace requesting information about
   *  objects in it. The query requests 1,000 results at a time. If there are more,
   *  the routine performs paged queries to retrieve all the results.
   *
   *  While processing the query results, the routine displays the name of the first
   *  and last object of the result set to system output.
   */
  protected void runQuery() {

    // Statistics counters
    Long TotalRecordsProcessed = 0L;
    Integer HTTPCalls = 0;

    try {
      /*
       * Set up the query request.
       */

      // Set up for an object query by calling the
      // com.hds.hcp.apihelpers.query.request.Object constructor.
      Object mObjQuery = new Object();

      // Get only 50 objects at a time.
      mObjQuery.setCount(50);

      // Retrieve only those that reside in the namespace specified in the command.
      mObjQuery.setQuery("+namespace:" + sQueryNamespace + "." + sQueryTenant);

      // Retrieve the "size" and "ingestTimeString" properties for the object.
      mObjQuery.setObjectProperties("size,ingestTimeString");
```

```
// Set up the query request.
QueryRequest mQuery = new QueryRequest(mObjQuery);

/*
 * Loop through and process all the objects one response at a time or until
 * an error occurs.
 */
QueryResult mQueryResult = null;
do {
  System.out.println("Issuing query: \n" + mQuery.toString(true));

  /*
   * Execute the query using the HTTP POST method.
   */
  HttpPost httpRequest = new HttpPost(sHTTPQueryURL);

  // Add the body of the POST request.
  httpRequest.setEntity(new StringEntity(mQuery.toString()));

  // Set the Authorization header.
  httpRequest.setHeader("Authorization: HCP " + sEncodedUserName + ":"
    + sEncodedPassword);

  // Indicate that the input and output are in JSON format.
  httpRequest.setHeader("Content-Type", "application/json");
  httpRequest.setHeader("Accept", "application/json");

  // Execute the query.
  HttpResponse httpResponse = mHttpClient.execute(httpRequest);

  // For debugging purposes, dump out the HTTP response.
  HCPUtils.dumpHttpResponse(httpResponse);

  // If the return code is anything BUT in the 200 range indicating success,
  // throw an exception.
  if (2 != (int)(httpResponse.getStatusLine().getStatusCode() / 100))
  {
    // Clean up after ourselves and release the HTTP connection to the
    // connection manager.
    EntityUtils.consume(httpResponse.getEntity());

    throw new HttpResponseException(httpResponse.getStatusLine()
    .getStatusCode(),
      "Unexpected status returned from " + httpRequest.getMethod() + " ("
      + httpResponse.getStatusLine().getStatusCode() + ": "
      + httpResponse.getStatusLine().getReasonPhrase() + ")");
  }

  /*
   *  Process the response from the query request.
   */
```

```
      // Put the response in a buffered reader.
      BufferedReader bodyReader = new BufferedReader(newInputStreamReader
        (httpResponse.getEntity().getContent()));
      HTTPCalls += 1;

      // Parse the response into the QueryResult object.
      mQueryResult = QueryResult.parse(bodyReader);

      // Get a copy of the query status from the query result.
      Status mStatus = mQueryResult.getStatus();

      // Display the status of what we just accomplished.
      System.out.println();
      System.out.println("Batch " + HTTPCalls + " Status: " + mStatus.getCode()
        + " Record Count:" + mStatus.getResults());

      // Display the first and last object of the result set.
      List<ResultSetRecord> mResultSet = mQueryResult.getResultSet();
      ResultSetRecord mFirstRecord = mResultSet.get(0);

      System.out.println("  First Record (" + (TotalRecordsProcessed+1) + ") "
        + mFirstRecord.getUrlName());
      System.out.println("        Size: " + mFirstRecord.getSize());

      TotalRecordsProcessed += mStatus.getResults();

      ResultSetRecord mLastRecord = mResultSet.get(mResultSet.size()-1);
      System.out.println("  Last Record  (" + TotalRecordsProcessed
        + ") "+ mLastRecord.getUrlName());
      System.out.println("        Size: " + mLastRecord.getSize());
      System.out.println();

      // Now we need to see whether the query is complete or whether there are more
      // objects.  If INCOMPLETE, it is a successful paged query.
      if (Status.Code.INCOMPLETE == mStatus.getCode())
      {
        // We have more, so update the offset for the next query to be the previous
        // offset plus the number we just read.
        mObjQuery.setOffset(
          (null == mObjQuery.getOffset() ? 0 : mObjQuery.getOffset())
          + mStatus.getResults()
          );
      }

      // Clean up after ourselves and release the HTTP connection to the connection
      // manager.
      EntityUtils.consume(httpResponse.getEntity());

    } // Keep doing this while we have more results.

    while (Status.Code.INCOMPLETE == mQueryResult.getStatus().getCode());
```

```
      /*
       * Print out the final statistics.
       */
      System.out.println("Total Records Processed: " + TotalRecordsProcessed);
      System.out.println("HTTP Calls: " + HTTPCalls);
    } catch(Exception e) {
      e.printStackTrace();
    }
  }

  /*
   * @param args
   */
  public static void main(String[] args) {

    PagedObjectQuery myClass = new PagedObjectQuery();

    if (args.length != 3) {
      System.out.println();
      System.out.println("Usage: " + myClass.getClass().getSimpleName()
        + " <DNS-Namespace> <Username> <Password>\n");
      System.out.println("  where ");
      System.out.println("    <DNS-Namespace> is the fully qualified domain name"
        + "  of the HCP Namespace.");
      System.out.println("        For example: \"ns1.ten1.myhcp.example.com\"");
      System.out.println("    <Username> and <Password> are the credentials of the"
        + " HCP user with data access permissions for the namespace");
      System.out.println();

      System.exit(-1);
    }

    try {
      // Initialize the class with the input parameters
      myClass.initialize(args[0], args[1], args[2]);

      // Issue the query and process the results
      myClass.runQuery();

      // Clean up before object destruction
      myClass.shutdown();

    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Example 3: Using a faceted query to retrieve object information

Here's a sample metadata query API request that retrieves metadata for all objects added to namespaces owned by the europe tenant between March 1, 2012, and March 31, 2012, inclusive. The **verbose** entry specifies **true** to request all metadata for each object in the result set. This request also retrieves namespace facet information for objects in the result set. The query uses an XML request body and requests results in XML format.

Request body in the XML file named March.xml

```
<queryRequest>
    <object>
        <query>ingestTime:[1330560000 TO 1333238399]</query>
        <facets>namespace</facets>
        <verbose>true</verbose>
    </object>
</queryRequest>
```

Request with cURL command line

```
curl -k -H "Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d"
    -H "Content-Type: application/xml" -H "Accept: application/xml"
    -d @March.xml "https://europe.hcp.example.com/query?prettyprint"
```

Request in Python using PycURL

```
import pycurl
import os
curl = pycurl.Curl()

# Set the URL, command, and headers
curl.setopt(pycurl.URL, "https://europe.hcp.example.com/" +
    "query?prettyprint")
curl.setopt(pycurl.SSL_VERIFYPEER, 0)
curl.setopt(pycurl.SSL_VERIFYHOST, 0)
curl.setopt(pycurl.POST, 1)
curl.setopt(pycurl.HTTPHEADER,
    ["Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d",
    "Content-Type: application/xml", "Accept: application/xml"])

# Set the request body from an XML file
filehandle = open("March.xml", 'rb')
curl.setopt(pycurl.UPLOAD, 1)
curl.setopt(pycurl.CUSTOMREQUEST, "POST")
curl.setopt(pycurl.INFILESIZE,
         os.path.getsize("March.xml"))
curl.setopt(pycurl.READFUNCTION, filehandle.read)

curl.perform()
```

```
print curl.getinfo(pycurl.RESPONSE_CODE)
curl.close()
filehandle.close()
```

Request headers

```
POST /query?prettyprint HTTP/1.1
Host: europe.hcp.example.com
Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d
Content-Type: application/xml
Accept: application/xml
Content-Length: 134
```

Response headers

```
HTTP/1.1 200 OK
Server: HCP V7.0.0.16
Transfer-Encoding: chunked
```

XML response body

To limit the example size, the XML below shows only one **object** entry in the response body.

```
<?xml version='1.0' encoding='UTF-8'?>
<queryResult xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="/static/xsd/query-result-7.0.xsd">
    <query>
        <expression>ingestTime:[1333238400 TO 1335830399]</expression>
    <resultSet>
        <object
            version="84689595801123"
            utf8Name="Q1_2012.ppt"
            urlName="https://marketing.europe.hcp.example.com/rest/
                presentations/Q1_2012.ppt"
            updateTimeString="2012-03-31T15:41:35-0400"
            updateTime="1333222895"
            uid="0"
            type="object"
            size="6628"
            shred="false"
            retentionString="Deletion Allowed"
            retentionClass=" "
            retention="0"
            replicated="true"
            permissions="555"
            owner="USER,europe,lgreen"
            operation="CREATED"
            namespace="marketing.europe"
```

```
                ingestTimeString="2012-03-31T15:41:35-0400"
                ingestTime="1333222895"
                index="true"
                hold="false"
                hashScheme="SHA-256"
                hash="SHA-256 0662D2A2DEF74EF02A8DF5A4F16BF4D55FEE582..."
                gid="0"
                objectPath="/presentations/Q1_2012.ppt"
                dpl="2"
                customMetadata="false"
                changeTimeString="2012-03-31T15:41:35-0400"
                changeTimeMilliseconds="1333222895615.00"
                accessTimeString="2012-03-31T15:41:35-0400"
                accessTime="1333222895"
                acl="false" />
            .
            .
            .
        </resultSet>
        <status
            results="7"
            message=""
            code="COMPLETE" />
        <facets>
            <facet
                property="namespace
                <frequency
                    count="4"
                    value="finance.europe" />
                <frequency
                    count="3"
                    value="marketing.europe" />
            </facet>
        </facets>
    </queryResult>
```

Example 4: Querying for replication collisions in a namespace

Here's a sample metadata query API request that retrieves metadata for all objects that are:

- Flagged as replication collisions
- In the finance namespace owned by the europe tenant

The query uses an XML request body and requests results in XML format.

This request returns only the URL, version ID, operation type, and change time for the objects in the result set. The request specifies that the result set be sorted by object path in ascending order.

Request body in the XML file named FinanceCollisions.xml

```
<queryRequest>
    <object>
        <query>
            +namespace:finance.europe
            +replicationCollision:true
        </query>
        <objectProperties>objectPath</objectProperties>
        <sort>objectPath+asc</sort>
    </object>
</queryRequest>
```

Request with cURL command line

```
curl -k -H "Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d"
    -H "Content-Type: application/xml" -H "Accept: application/xml"
    -d @FinanceCollisions.xml "https://europe.hcp.example.com/query?prettyprint"
```

Request in Python using PycURL

```
import pycurl
import os
curl = pycurl.Curl()

# Set the URL, command, and headers
curl.setopt(pycurl.URL, "https://europe.hcp.example.com/" +
    "query?prettyprint")
curl.setopt(pycurl.SSL_VERIFYPEER, 0)
curl.setopt(pycurl.SSL_VERIFYHOST, 0)
curl.setopt(pycurl.POST, 1)
curl.setopt(pycurl.HTTPHEADER,
    ["Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d",
    "Content-Type: application/xml", "Accept: application/xml"])

# Set the request body from an XML file
filehandle = open("FinanceCollisions.xml", 'rb')
curl.setopt(pycurl.UPLOAD, 1)
curl.setopt(pycurl.CUSTOMREQUEST, "POST")
curl.setopt(pycurl.INFILESIZE,
         os.path.getsize("FinanceCollisions.xml"))
curl.setopt(pycurl.READFUNCTION, filehandle.read)

curl.perform()
```

```
print curl.getinfo(pycurl.RESPONSE_CODE)
curl.close()
filehandle.close()
```

Request headers

```
POST /query?prettyprint HTTP/1.1
Host: europe.hcp.example.com
Content-Type: application/xml
Accept: application/xml
Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d
Content-Length: 205
```

Response headers

```
HTTP/1.1 200 OK
Server: HCP V7.0.0.16
Transfer-Encoding: chunked
```

XML response body

```
<?xml version='1.0' encoding='UTF-8'?>
<queryResult xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="/static/xsd/query-result-7.0.xsd">
<query>
    <expression>+namespace:t1-ns2.LisaTenant-1  +replicationCollision:true
</expression>
</query>
<resultSet>
    <object
        version="89322738450881"
        urlName="https://finance.europe.hcp.example.com/rest/budgets/2014/
            sales_budget_2014.xlsx.collision"
        operation="CREATED"
        objectPath="/budgets/2014/sales_budget_2014.xlsx.collision"
        changeTimeMilliseconds="1395668086005.00" />
    <object
        version="89322749144130"
        urlName="https://finance.europe.hcp.example.com/rest/quarterly_rpts/
            Q3_2013.ppt.collision"
        operation="CREATED"
        objectPath="/quarterly_rpts/Q3_2013.ppt.collision"
        changeTimeMilliseconds="1395668327386.00" />
</resultSet>
<status
    totalResults="2"
    results="2"
```

```
    message=""
    code="COMPLETE" />
</queryResult>
```

Example 5: Listing content properties

Here's a sample metadata query API request that lists the content properties for all indexed objects in the medical namespace owned by the employees tenant. The query uses an XML request body and requests results in XML format.

Request body in the XML file named MedicalQuery.xml

```
<queryRequest>
    <object>
        <query>namespace:medical.employees</query>
        <count>0</count>
        <contentProperties>true</contentProperties>
    </object>
</queryRequest>
```

Request with cURL command line

```
curl -i -k -H "Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d"
     -H "Content-Type: application/xml" -H "Accept: application/xml"
     -d @MedicalQuery.xml "https://employees.hcp.example.com/
        query?prettyprint"
```

Request in Python using PycURL

```
import pycurl
import os
curl = pycurl.Curl()

# Set the URL, command, and headers
curl.setopt(pycurl.URL, "https://employees.hcp.example.com/" +
    "query?prettyprint")
curl.setopt(pycurl.SSL_VERIFYPEER, 0)
curl.setopt(pycurl.SSL_VERIFYHOST, 0)
curl.setopt(pycurl.POST, 1)
curl.setopt(pycurl.HTTPHEADER,
    ["Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d",
    "Content-Type: application/xml", "Accept: application/xml"])

# Set the request body from an XML file
filehandle = open("MedicalQuery.xml", 'rb')
curl.setopt(pycurl.UPLOAD, 1)
curl.setopt(pycurl.CUSTOMREQUEST, "POST")
curl.setopt(pycurl.INFILESIZE,
```

```
            os.path.getsize("MedicalQuery.xml"))
    curl.setopt(pycurl.READFUNCTION, filehandle.read)

    curl.perform()
    print curl.getinfo(pycurl.RESPONSE_CODE)
    curl.close()
    filehandle.close()
```

Request headers

```
POST /query?prettyprint HTTP/1.1
Host: employees.example.com
Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d
Content-Type: application/xml
Accept: application/xml
Content-Length: 155
```

Response headers

```
HTTP/1.1 200 OK
Server: HCP V7.0.0.16
Transfer-Encoding: chunked
```

XML response body

To limit the example size, the XML below shows only two **contentProperty** entries in the response body.

```
<?xml version='1.0' encoding='UTF-8'?>
<queryResult xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="/static/xsd/query-result-7.0.xsd">
<query>
    <expression>namespace:medical.employees</expression>
</query>
<resultSet />
<status
    totalResults="0"
    results="0"
    message=""
    code="COMPLETE" />
<contentProperties>
    <contentProperty>
        <name>DocDateOfBirth</name>
        <expression>/record/doctor/dob</expression>
        <type>DATE</type>
        <multivalued>false</multivalued>
        <format>MM/dd/yyy</format>
    </contentProperty>
    <contentProperty>
```

```
            <name>DocLastName</name>
            <expression>/record/doctor/name/lastName</expression>
            <type>STRING</type>
            <multivalued>false</multivalued>
            <format></format>
        </contentProperty>
    </contentProperties>
    </queryResult>
```

Operation-based query examples

This section contains examples of operation-based queries.

Example 1: Retrieving all operation records for all existing and deleted objects in a directory

Here's a sample metadata query API request that retrieves operation records for all objects currently in or deleted from the sales namespace owned by the midwest tenant. The query uses an XML request body and requests results in JSON format.

The **verbose** entry is set to true to request detailed information for all operation records in the result set.

The response body includes records for all create, delete, and purge operations that occurred since the namespace was created up to one minute before the request was made at March 14, 2013 at 14:59:37 EST.

Request body in the XML file named AllSales.xml

```
<queryRequest>
    <operation>
        <count>-1</count>
        <systemMetadata>
            <namespaces>
                <namespace>sales.midwest</namespace>
            </namespaces>
        </systemMetadata>
        <verbose>true</verbose>
    </operation>
</queryRequest>
```

Request with cURL command line

```
curl -i -k -H "Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d"
    -H "Content-Type: application/xml" -H "Accept: application/json"
    -d @AllSales.xml "https://midwest.hcp.example.com/query?prettyprint"
```

Request in Python using PycURL

```
import pycurl
import os
curl = pycurl.Curl()

# Set the URL, command, and headers
curl.setopt(pycurl.URL, "https://midwest.hcp.example.com/" +
    "query?prettyprint")
curl.setopt(pycurl.SSL_VERIFYPEER, 0)
curl.setopt(pycurl.SSL_VERIFYHOST, 0)
curl.setopt(pycurl.POST, 1)
curl.setopt(pycurl.HTTPHEADER,
    ["Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d",
    "Content-Type: application/xml", "Accept: application/json"])

# Set the request body from an XML file
filehandle = open("AllSales.xml", 'rb')
curl.setopt(pycurl.UPLOAD, 1)
curl.setopt(pycurl.CUSTOMREQUEST, "POST")
curl.setopt(pycurl.INFILESIZE,
         os.path.getsize("AllSales.xml"))
curl.setopt(pycurl.READFUNCTION, filehandle.read)

curl.perform()
print curl.getinfo(pycurl.RESPONSE_CODE)
curl.close()
filehandle.close()
```

Request headers

```
POST /query HTTP/1.1
Host: finance.hcp.example.com
Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d
Content-Type: application/xml
Accept: application/json
Content-Length: 258
```

Response headers

```
HTTP/1.1 200 OK
Server: HCP V7.0.0.16
Transfer-Encoding: chunked
```

JSON response body

To limit the example size, the JSON below shows only one **object** entry in the response body.


```
{"queryResult":
    {"query":{"start":0,"end":1331751577658},
     "resultSet":[
        {"version":81787144560449
         "utf8Name":"C346527",
         "urlName":"https://sales.midwest.hcp.example.com/rest/
                customers/widgetco/orders/C346527",
         "updateTimeString":"2012-03-10T14:55:33-0500"
         "updateTime":1331409333,
         "uid":0,
         "type":"object",
         "size":4985,
         "shred":false,
         "retentionString":"Deletion Allowed",
         "retentionClass":"",
         "retention":"0",
         "replicated":true,
         "permissions":"256"
         "owner":"USER,midwest,rblack"
         "operation":"CREATED",
         "namespace":"sales.midwest",
         "ingestTimeString":"2012-03-10T14:55:33-0500",
         "ingestTime":1331409333,
         "index":true,
         "hold":false,
         "hashScheme":"SHA-256",
         "hash":"SHA-256 C67EF26C0E5EDB102A2DEF74EF02A8DF5A4F16BF4D...",
         "gid":0,
         "objectPath":"/customers/widgetco/orders/C346527",
         "dpl":2,
         "customMetadata":false,
         "changeTimeString":"2012-03-10T14:55:33-0500",
         "changeTimeMilliseconds":"1331409333948.00",
         "accessTimeString":"2012-03-10T14:55:33-0500",
         "accessTime":1331409333,
         "acl":false},
        .
        .
        .
        ],
     "status":{"results":7,"message":"","code":"COMPLETE"}
    }
}
```

Example 2: Retrieving metadata for changed objects

Here's a sample metadata query API request that uses a JSON body specified directly in the cURL command line and Python code to retrieve operation records for objects that:

- Are in the finance namespace, which is owned by the europe tenant
- Were modified during 2011

The **start** entry specifies 12:00:00.00 a.m. on January 1, 2011, and the **end** entry specifies 12:00:00.00 a.m. on January 1, 2012.

The response body is XML. The information returned for each operation record that meets the query criteria consists of the object URL, version ID, operation, and change time.

Request with cURL command line

```
curl -k -H "Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d"
    -H "Content-Type: application/json" -H "Accept: application/xml"
    -d '{"operation":{"systemMetadata":{"changeTime":
    {"start":1293840000000,"end":1325376000000},"namespaces":
    {"namespace":["finance.europe"]}}}}'
    "https://europe.hcp.example.com/query?prettyprint"
```

Request in Python using PycURL

```
import pycurl
curl = pycurl.Curl()

# Set the URL, command, and headers
curl.setopt(pycurl.URL, "https://europe.hcp.example.com/" +
    "query?prettyprint")
curl.setopt(pycurl.SSL_VERIFYPEER, 0)
curl.setopt(pycurl.SSL_VERIFYHOST, 0)
curl.setopt(pycurl.POST, 1)
curl.setopt(pycurl.HTTPHEADER,
    ["Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d",
    "Content-Type: application/json", "Accept: application/xml"])

# Set the request body
theFields = '{"operation":{"systemMetadata":{"changeTime": \
  {"start":1293840000000,"end":1325376000000},"namespaces": \
  {"namespace":["finance.europe"]}}}}'
curl.setopt(pycurl.POSTFIELDS, theFields)
```

```
curl.perform()
print curl.getinfo(pycurl.RESPONSE_CODE)
curl.close()
```

Request headers

```
POST /query HTTP/1.1
Host: europe.hcp.example.com
Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d
Content-Type: application/json
Accept: application/xml
Content-Length: 81
```

Response headers

```
HTTP/1.1 200 OK
Server: HCP V7.0.0.16
Transfer-Encoding: chunked
```

Response body

To limit the example size, the XML below shows only two **object** entries in the response body.

```
<?xml version='1.0' encoding='UTF-8'?>
<queryResult xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="/static/xsd/query-result-7.0.xsd">
    <query start="1293840000000" end="1325376000000" />
    <resultSet>
        <object
            version="81787101672577"
            urlName="https://finance.europe.hcp.example.com/rest/
                Presentations/Q2_2011.ppt"
            operation="CREATED"
            changeTimeMilliseconds="1310392057456.00" />
        <object
            version="81787102472129"
            urlName="https://finance.europe.hcp.example.com/rest/
                Presentations/Images/thankYou.jpg"
            operation="CREATED"
            changeTimeMilliseconds="1310392336286.00" />
        .
        .
        .
    </resultSet>
    <status results="11" message="" code="COMPLETE" />
</queryResult>
```

Example 3: Using a paged query to retrieve a large number of records

The Python example below implements a paged query that uses multiple requests to retrieve a large number of operation records in batches of 50 per request. This query retrieves records for all create operations on objects in the `/customers/widgetco/orders` directory in the default namespace and returns basic information for each record.

The query uses a JSON request body and requests results in JSON format.

```
#!/usr/bin/env python
# encoding: utf-8

import pycurl
import StringIO
import time
import json

class OperationBasedQueryTool():
  queryArguments = {'operation': {'count': 1, 'verbose': 'false',
    'objectProperties': 'utf8Name, type, size',
    'systemMetadata': {'changeTime': {},
      'directories': {'directory': []},
      'namespaces': {'namespace': []},
      'transactions': {'transaction': []}}}}

  def __init__(self):
    self.complete = False

  def setConnectionInfo(self, authToken, hostName, urlName):
    """ Set all connection info for subsequent query requests.
    @param authToken: authorization token
    @param hostName: Hostname of the target cluster
    @param urlName: Full URL for the query interface """
    self.curl = pycurl.Curl()
    requestHeaders = {pycurl.HTTPHEADER :["Authorization: HCP
      "authToken, "Accept:application/json", "Content-Type:
      application/json", "Host: admin.%s" % (hostName)]}
    self.curl.setopt(pycurl.FAILONERROR, 1)
    self.curl.setopt(pycurl.HTTPHEADER,
      requestHeaders[pycurl.HTTPHEADER])
    self.curl.setopt(pycurl.URL, urlName)
    for header, value in requestHeaders.iteritems():
      self.curl.setopt(header, value)
    self.curl.setopt(pycurl.CUSTOMREQUEST, 'POST')
    self.curl.setopt(pycurl.SSL_VERIFYPEER, 0)
    self.curl.setopt(pycurl.SSL_VERIFYHOST, 0)
    self.curl.setopt(pycurl.VERBOSE, 0)

  def setQueryParameters(self, count, verbose, directories, namespaces,
```

```
    transactions, objectProperties, startTimeMillis=0,
    endTimeMillis=int(round(time.time() * 1000))):
    """ Set all parameters related to the query.
    @param count: The number of results to return for each query.
    @param verbose: Indication to return all object property values.
      Value is either true or false.
    @param directories: Dictionary containing list of directory paths.
    @param namespaces: Dictionary containing list of namespaces.
    @param transactions: Dictionary containing list of transaction
      types.
    @param objectProperties: String containing comma-separated list of
      object properties to return for each operation record.
    @param startTimeMillis: The starting timestamp in milliseconds of
      the query window.  Default is 0 (zero).
    @param endTimeMillis: The ending timestamp in milliseconds of the
      query window. Default is one minute before time of request. """
    self.queryArguments['operation']['count'] = count
    self.queryArguments['operation']['objectProperties'] =
      objectProperties
    self.queryArguments['operation']['verbose'] = verbose
    self.queryArguments['operation']['systemMetadata']['directories'] =
      directories
    self.queryArguments['operation']['systemMetadata']['namespaces'] =
      namespaces
    self.queryArguments['operation']['systemMetadata']['transactions'] =
      transactions
    self.queryArguments['operation']['systemMetadata']['changeTime']
      ['start'] = startTimeMillis
    self.queryArguments['operation']['systemMetadata']['changeTime']
      ['end'] = endTimeMillis

  def issueQuery(self):
    """ Issue an operation-based query request. """
    self.curl.setopt(pycurl.POSTFIELDS, json.dumps(self.queryArguments))
    cout = StringIO.StringIO()
    self.curl.setopt(pycurl.WRITEFUNCTION, cout.write)
    print("Performing query with the following arguments: %s"
      % json.dumps(self.queryArguments))
    self.curl.perform()
    responseCode = self.curl.getinfo(pycurl.RESPONSE_CODE)
    if responseCode == 200:
      queryResult = eval(cout.getvalue())
      if queryResult['queryResult']['status']['code'] == "COMPLETE":
        self.complete = True
      cout.close()
      return queryResult
    else:
      raise Exception("Error: Expected result code 200, but received %s"
        % responseCode)

  def setLastResult(self, lastResult):
    """ Sets the last result we received as the starting point for the
      next query we issue.
```

```
    @param lastResult: The dictionary containing the last result
      returned by the previous query. """
    self.queryArguments['operation']['lastResult'] = dict()
    self.queryArguments['operation']['lastResult']['urlName'] =
      lastResult['urlName']
    self.queryArguments['operation']['lastResult']
      ['changeTimeMilliseconds'] = lastResult['changeTimeMilliseconds']
    self.queryArguments['operation']['lastResult']['version'] =
      str(lastResult['version'])

  def closeConnection(self):
    """ Cleanup the curl connection after we are finished with it. """
    self.curl.close()

if __name__ == '__main__':
  authToken = "bXl1c2Vy:3f3c6784e97531774380db177774ac8d"
  hostName = "clusterName.com"
  urlName = "https://admin.%s/query" % hostName
  resultsPerQuery = 50
  objectUrls = []
  queryTool = OperationBasedQueryTool()
  queryTool.setConnectionInfo(authToken, hostName, urlName)
  queryTool.setQueryParameters(resultsPerQuery, "false",
    {'directory':['/customers/widgetco/orders']},
    {'namespace':['Default.Default']},
    {'transaction':['create']})
  try:
    while not queryTool.complete:
      queryResults = queryTool.issueQuery()
      for result in queryResults['queryResult']['resultSet']:
        objectUrls.append(result['urlName'])
      resultCount = len(queryResults['queryResult']['resultSet'])
      queryTool.setLastResult(queryResults['queryResult']['resultSet']
        [resultCount-1])
    print("Query completed. Total objects found: %d" % len(objectUrls))
  finally:
    queryTool.closeConnection()
```

Example 4: Checking for replication collisions

Here's a sample metadata query API request that checks whether any namespaces owned by the europe tenant currently contain objects that are flagged as replication collisions. The response to the query does not include operation records for any of those objects, but the status of INCOMPLETE indicates that records for such objects exist.

The query uses an XML request body and requests results in XML format.

Request body in the XML file named ReplicationCollisions.xml

```
<queryRequest>
    <operation>
        <count>0</count>
        <systemMetadata>
            <replicationCollision>true</replicationCollision>
            <transactions>
                <transaction>create</transaction>
            </transactions>
        </systemMetadata>
    </operation>
</queryRequest>
```

Request with cURL command line

```
curl -i -k -H "Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d"
    -H "Content-Type: application/xml" -H "Accept: application/xml"
    -d @ReplicationCollisions.xml
    "https://europe.hcp.example.com/query?prettyprint"
```

Request in Python using PycURL

```
import pycurl
import os
curl = pycurl.Curl()

# Set the URL, command, and headers
curl.setopt(pycurl.URL, "https://europe.hcp.example.com/" +
    "query?prettyprint")
curl.setopt(pycurl.SSL_VERIFYPEER, 0)
curl.setopt(pycurl.SSL_VERIFYHOST, 0)
curl.setopt(pycurl.POST, 1)
curl.setopt(pycurl.HTTPHEADER,
    ["Authorization: HCP bXl1c2Vy:3f3c6784e97531774380db177774ac8d",
    "Content-Type: application/xml", "Accept: application/xml"])

# Set the request body from an XML file
filehandle = open("ReplicationCollisions.xml", 'rb')
curl.setopt(pycurl.UPLOAD, 1)
curl.setopt(pycurl.CUSTOMREQUEST, "POST")
curl.setopt(pycurl.INFILESIZE,
         os.path.getsize("ReplicationCollisions.xml"))
curl.setopt(pycurl.READFUNCTION, filehandle.read)

curl.perform()
print curl.getinfo(pycurl.RESPONSE_CODE)
curl.close()
filehandle.close()
```

Request headers

```
POST /query?prettyprint HTTP/1.1
Host: europe.hcp.example.com
Content-Type: application/xml
Accept: application/xml
Authorization: HCP YWxscm9sZXM=:04EC9F614D89FF5C7126D32ACB448382
Content-Length: 233
```

Response headers

```
HTTP/1.1 200 OK
Server: HCP V7.0.0.16
Transfer-Encoding: chunked
```

XML response body

```
<?xml version='1.0' encoding='UTF-8'?>
<queryResult xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="/static/xsd/query-result-7.0.xsd">
<query
    start="0"
    end="1395694699683" />
<resultSet />
<status
    results="0"
    message=""
    code="INCOMPLETE" />
</queryResult>
```

6

Usage considerations

This chapter contains usage considerations for the metadata query API.

Hostname and IP address considerations for paged queries

For operation-based queries, HCP caches each query for a period of time on the node that receives the request. If you use an IP address in the URL in each request, you access the cached query and avoid having to recreate the query with each request. This can significantly improve the performance of paged queries that return metadata for large numbers of objects.

Some HTTP libraries cache HTTP connections. Programs using these libraries may automatically reconnect to the same node for paged queries. In this case, using a hostname to establish the connection provides the same performance benefit as using an IP address.

For more information on the relative advantages of hostnames and IP addresses, see *Using a Namespace* or *Using the Default Namespace*.

Maximum concurrent queries

HCP nodes can each process a maximum of five concurrent queries. If a query arrives at a node that is currently processing five queries, HCP returns a 503 (Service Unavailable) error code.

In response to a 503 error code, you should try the query again, gradually increasing the delay between each successive attempt.

Query performance with object-based queries

For improved query performance with object-based queries:

- Do not specify a **true** value for the **verbose** request entry.
- Specify the **facets** or **sort** request entries only when necessary.
- In the **objectProperties** entry, specify only **urlName**, **objectPath**, **version**, and **changeTimeMilliseconds**.
- When performing paged queries, specify a value of one hundred or less in the **count** request entry.

Queries based on object names

The metadata query API relies on UTF-8 encoding conventions to find objects by name. If the name of an object is not UTF-8 encoded, queries for the object by name may return unexpected results.

Querying specified namespaces

When you specify namespaces in query requests, responses to object-based queries differ from responses to operation-based queries in these situations:

- If you specify a namespace that doesn't exist:
  - In an object-based query, HCP returns zero results. If you specify both namespaces that exist and namespaces that don't exist, HCP returns results for objects in the namespaces that exist.
  - In an operation-based query, HCP returns a 400 (Bad Request) error.
- If you use a tenant-level user account and specify an HCP namespace for which you do not have search permission:
  - In an object-based query, HCP returns zero results for the namespace.
  - In an operation-based query, HCP returns a 403 (Forbidden) error.
- If you use a system-level user account and specify an HCP namespace owned by a tenant that has not granted administrative access to system-level users:
  - In an object-based query, HCP returns zero results for the namespace.
  - In an operation-based query, HCP returns a 403 (Forbidden) error.

HTTP return code considerations

Applications should account for all possible HTTP return codes for query requests. For a list of possible HTTP return codes for query requests, see "HTTP return codes" on page 73.

Glossary

A

access control list (ACL)

Optional metadata consisting of a set of grants of permissions to perform various operations on an object. Permissions can be granted to individual users or to groups of users.

ACLs are provided by users or applications and are specified as either XML or JSON.

ACL

See access control list (ACL).

Active Directory (AD)

A Microsoft product that, among other features, provides user authentication services.

AD

See Active Directory (AD).

annotation

A discrete unit of custom metadata. Annotations are typically specified in XML format.

authenticated access

A method of access to a namespace wherein the user or application presents credentials to gain access.

C

content property

A content property is a named construct used to extract an element or attribute value from custom metadata that's well-formed XML. Content properties use XPath expressions to identify the metadata of interest.

cryptographic hash value

A system-generated metadata value calculated by a cryptographic hash algorithm from object data. This value is used to verify that the content of an object has not changed.

custom metadata

User-supplied information about an HCP object. Custom metadata is specified as one or more annotations, where each annotation is a discrete unit of information about the object. Users and applications can use custom metadata to understand and repurpose object content.

D

data protection level (DPL)

The number of copies of the data for an object HCP must maintain in the repository. The DPL for an object is determined by the service plan that applies to the namespace containing the object.

default namespace

A namespace that supports only anonymous access through the HTTP protocol. An HCP system can have at most one default namespace. The default namespace is used mostly with applications that existed before release 3.0 of HCP.

default tenant

The tenant that manages the default namespace.

disposition

The automatic deletion of an expired object by HCP.

DNS

See domain name system (DNS).

domain

A group of computers and devices on a network that are administered as a unit.

domain name system (DNS)

A network service that resolves domain names into IP addresses for client access.

DPL

See data protection level (DPL).

E

See HCP S Series Node.

expired object

An object that is no longer under retention.

F

facet

An object property for which a query returns summary information, specifically the values of that property that occur in the result set and the numbers of objects that have each of those values.

G

GID

POSIX group identifier.

H

hash value

See cryptographic hash value.

HCP

See Hitachi Content Platform (HCP).

HCP metadata query API

See metadata query API.

HCP namespace

A namespace that supports user authentication for data access through the HTTP, HS3, and CIFS protocols. HCP namespaces also support access control lists and versioning. An HCP system can have multiple HCP namespaces.

HCP tenant

A tenant created to manage HCP namespaces.

Hitachi Content Platform (HCP)

A distributed storage system designed to support large, growing repositories of fixed-content data. HCP provides a single scalable environment that can be used for archiving, business continuity, content depots, disaster recovery, e-discovery, and other services. With its support for multitenancy, HCP securely segregates data among various constituents in a shared infrastructure. Clients can use a variety of industry-standard protocols and various HCP-specific interfaces to access and manipulate objects in an HCP repository.

hold

A condition that prevents an object from being deleted by any means and from having its metadata modified, regardless of its retention setting, until it is explicitly released.

HTTP

HyperText Transfer Protocol. The protocol used to provide access to namespace content through the metadata query API.

HTTPS

HTTP with SSL security. *See* HTTP *and* SSL.

I

index

An index of the objects in namespaces that is used to support object-based queries. HCP builds this index from object metadata, including custom metadata and ACLs.

index setting

The property of an object that determines whether the metadata query engine indexes the custom metadata associated with the object.

J

JSON

JavaScript Object Notation. A language-independent format for encoding data in the form of name/value pairs.

M

metadata

System-generated and user-supplied information about an object. Metadata is stored as an integral part of the object it describes, thereby making the object self-describing.

metadata query API

A RESTful HTTP interface that lets you search HCP for objects that meet specified metadata-based or operation-based criteria. With this API, you can search not only for objects currently in the repository but also for information about objects that are no longer in the repository.

N

namespace

A logical partition of the objects stored in an HCP system. A namespace consists of a grouping of objects such that the objects in one namespace are not visible in any other namespace. Namespaces are configured independently of each other and, therefore, can have different properties.

O

object

An exact digital representation of data as it existed before it was ingested into HCP, together with the system and custom metadata that describes that data. Objects can also include ACLs that give users and groups permission to perform certain operations on the object.

An object is handled as a single unit by all transactions and internal processes, including shredding, indexing, versioning, and replication.

object-based query

A query that searches for objects based on object metadata. This includes both system metadata and the content of custom metadata and ACLs. The query criteria can also include the object location (that is, the namespace and/or directory that contains the object).

Object-based queries search only for objects that currently exist in the repository. For objects with multiple versions, object-based queries return only the current version.

operation-based query

A query that searches not only for objects currently in the repository but also for information about objects that have been deleted by a user or application, deleted through disposition, purged, or pruned. For namespaces that support versioning, operation-based queries can return both current and old versions of objects.

Criteria for operation-based queries can include object status (for example, created or deleted), change time, index setting, and location (that is, the namespace and/or directory that contains the object).

operation record

A record of a create, delete, purge, prune, or disposition operation. The record identifies the object involved, the type of operation, and the time at which the operation occurred and also contains system metadata for the object. HCP updates the applicable creation record when object metadata changes.

P

permission

One of these:

- In POSIX permissions, the ability granted to the owner, the members of a group, or other users to access an object. A POSIX permission can be read, write, or execute.

- In a tenant-level user account, the granted ability to perform a specific type of operation in a given namespace.

- In an ACL associated with an object, the granted ability to perform a specific type of operation on the object.

policy

One or more settings that influence how transactions and internal processes work on objects in HCP. Such a setting can be a property of an object, such as retention, or a property of a namespace, such as versioning.

POSIX

Portable Operating System Interface for UNIX. A set of standards that define an application programming interface (API) for software designed to run under heterogeneous operating systems.

pruning

See version pruning.

purge

The operation that deletes all versions of an object.

Q

query

A request submitted to HCP to return metadata or operation records for objects that satisfy a specified set of criteria. Also, to submit such a request.

query API

See metadata query API.

R

replication

The process of keeping selected HCP tenants and namespaces and selected default-namespace directories in two HCP systems in sync with each other. This entails copying object creations, deletions, and metadata changes from each system to the other or from one system to the other.

repository

The aggregate of the namespaces defined for an HCP system.

REST

Representational State Transfer. A software architectural style that defines a set of rules (called constraints) for client/server communication. In a REST architecture:

- Resources (where a resource can be any coherent and meaningful concept) must be uniquely addressable.
- Representations of resources (for example, in XML format) are transferred between clients and servers. Each representation communicates the current or intended state of a resource.
- Clients communicate with servers through a uniform interface (that is, a set of methods that resources respond to) such as HTTP.

retention class

A named retention setting. The value of a retention class can be a duration, Deletion Allowed, Deletion Prohibited, or Initial Unspecified.

retention hold

See hold.

retention period

The period of time during which an object stored in HCP cannot be deleted.

retention setting

The property that determines the retention period for an object.

S

shred setting

The property that determines whether an object will be shredded or simply removed when it's deleted from HCP.

shredding

The process of deleting an object and overwriting the locations where all its copies were stored in such a way that none of its data or metadata can be reconstructed. Also called **secure deletion**.

SPNEGO

Simple and Protected GSSAPI Negotiation. A protocol used for client authentication against a remote server.

SSL

Secure Sockets Layer. A key-based Internet protocol for transmitting documents through an encrypted link.

SSL server certificate

A file containing cryptographic keys and signatures. When used with the HTTP protocol, an SSL server certificate helps verify that the web site holding the certificate is authentic. An SSL server certificate also helps protect data sent to or from that site.

system metadata

System-managed properties that describe the content of an object. System metadata includes policies, such as retention and data protection level, that influence how transactions and internal processes affect the object.

T

tenant

An administrative entity created for the purpose of owning and managing namespaces. Tenants typically correspond to customers or business units.

U

UID

POSIX user ID.

Unix

Any UNIX-like operating system (such as UNIX itself or Linux).

user account

A set of credentials that gives a user access to namespace content through the metadata query API.

user authentication

The process of checking that the combination of a specified username and password is valid when a user tries to access a namespace.

versioning

An optional namespace feature that enables the creation and management of multiple versions of an object.

version pruning

The automatic deletion of previous versions of objects that are older than a specified amount of time.

X

XML

Extensible Markup Language. A standard for describing data content using structural tags called elements.

XPath

A language used to formulate expressions that navigate through and select elements and attributes in XML documents.

Index

Symbols

A

B

C

D

E

Q

R

S

T

U

V

W

X

Hitachi Data Systems

Corporate Headquarters
2845 Lafayette Street
Santa Clara, California 95050-2627
U.S.A.
www.hds.com

Regional Contact Information

Americas
+1 408 970 1000
info@hds.com

Europe, Middle East, and Africa
+44 (0) 1753 618000
info.emea@hds.com

Asia Pacific
+852 3189 7900
hds.marketing.apac@hds.com

Hitachi Data Systems

MK-91ARC032-05