HITACHI
Inspire the Next

Hitachi Content Platform

Using the HCP HS3 API

Hitachi Data Systems

MK-92ARC037-00

Contents

Preface xi

Intended audience xi
Product version xii
Syntax notation xii
Related documents xiii
Getting help xv
Comments xv

1 Introduction to Hitachi Content Platform 1

About Hitachi Content Platform 2
    Object-based storage 2
    Buckets and tenants 3
    HCP nodes 3
About the HCP HS3 API 4
Other bucket access methods 5
    Namespace access protocols 5
    Namespace Browser 6
    HCP metadata query API 6
    HCP Search Console 7
    HCP Data Migrator 8
User accounts 9
Data access permissions 10
Examples in this book 12

2 Bucket and object properties 13

Bucket names 14
Object names 14
    Forward slashes in object names 15
    Object naming considerations 16

Retention . . . 17
Custom metadata . . . 18
  Storing custom metadata with HS3 . . . 18
  Retrieving custom metadata with HS3 . . . 20
  Custom metadata usage considerations . . . 20
Bucket owners . . . 21
Object owners . . . 22
Access control lists . . . 22
  ACL permissions . . . 23
  ACL grantees . . . 24
  Canned ACLs . . . 25
  Specifying ACLs . . . 25
    Specifying an ACL with headers . . . 25
    Specifying an ACL in the request body . . . 27
  Removing an ACL . . . 32
Versioning . . . 32
Allocated space . . . 34
Search . . . 34
Data access permission masks . . . 35

3 Access and authentication . . . 37

URLs for access to HCP . . . 38
  Using an IP address in a URL . . . 39
  Using a hosts file . . . 40
  URL considerations . . . 41
AWS authentication . . . 42
  Authorization header . . . 42
  Presigned URLs . . . 43
  Signatures . . . 44
  Anonymous access . . . 44
  Invalid credentials . . . 44
Changing your password . . . 45

4 Requests and responses . . . 47

Requests . . . 48
  Request line . . . 48
  Common request headers . . . 49
Responses . . . 51
  Response status line . . . 51
  Common response headers . . . 52
  Error response body . . . 53
  Error codes . . . 53

5 Working with buckets ... 59
Creating a bucket ... 60
Request line ... 60
Request headers ... 61
Response headers ... 62
HTTP status codes ... 62
Examples ... 64
Example 1: Creating a bucket ... 64
Example 2: Creating a bucket with an ACL ... 64
Listing the buckets you own ... 65
Request line ... 65
Request headers ... 66
Response headers ... 66
Response body ... 67
HTTP status codes ... 68
Example: Listing the buckets you own ... 69
Checking the existence of a bucket ... 70
Request line ... 70
Request headers ... 71
Response headers ... 71
HTTP status codes ... 71
Example: Checking the existence of a bucket ... 72
Adding an ACL to a bucket ... 73
Request line ... 74
Request headers ... 74
Response headers ... 75
HTTP status codes ... 76
Examples ... 77
Example 1: Adding an ACL to a bucket by specifying individual grants ... 77
Example 2: Adding an ACL to a bucket by using an ACL request body ... 78
Retrieving the ACL for a bucket ... 79
Request line ... 79
Request headers ... 80
Response headers ... 80
HTTP status codes ... 81
Example: Retrieving a bucket ACL ... 81
Enabling or disabling versioning for a bucket ... 83
Request body ... 84
Request line ... 85
Request headers ... 85
Response headers ... 85
HTTP status codes ... 86

Example: Enabling versioning for a bucket . . . 87
Checking the versioning status of a bucket . . . 87
Request line. . . . 88
Request headers . . . 88
Response headers . . . 89
HTTP status codes . . . 89
Example: Checking the versioning status of a bucket . . . 90
Listing bucket contents . . . 91
Limiting the bucket listing . . . 92
Limiting a listing of current items . . . 92
Limiting a version listing . . . 94
Request line. . . . 96
Request headers . . . 99
Response headers . . . 99
Response body. . . . 100
Response body for a listing of current items . . . 100
Response body for a version listing . . . 101
Response body elements. . . . 102
HTTP status codes . . . 106
Examples. . . . 107
Example 1: Listing the items in a folder . . . 107
Example 2: Listing items a few at a time . . . 109
Example 3: Listing the versions of an individual object . . . 113
Deleting a bucket . . . 115
Request line. . . . 115
Request headers . . . 115
Response headers . . . 116
HTTP status codes . . . 116
Example: Deleting a bucket . . . 117

6 Working with objects . . . 119

Storing an object . . . 120
Request line. . . . 121
Request headers . . . 122
Response headers . . . 123
HTTP status codes . . . 124
Examples. . . . 125
Example 1: Storing an object . . . 125
Example 2: Storing an object with custom metadata and an ACL . . . 126
Creating a folder . . . 127
Request line. . . . 128
Request headers . . . 128

Response headers 129
HTTP status codes 129
Example: Creating a folder 130
Checking the existence of an object or folder 131
Request line 132
Request headers 133
Response headers 135
HTTP status codes 137
Examples 138
Example 1: Checking the existence of an object 138
Example 2: Retrieving custom metadata for an object version 139
Example 3: Checking whether an object has been modified 140
Example 4: Checking the existence of a folder 141
Adding an ACL to an object 141
Request line 142
Request headers 142
Response headers 144
HTTP status codes 144
Examples 146
Example 1: Adding an ACL to an object by using a canned ACL 146
Example 2: Adding an ACL to an object by using an ACL request body . 147
Retrieving the ACL for an object 148
Request line 148
Request headers 149
Response headers 149
HTTP status codes 150
Example: Retrieving an object ACL 150
Copying an object 152
Request line 154
Request headers 154
Response headers 159
Response body 160
HTTP status codes 161
Examples 162
Example 1: Conditionally copying an object from one bucket to another 163
Example 2: Recovering an old version of an object 164
Example 3: Replacing custom metadata for an existing object 165
Retrieving an object 166
Request line 169
Request headers 170
Response headers 173

HTTP status codes 174
Examples. 176
Example 1: Conditionally retrieving an object 176
Example 2: Retrieving an old version of an object. 176
Example 3: Retrieving part of an object 177
Deleting an object or folder 178
Request line. 178
Request headers 179
Response headers 179
HTTP status codes 179
Examples. 180
Example 1: Deleting an object 181
Example 2: Deleting a folder. 181

7 Usage considerations...183

Hostname and IP address considerations 184
Folder structures 184
Concurrent writes of the same object 185
Failed PUT requests to store objects. 185
Empty objects 186
Deleting objects under repair. 186
Multithreading 186
Persistent connections 187
Connection failure handling 188

A Quick reference 189

B Alternative authentication method 201

URLs with HCP authentication 202
HCP authentication. 203

C Using third-party tools with HS3 205

General setup information for third-party tools 206
s3curl setup. 206
Specifying a access key and secret key. 206
Specifying a tenant. 207

D Sample Java application...209

Assumptions 209

What the application does 209
Required libraries 210
Java application 211

E HCP HS3 XML schema 217

Glossary 233

Index 245

Preface

This book is your guide to using the **Hitachi Content Platform** (**HCP**) HS3 API. HS3 is a RESTful, HTTP-based API that is compatible with Amazon® S3™. Using HS3, you can create, manage, and delete buckets and objects in HCP.

This book introduces the HCP concepts you need to understand in order to use HS3 effectively. It contains instructions and examples for each of the bucket and object operations you can perform with HS3. It also includes a quick reference for these operations and a sample Java® application that uses HS3 to perform a few operations.

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

Intended audience

This book is intended for people who need to know how to use the HS3 API to work with buckets and objects in HCP. The book addresses two audiences:

- People who are writing applications against HCP. These people are assumed to have programming experience.
- People who are accessing HCP though a third-party, S3-compatible tool (such as s3curl). These people are assumed to have experience using the applicable tool.

In either case, the book assumes you have a basic knowledge of HTTP, as specified by RFC 2616.

This book does not assume any prior knowledge of HCP concepts or functionality.

Product version

This book applies to release 6.0.1 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: **`?versionId=`***`version-id`* You enter: ?versionId=87288808614529
`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: **`+0000`**`|`**`GMT`** You enter: +0000 or: GMT
[ ]	Square brackets — Include none, one, or more of the elements between the brackets	This book shows: **`http`**`[`**`s`**`]` You enter: http or: https
( )	Parentheses — Include exactly one of the elements between the parentheses	This book shows: *`folder-name`*`(`**`/`**`|`**`%2F`**`)` You enter: mktg/ or: mktg%2F
...	Ellipsis — Optionally, repeat the preceding parameter as many times as needed	This book shows: **`"`***`value`***`"`**`[`**`, "`***`value`***`"`**`]...` You enter: "6ed7faad1e0661c03ad65a4317d4a94c" or: "6ed7faad1e0661c03ad65a4317d4a94c", "7ad452af1e2f61b33a865c4362be5921" or: "6ed7faad1e0661c03ad65a4317d4a94c", "7ad452af1e2f61b33a865c4362be5921", "74d824cd5076a1361da128ee18e5a42b"

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 copying tenants and namespaces from one HCP system to another 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 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.

- *HCP Metadata Query API Reference* — This book describes the HCP metadata query API. This RESTful HTTP API enables you to query namespaces 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.

- *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.

- *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 Hitachi Content Platform

Hitachi Content Platform (**HCP**) is a robust storage system designed to support large, growing repositories of fixed-content data. HCP stores objects that include both data and metadata that describes the data. Objects exist in buckets, which are logical partitions of the repository.

HCP provides access to the repository through a variety of industry-standard protocols, as well as through various HCP-specific interfaces. One of these interfaces is the HCP HS3 API — a RESTful, HTTP-based API that is compatible with Amazon S3.

This chapter introduces basic HCP concepts and includes information on what you can do with the HCP HS3 API.

About Hitachi Content Platform

Hitachi Content Platform is a combination of hardware and software that provides an object-based data storage environment. An HCP repository stores all types of data, from simple text files to medical images to multigigabyte database images.

HCP provides easy access to the repository for adding, retrieving, and deleting data. HCP uses write-once, read-many (WORM) storage technology and a variety of policies and internal processes to ensure the integrity and availability of the stored data.

Object-based storage

HCP stores **objects** in a repository. Each object permanently associates data HCP receives (for example, a document, an image, or a movie) with information about that data, called **metadata**.

An object encapsulates:

- **Fixed-content data** — An exact digital reproduction of data as it existed before it was stored in HCP. Once it’s in the repository, this fixed-content data cannot be modified.

- **System metadata** — System-managed properties that describe the fixed-content data (for example, its size and creation date). System metadata includes policies, such as retention, that influence how transactions and internal processes affect the object.

- **Custom metadata** — Optional metadata that a user or application provides to further describe the object. Custom metadata is specified as one or more **annotations**, where each annotation is a discrete unit of information about the object.

  You can use custom metadata to create self-describing objects. Users and applications can use this metadata to understand and repurpose object content.

- **Access control list (ACL)** — Optional metadata consisting of a set of grants of permissions to perform various operations on the object. Permissions can be granted to individual users or to groups of users.

  Like custom metadata, ACLs are provided by users or applications.

HCP can store multiple versions of an object, thus providing a history of how the data has changed over time. Each version is an object in its own right, with system metadata and, optionally, custom metadata and an ACL.

Buckets and tenants

An HCP repository is partitioned into buckets. A **bucket** is a logical grouping of objects such that the objects in one bucket are not visible in any other bucket. Buckets are also called **namespaces**.

Buckets provide a mechanism for separating the data stored for different applications, business units, or customers. For example, you could have one bucket for accounts receivable and another for accounts payable.

Buckets also enable operations to work against selected subsets of objects. For example, you could perform a query that targets the accounts receivable and accounts payable buckets but not the employees bucket.

Buckets are owned and managed by administrative entities called **tenants**. A tenant typically corresponds to an organization, such as a company or a division or department within a company.

In addition to being owned by a tenant, each bucket can have an owner that corresponds to an individual HCP user. The owner of a bucket automatically has permission to perform certain operations on that bucket.

HCP nodes

The core hardware for an HCP system consists of servers that are networked together. These servers are called **nodes**.

When you access an HCP system, your point of access is an individual node. To identify the system, however, you can use either the domain name of the system or the IP address of an individual node. When you use the domain name, HCP selects the access node for you. This helps ensure an even distribution of the processing load.

For information on the URLs you can use to access an HCP system, see “URLs for access to HCP” on page 38. For information on when to use an IP address instead of a domain name, see “Hostname and IP address considerations” on page 184.

About the HCP HS3 API

The HCP HS3 API is a RESTful, HTTP-based API that is compatible with Amazon S3. Using this API, you can:

- Create buckets (PUT bucket)
- List the buckets you own (GET service)
- Check the existence of a bucket (HEAD bucket)
- Add ACLs to existing buckets (PUT bucket acl)
- Retrieve ACLs for buckets (GET bucket acl)
- Enable or suspend object versioning for buckets you own (PUT bucket versioning)
- Check the status of object versioning for buckets you own (GET bucket versioning)
- List objects that are in a bucket (GET bucket)
- List versions of objects that are in a bucket (GET bucket versions)
- Delete buckets you own, as long as the buckets don’t have any objects in them (DELETE bucket)
- Store objects in a bucket (PUT object)
- Create folders in a bucket (PUT object/)
- Add custom metadata to existing objects, where the custom metadata is specified as property/value pairs (PUT object copy replace)
- Check the existence of an object or folder (HEAD object)
- Retrieve custom metadata for objects (HEAD object)
- Add ACLs to existing objects (PUT object acl)
- Retrieve ACLs for objects (GET object acl)
- Copy objects (PUT object copy)

- Retrieve objects (GET object)
- Delete objects (DELETE object)

To use the HS3 API to perform the operations listed above, you can write applications that use any standard HTTP client library. HS3 is also compatible with many third-party tools that support Amazon S3. For information on configuring third-party tools for use with HS3, see Appendix C, "Using third-party tools with HS3," on page 205.

Other bucket access methods

HCP allows access to bucket (namespace) content through:

- Several namespace access protocols
- The Namespace Browser
- A metadata query API
- The Search Console
- HCP Data Migrator

Namespace access protocols

Along with the HS3 API, HCP allows access to namespace content through these industry-standard protocols: HTTP (also called the REST API), WebDAV, CIFS, and NFS. With these protocols, you can access namespaces programmatically with applications, interactively with a command-line tool, or through a GUI. You can use these protocols to perform actions such as storing objects in a namespace, viewing and retrieving objects, changing object metadata, and deleting objects.

HCP allows special-purpose access to namespaces through the SMTP protocol. This protocol is used only for storing email.

The namespace access protocols are configured separately for each namespace and are enabled or disabled independently of each other.

When you use the HS3 API to create a namespace (bucket), both the HS3 API and the HTTP protocol are automatically enabled for that namespace. Additionally, both the HTTP and HTTPS ports are open for both protocols (that is, the namespace can be accessed with or without SSL security).

Tenant administrators can enable and disable namespace access protocols for any namespace. This includes enabling HS3 for namespaces created through other HCP interfaces and disabling HS3 for namespaces created using HS3.

Tip: You can ask your tenant administrator to close the HTTP port for the namespaces you create, thereby allowing only secure access to those namespaces.

Objects added to a namespace through any protocol, including HS3, are immediately accessible through any other protocol that's enabled for the namespace.

For information on using namespace access protocols other than the HS3 API, see *Using a Namespace*.

Namespace Browser

The HCP Namespace Browser lets you manage content in and view information about namespaces. With the Namespace Browser, you can:

- List, view, and retrieve objects, including old versions of objects
- View custom metadata and ACLs for objects, including old versions of objects
- Store and delete objects
- Create empty directories
- Display namespace information

For information on using the Namespace Browser, see *Using a Namespace*.

HCP metadata query API

The **HCP metadata query API** lets you search HCP for objects that meet specified criteria. The API supports two types of queries:

- **Object-based queries** search 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). These queries use a robust query language that lets you combine search criteria in multiple ways.

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 queries** search not only for objects currently in the repository but also for information about objects that have been deleted. 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.

The metadata query API returns object metadata only, not object data. The metadata is returned as XML, with each object represented by a separate element, or JSON, with each object represented by a separate name/value pair. For queries that return large numbers of objects, you can use paged requests.

For information on using the metadata query API, see *HCP Metadata Query API Reference*.

HCP Search Console

The **HCP Search Console** is an easy-to-use web application that lets you search for and manage objects based on specified criteria. For example, you can search for objects stored before a certain date or larger than a specified size and then delete them or prevent them from being deleted. Like the metadata query API, the Search Console returns object metadata only, not object data.

By offering a structured environment for performing searches, the Search Console facilitates e-discovery, namespace analysis, and other activities that look at the contents of namespaces. From the Search Console, you can:

- Open objects
- Perform bulk operations on objects
- Export search results in standard file formats for use as input to other applications
- Publish feeds to make search results available to web users

The Search Console works with any of three search facilities:

- The **HCP metadata query engine** — This facility is integrated with HCP and works internally to perform searches and return results to the Search Console. The metadata query engine is also used by the metadata query API.

Note: When working with the metadata query engine, the Search Console is called the **Metadata Query Engine Console**.

- The **HDDS search facility** — This facility interacts with Hitachi Data Discovery Suite (HDDS), which performs searches and returns results to the HCP Search Console. HDDS is a separate product from HCP.
- The **HCP search facility** — Like the metadata query engine, this facility is integrated with HCP and works internally to perform searches and return results to the Search Console.

The Search Console can use only one of these search facilities at any given time. This facility is selected at the HCP system level. If no facility is selected, the HCP system does not support using the Search Console to search namespaces.

Each search facility maintains its own index of objects in each search-enabled namespace and uses this index for fast retrieval of search results. The search facilities automatically update their indexes to account for new and deleted objects and changes to object metadata.

For information on using the Search Console, see *Searching Namespaces*.

Note: Not all namespaces support search. To find out whether a namespace is search enabled, see your tenant administrator.

HCP Data Migrator

HCP Data Migrator (**HCP-DM**) is a high-performance, multithreaded, client-side utility for viewing, copying, and deleting data. With HCP-DM, you can:

- Copy objects, files, and directories between the local file system, HCP namespaces, default namespaces, and earlier HCAP archives
- Delete objects, files, and directories, including performing bulk delete operations

- View the content of objects and files, including the content of old versions of objects
- Purge all versions of an object
- Rename files and directories on the local file system
- View object, file, and directory properties
- Change system metadata for multiple objects in a single operation
- Add, replace, or delete custom metadata for objects
- Add, replace, or delete ACLs for objects
- Create empty directories

HCP-DM has both a graphical user interface (GUI) and a command-line interface (CLI).

For information on using HCP-DM, see *Using HCP Data Migrator*.

User accounts

To use the HS3 API to create and manage buckets, you need a user account that's configured to allow you to take those actions. To work with objects in a bucket, you may or may not need a user account. This depends on how the HS3 API is configured for the bucket.

By default, when you create a bucket, both the HS3 API and the HTTP protocol are configured to require users to have user accounts in order to work with objects in that bucket. You cannot use the HS3 API to change this configuration. However, tenant administrators can change this configuration for the buckets you create.

A user account can be either an account created by a tenant administrator in HCP or, if the tenant is configured to support Active Directory® (AD) authentication, an AD user account that HCP recognizes. (With an AD user account, you cannot create buckets.)

When you use the HS3 API with a user account, you provide credentials that are based on the username and password for your account. HCP checks these credentials to ensure that they are valid. The process of checking credentials is called **user authentication**. If the credentials you supply are valid, you are an **authenticated user**.

When you use HS3 without a user account, you are an **anonymous user**.

For more information on using the HS3 API with or without a user account, see “AWS authentication” on page 42.

Note: If the HS3 API is not working for you, the reason may be either that the tenant is not configured to support the API or that your user account is not configured to allow the operation you’re trying to perform. To resolve the problem, contact your tenant administrator.

Data access permissions

Data access permissions allow you to access bucket content through the various HCP interfaces. You get these permissions either from your user account or from the bucket configuration.

Data access permissions are bucket specific. That is, they are granted separately for individual buckets.

Each data access permission allows you to perform certain operations. However, not all operations allowed by data access permissions apply to every HCP interface. For example, you can view and retrieve ACLs through the HTTP protocol and HS3 API but not through any other namespace access protocol.

Although many of the operations allowed by data access permissions are not supported by the HS3 API, a tenant administrator can give you permission for those operations. You can then perform them through other HCP interfaces that support them.

The data access permissions that you can have for a bucket are:

- **Browse** — Lets you list bucket contents.
- **Read** — Lets you:
  - View and retrieve objects in the bucket, including the system and custom metadata for objects
  - View and retrieve previous versions of objects
  - List annotations for objects
  - Check the existence of objects

Users with read permission also have browse permission.

- **Read ACL** — Lets you view and retrieve bucket and object ACLs.
- **Write** — Lets you:
  - Add objects to the bucket
  - Modify system metadata (except retention hold) for objects in the bucket
  - Add or replace custom metadata for objects in the bucket
- **Write ACL** — Lets you add, replace, and delete bucket and object ACLs.
- **Change owner** — Lets you change the bucket owner and the owners of objects in the bucket.
- **Delete** — Lets you delete objects, custom metadata, and bucket and object ACLs.
- **Purge** — Lets you delete all versions of an object with a single operation. Users with purge permission also have delete permission.
- **Privileged** — Lets you:
  - Delete or purge objects that are under retention, provided that you also have delete or purge permission for the bucket
  - Hold or release objects, provided that you also have write permission for the bucket
- **Search** — Lets you use the HCP metadata query API and the HCP Search Console to query or search the bucket for objects that meet specified criteria. Users with search permission also have read permission.

If you have any data access permissions for a bucket, you can view information about that bucket through the HTTP protocol and Namespace Browser.

For more information on:

- Bucket and object ACLs, see “Access control lists” on page 22

- Object versions, see “Versioning” on page 32
- Object owners, see “Object owners” on page 22
- Object retention and hold, see “Retention” on page 17
- The HCP Search Console, see “HCP Search Console” on page 7

Examples in this book

This book contains instructions and examples for using HS3 to perform the operations listed in “About the HCP HS3 API” on page 4. The examples use a command-line tool called **s3curl**. **s3curl** is freely available open-source software. You can download it from http://aws.amazon.com/code/128.

After downloading **s3curl**, you need to configure it to work with HCP. For information on this, see Appendix C, “Using third-party tools with HS3,” on page 205.

The examples in this book are based on a bucket named finance in which these objects are stored:

AcctgBestPractices.doc *(four versions stored and one deleted)*
acctg/budget_proposals/BudgProp-2013
hum_res/budget_proposals/BudgProp-2013
mktg/budget_proposals/BudgProp-2013<
mktg/campaign_GoGetEm_expenses.xls *(two versions stored)*
mktg/campaign_LiveIt_expenses.xls
quarterly_rpts/Q2_2012.ppt
quarterly_rpts/Q3_2012.ppt
quarterly_rpts/Q4_2012.ppt
sales/budget_proposals/BudgProp-2013
sales_quotas_2013.pdf

2

Bucket and object properties

Buckets and objects have various properties that affect what you can do with them and what happens when you take action on them. Some of these properties, such as versioning for a bucket, are visible through the HS3 API. Others, such as object retention, aren’t visible through HS3 but still affect that API.

This chapter describes bucket and object properties that have an impact on the HS3 API.

Buckets and objects have additional properties that have no impact on the HS3 API and are, therefore, not described in this book. For information on these properties, see *Using a Namespace*.

Bucket names

When you create a bucket, you give it a name. HCP derives the hostname for the bucket from this name. The hostname is used in URLs for access to the bucket.

In English, the name you specify for a bucket must be from one through 63 characters long and can contain only alphanumeric characters and hyphens (-) but cannot start or end with a hyphen. In other languages, because the derived hostname cannot be more than 63 characters long, the name you specify may be limited to fewer than 63 characters.

Bucket names cannot contain special characters other than hyphens and are not case sensitive. White space is not allowed.

The name you give a bucket must be unique for the tenant for which you create the bucket. Different tenants can have buckets with the same name.

You can reuse bucket names that are not currently in use. So, for example, if you delete a bucket, you can give a new bucket the same name as the deleted bucket had.

Tenant administrators can change the name of a bucket any time after you create the bucket. When the name of a bucket changes, the URL for the bucket may change as well.

Object names

When you create an object, you give it a name. Object names:

- Can contain any characters except the NULL character (ASCII 0 (zero)). This includes nonprinting characters such as spaces and line breaks.
- Are case sensitive.
- Can be up to 4,095 bytes long.

.directory-metadata is a reserved name. You cannot create an object with this name.

You cannot directly change the name of an object. However, you can effectively rename an object by making a copy of it with a different name and then deleting the original. For the object with the new name to have the same ACL as the original object, you need to specify the ACL in the copy request.

For information on copying objects, see "Copying an object" on page 152. For information on ACLs, see "Access control lists" on page 22.

Forward slashes in object names

When using the HS3 API, you can view objects as being in a flat space or in a structured space:

- In a flat space, objects are not grouped by any sort of structural elements, such as folders (also called directories), and forward slashes (/) in object names are simply part of those names.
- In a structured space, forward slashes in object names serve as folder separators, and objects can be grouped into folders.

To support the structured view, when you use HS3 to store an object that has forward slashes in its name, HCP also creates folders and subfolders that conform to the pattern of those forward slashes. For example, if you store an object named quarterly_rpts/Q4_2012, HCP not only stores that object but also creates a folder named quarterly_rpts.

HCP does not create duplicate folders. If, after storing the object named quarterly_rpts/Q4_2012, you store an object named quarterly_rpts/Q3_2012, the single quarterly_rpts folder appears to contain two objects: Q4_2012 and Q3_2012.

Folders provide a method for organizing the objects you store in a bucket. By using meaningful names for the portions of object names from which HCP creates folders, you can more easily manage the contents of the bucket.

You can also use HS3 to create folders and subfolders by themselves. For example, you could create a folder named mktg with a subfolder named budget_proposals. If you then store an object named mktg/budget_proposals/BudgProp-2013, that object appears to be in the mktg/budget_proposals folder. Folder names follow the same rules as object names.

When you use HCP interfaces other than HS3 to view or manage objects stored through HS3, HCP always treats forward slashes in object names as folder separators. So, for example, in the Namespace Browser, the object you stored as quarterly_rpts/Q4_2012 shows up as an object named Q4_2012 in the quarterly_rpts folder.

Grouping objects into folders not only lets you more easily manage those objects, it can also enhance HCP performance. For information on how the folder structures you create can affect performance, see “Folder structures” on page 184.

Object naming considerations

The following considerations apply to object names.

Names ending with a forward slash

A forward slash at the end of a name indicates that the item is a folder. So, for example, if you include a forward slash at the end of the object name in a request to store an object, HCP creates an empty folder with that name and does not store the object.

Object names with non-ASCII, nonprintable characters

When you store an object with non-ASCII, nonprintable characters in its name, those characters are percent-encoded in the name displayed back to you.

Regardless of how the name is displayed, the object is stored with its original name, and you can access it either by its original name or by the names with the percent-encoded characters.

Object names and access through the CIFS and NFS protocols

The CIFS and NFS protocols cannot handle object or folder names that are longer than 255 bytes. An object stored through HS3 is inaccessible through CIFS and NFS if:

- The object name is longer than 255 bytes and does not include any forward slashes.
- The object name includes one or more forward slashes and any part of the name is longer than 255 bytes. In this case, a part of an object name is any character string that either precedes the first forward slash, comes between two forward slashes, or follows the last forward slash.

Percent-encoding special characters

With the HS3 API, object names are specified in URLs. Some characters have special meaning in URLs and may be interpreted incorrectly when used for other purposes. To avoid ambiguity, percent-encode the special characters listed in the table below.

Character	Percent-encoded value
Space	%20
Tab	%09
New line	%0A
Carriage return	%0D
+	%2B
%	%25
#	%23
?	%3F
&	%26

Percent-encoded values are not case sensitive.

UTF-8 encoding

These considerations apply to using UTF-8 encoding for object names:

- Some character-set encoding schemes, such as UTF-8, can require more than one byte to encode a single character. As a result, such encoding can invisibly increase the length of an object name, causing it to exceed the HCP limit of 4,095 bytes.

- When searching buckets, HDDS and HCP rely on UTF-8 encoding conventions to find objects by name. If the name of an object is not UTF-8 encoded, searches for the object by name may return unexpected results.

- When the metadata query engine or HCP search facility indexes an object with a name that includes certain characters that cannot be UTF-8 encoded, it percent-encodes those characters. Searches for such objects by name must explicitly include the percent-encoded characters in the name.

Retention

Every object has a retention setting. This setting determines how long the object must remain in the bucket before it can be deleted. This can range from allowing the object to be deleted at any time to preventing the object from ever being deleted. While an object cannot be deleted due to retention, it is said to be **under retention**.

You cannot use the HS3 API to change the retention setting for an object. However, you can use other namespace access protocols to change it.

Default retention setting

Each bucket in HCP has a default retention setting. This is the setting that's applied to objects when they're first stored in the bucket. When you create a bucket, its default retention setting is to allow deletion at any time.

You cannot use the HS3 API to change the default retention setting for a bucket. However, tenant administrators can change this setting for the buckets you create.

Hold

Objects can be placed on hold. An object on hold cannot be deleted by any means regardless of its retention setting. Additionally, its retention setting cannot be changed.

You cannot use the HS3 API to place objects on hold or release them from being on hold. However, you can use other bucket access methods to do this.

Custom metadata

Objects can optionally have custom metadata in the form of one or more annotations. Annotations are a powerful means for understanding and analyzing the contents of buckets. Using the HCP metadata query API or Search Console, you can search for objects based on the content of their annotations.

Storing custom metadata with HS3

With the HS3 API, you use x-amz-meta- request headers to specify custom metadata. You can use these headers when you store or copy an object.

With the x-amz-meta- header, you specify custom metadata as property/value pairs. You append the property name to the header and specify the value of the property as the value of the header. For example, to give an object a department property with a value of Sales&Mktg and a year property with a value of 2013, you would specify these headers:

```
x-amz-meta-department: Sales&Mktg
x-amz-meta-year: 2013
```

HCP stores the custom metadata you specify with HS3 as an annotation named .metapairs. In this annotation, the property/value pairs are stored as well-formed XML in which each property is represented by an element. For example, the XML stored for the headers shown above is:

```
<metapairs version="600">
    <meta-department><![CDATA[Sales&Mktg]]></meta-department>
    <meta-year><![CDATA[2013]]></meta-year>
</metapairs>
```

The root element in the .metapairs annotation is **metapairs**.

For each property/value pair, the name of the corresponding element is the concatenation of *meta-* and the property name, modified if necessary to be a valid XML element name. Valid XML element names can contain alphanumeric characters, periods (.), hyphens (-), underscores (_), and colons (:). When creating element names from property names, HCP changes any other character to an underscore. For example, the property name city/town becomes the element name **city_town**.

For each property/value pair, the property value becomes the value of the corresponding element. This value is enclosed in a CDATA section.

If you specify an x-amz-meta- header with no value, HCP doesn't store an element for the property named in the header. If all the x-amz-meta- headers you specify have no value, HCP doesn't create a .metapairs annotation for the object.

In a request to store or copy an object, you can specify the same x-amz-meta- property multiple times with different values. In the .metapairs annotation XML, these values are stored as comma-separated values for a single element.

Here's an example that shows three occurrences of the same property along with the resulting XML:

```
x-amz-meta-author: P.D. Grey
x-amz-meta-author: Morgan White
x-amz-meta-author: Paris Black

<metapairs>
    <meta-author>
        <![CDATA[P.D. Grey,Morgan White,Paris Black]]>
    </meta-author>
</metapairs>
```

Property names are case sensitive, so names that differ only in case correspond to separate XML elements. For example, these x-amz-meta- headers result in three separate XML elements: x-amz-meta-date_written, x-amz-meta-Date_Written, and x-amz-meta-DATE_WRITTEN.

Retrieving custom metadata with HS3

When you check the existence of or retrieve an object that has a .metapairs annotation containing well-formed XML, the response headers include x-amz-meta- headers with property/value pairs that correspond to the XML elements in the annotation. The property names in these headers are the element names with the initial *meta-* removed.

An element with comma-separated values is returned as a single x-amz-meta- header with comma-separated values. Here's an example that shows an XML element with comma-separated values and the x-amz-meta- header that results from that element:

```
<meta-author>
    <![CDATA[P.D. Grey,Morgan White,Paris Black]]>
</meta-author>

x-amz-meta-author:P.D. Grey,Morgan White,Paris Black
```

Custom metadata usage considerations

These following considerations apply to using custom metadata with the HS3 API.

Property names

When naming properties, you should use names that, when concatenated with *meta-*, result in valid XML element names. That way, the x-amz-meta- headers returned when you retrieve or check the existence of an object match the x-amz-meta- headers you specified when you stored or copied the object.

If HCP has to modify a property name to create a valid element name, the returned x-amz-meta- header won't match the x-amz-meta- header specified when the object was stored or copied. For example, if the specified header is x-amz-meta-city/town, the returned header is x-amz-meta-city_town.

Custom metadata size

When you use HS3 to store or copy an object, you can specify at most two KB of custom metadata. The size of the custom metadata you specify is the sum of the number of bytes in the UTF-8 encoding of each property name and value.

Allowed operations

Whether you can add, replace, or delete custom metadata for objects under retention depends on a bucket setting. When you create a bucket, it's set to allow only the addition of custom metadata for objects under retention. You cannot use the HS3 API to change this setting. However, tenant administrators can change this setting for the buckets you create.

.metapairs annotations with unexpected content

You should not use HCP interfaces other than HS3 to store annotations named .metapairs. However, HCP does not prevent you from doing this. As a result, annotations named .metapairs are not guaranteed to be compatible with HS3.

Here are some ways in which HCP responds to HS3 requests for objects that have .metapairs annotations with unexpected content:

- If the .metapairs annotation doesn't contain valid XML or if the first line in the annotation doesn't begin with the **metapairs** element, HCP returns an x-amz-missing-meta header with a value of **1** (one) and does not return any x-amz-meta- headers.

- If an element name doesn't start with *meta-*, HCP doesn't return an x-amz-meta- header for the element.

- If a *meta-* element has no value, HCP doesn't return an x-amz-meta- header for the element.

- If a *meta-* element has an attribute, HCP ignores the attribute and returns the applicable x-amz-meta- header.
- If the XML contains nested elements and the lowest-level element is a *meta-* element, HCP returns an x-amz-header for that element. It does not return x-amz-headers for any other elements in that nested structure.

Bucket owners

Buckets can have owners that correspond to HCP or AD user accounts. By default, when using an HCP user account, you own the buckets you create through the HS3 API. With an AD user account, you cannot use HS3 to create buckets.

Normally, as the owner of a bucket, you can use HS3 to view and change the versioning status of the bucket and to delete the bucket if it's empty. If you have write ACL and change owner permissions for a bucket you own, you can use HS3 to change the bucket owner to a different user. However, tenant administrators can reconfigure user accounts such that users using those accounts cannot manage the buckets they own.

With HS3, you use an ACL to change the owner of a bucket. For information on ACLs, see "Access control lists" on page 22.

Tenant administrators can take the same actions on a bucket as the bucket owner can. Additionally, tenant administrators can change a bucket to have no owner.

The maximum number of buckets you can own is limited by a tenant-level configuration setting.

Object owners

Like buckets, objects can have owners that correspond to HCP or AD user accounts. By default, if you're an authenticated user, you own the objects you create in a bucket, regardless of whether you own the bucket. As the owner of the object, you have read, read ACL, write, write ACL, and delete permissions for that object.

If you're not an authenticated user, the objects you create have no owner.

If you have write ACL and change owner permissions for a bucket, you can use HS3 to change the owner of any object in that bucket to a different user. To change the owner of an object, you use an ACL. For information on ACLS, see "Access control lists" below.

Users with the change owner data access permission for a bucket can use the HTTP protocol to change the owner of any object in the bucket to a different user. They can also change any object in the bucket to have no owner.

Access control lists

HCP supports access control lists (ACLs) for both buckets and objects. An ACL grants specified users or groups of users permissions to perform specific operations. An ACL can also be used to change the owner of a bucket or object.

A bucket ACL grants permissions to perform operations on a bucket and on all objects in the bucket. For example, an ACL for a bucket could give all users read permission for that bucket. All users in that group would then be able to retrieve all the objects in that bucket.

An object ACL grants permissions to perform operations on an individual object. For example, an ACL for an object could give a specified user write ACL permission for that object. That user would then be able to change the ACL for that object regardless of whether the user had write ACL permission for the bucket that contained the object.

You can add an ACL to a bucket when you create the bucket or in a separate operation. You can add an ACL to an object when you create or copy the object or in a separate operation. When you add an ACL to an existing bucket or object that already has an ACL, the new ACL replaces the old one in its in entirety.

ACLs can be added to buckets and objects through other HCP interfaces. However, regardless of how they are added, they apply to all HCP interfaces that provide access to objects.

An ACL added through the HS3 API can include at most one hundred permission grants. ACLs added through other HCP interfaces can include more than that. If you retrieve an ACL with more than one hundred grants, HCP returns only the first hundred.

Whether objects in a bucket can have ACLs and whether those ACLs are enforced depend on bucket settings. When you use HS3 to create a bucket, the use of ACLs is automatically enabled. This setting cannot be disabled through any HCP interface.

Also when you use HS3 to create a bucket, ACLs are automatically set to be enforced. When enforcing ACLs, HCP honors the permission grants in object ACLs. When ACLs are not enforced, HCP ignores those grants. HCP always honors permission grants in bucket ACLs.

You cannot use the HS3 API to change the ACL enforcement setting. However, tenant administrators can use other HCP interfaces to change this setting.

ACL permissions

Granting a permission in an ACL for a bucket gives the grantee certain data access permissions for that bucket. Granting a permission in an ACL for an individual object gives the user certain data access permissions just for that object.

The table below lists the permissions you can grant in an ACL and shows the data access permissions that correspond to each ACL permission. For more information on data access permissions, see "Data access permissions" on page 10.

ACL permission	Data access permissions
Read	Browse and read
Read ACP	Read ACL
Write	Write and delete
Write ACP	Write ACL
Full control	Browse, read, read ACL, write, write ACL, and delete

By default, a bucket or object owner that corresponds to an HCP user account or an object owner that corresponds to an AD user account has full control over the applicable bucket or object. For a bucket owner that corresponds to an AD user account, the permissions depend on the tenant configuration.

When adding an ACL to a bucket or object, you can grant only the permissions you already have for that bucket or object. For example, suppose you have read, read ACP, and write ACP permissions for an object.

In this case, you can grant read, read ACP, and write ACP permissions for the object to other users, but you cannot grant write permission or full control.

Tenant administrators can change the permissions that users, including the bucket owner, have for a bucket. They cannot change the permissions users have for objects.

ACL grantees

An ACL can grant permissions to individual users or to groups of users. An individual user is represented by either an HCP user account or, for object ACLs only, an AD user account. A group can be either all authenticated users or all users (both authenticated and anonymous).

To specify an HCP user account, you can use either the account username or the account user ID. To specify an AD user account, you can use either the account username followed by an at sign (@) and the AD domain name (for example, sgold@ad-1.example.com) or the security ID (SID) for the account.

To specify the group of all authenticated users, you can use either the name **authenticated** or this URI:

```
http://acs.amazonaws.com/groups/global/AuthenticatedUsers
```

To specify the group of all users, you use either the name **all_users** or this URI:

```
http://acs.amazonaws.com/groups/global/AllUsers
```

The names **authenticated** and **all_users** are case sensitive. In the URIs, AuthenticatedUsers and AllUsers are case sensitive.

For information on the contexts in which you can specify each type of grantee identifier, see “Using individual grant headers” on page 26 and “Specifying ACLs” on page 25.

Canned ACLs

When specifying an ACL for a bucket or object, you can use a **canned ACL** instead of specifying permission grants individually. A canned ACL is a predefined set of grants of permissions.

HCP has these canned ACLs:

- **private** — Grants full control to the bucket or object owner
- **public-read** — Grants full control to the bucket or object owner and read permission to all users
- **public-read-write** — Grants full control to the bucket or object owner and read and write permissions to all users
- **authenticated-read** — Grants full control to the bucket or object owner and read permission to all authenticated users

Specifying ACLs

To specify an ACL in an HS3 request to create a bucket or store or copy an object, you use request headers. To specify an ACL in a separate operation, you can either use request headers or specify the grants in the request body.

You can also use an ACL request body to change the owner of a bucket or object. You cannot do that with request headers.

Specifying an ACL with headers

Using request headers, you can specify either a canned ACL or individual ACL grants of permissions. You cannot specify both a canned ACL and individual grants in the same request.

Using a canned ACL

To specify a canned ACL, you use the x-amz-acl request header. The value of this header can be the name of any one of the canned ACLs listed in "Canned ACLs" on page 25. These names are case sensitive.

Here's a sample x-amz-acl header that specifies the canned ACL named authenticated-read:

```
x-amz-acl: authenticated-read
```

Using individual grant headers

To grant specific permissions to specific users or groups, you use these headers:

```
x-amz-grant-read
x-amz-grant-read-acp
x-amz-grant-write
x-amz-grant-write-acl
x-amz-grant-full-control
```

Each header grants the permission indicated by the header itself. For information on these permissions, see "ACL permissions" on page 23.

The value for any of these headers is a comma-separated list of one or more grantees, in this format:

identifier-type=*grantee-identifier*

The table below lists the identifier types and indicates how you identify the grantee with each type.

Identifier type	Grantee identifier
id	User ID of an HCP user account or, for object ACLs only, SID of an AD user account. To learn the ID or SID for a user account, see your tenant administrator.
emailAddress	One of these: • Username of an HCP user account • For object ACLs only, username of an AD user account followed by an at sign (@) and the AD domain name • **authenticated** • **all_users** When specifying a username, percent-encode non-ASCII characters and reserved special characters such as ampersands (&), commas (,) and equal signs (=). If a username contains spaces, enclose it in quotation marks.*
uri	URI for the group of all authenticated users or the group of all users (for the URIs, see "ACL grantees" on page 24).
*HS3-compatible third-party tools may not be able to handle usernames with non-ASCII characters, special characters, or spaces. When using such tools, identify the user by user ID rather than by username.	

Identifier types are case sensitive.

Here's a sample x-amz-grant-write header that grants write permission to two users who are identified by their HCP user account IDs:

```
x-amz-grant-write: id=53344e3b-00de-4941-962e-827ac143fa84,
    id=53344e3b-00de-494e-962e-827ac143fa84
```

Heres a sample x-amz-grant-read header that grants read permission to all users:

```
x-amz-grant-read: uri=http://acs.amazonaws.com/groups/global/AllUsers
```

If you include the same header multiple times in a single request, HCP uses only the first one.

Specifying an ACL in the request body

An ACL request body can specify one or more permission grants and/or an owner for the bucket or object. If the specified owner is not the current owner, the owner changes to the specified owner (provided that you change owner permission for the bucket).

For the content of an ACL request body, you use XML in this format:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
To specify the current owner or change the owner, include the Owner
element.
  <Owner>
    <ID>user-id</ID>
    <DisplayName>username</DisplayName>
  </Owner>
  <AccessControlList>
Include one Grant element for each combination of grantee and permission.
    <Grant>
      <Grantee identifier-type
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
To identify the grantee, use either the ID and, optionally, DisplayName
elements, the URI element, or the EmailAddress element.
        <ID>user-id</ID>
        <DisplayName>username</DisplayName>
        <URI>group-uri</URI>
        <EmailAddress>username</EmailAddress>
      </Grantee>
      <Permission>permission</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

The table below describes XML elements in an ACL request body. The elements are listed in alphabetical order.

Element	Description
AccessControlList	Child of the **AccessControlPolicy** element and container for zero or more grants of permissions to individual users or groups. Each grant is represented by a **Grant** element. The **AccessControlList** element is required in an ACL request body.
AccessControlPolicy	Root element. This must be the first element in the ACL request body. The **AcessControlPolicy** element must include this XML namespace specification: xmlns="http://s3.amazonaws.com/doc/2006-03-01/" The **AcessControlPolicy** element is a container for the **Owner** and **AccessControlList** elements, which can occur in either order.
DisplayName	Child of the **Owner** element or of the **Grantee** element when the identifier type is CanonicalUser. The value of the **DisplayName** element can be: • Username of an HCP user account • Username of an AD user account followed by an at sign (@) and the AD domain name • **authenticated** • **all_users** The **DisplayName** element is optional and ignored. The **ID** and **DisplayName** elements can occur in either order.
EmailAddress	Child of the **Owner** element or of the **Grantee** element when the identifier type is AmazonCustomerByEmail. The value of the **DisplayName** element can be: • Username of an HCP user account • For object ACLs only, username of an AD user account followed by an at sign (@) and the AD domain name

(Continued)

Element	Description
Grant	Child of the **AccessControlList** element and container for the **Grantee** and **Permission** elements, which can occur in either order. Each occurrence of the **Grant** element grants one permission to one grantee.
Grantee	Child of the **Grant** element and container for the grantee identifier. The **Grantee** element must include this XML namespace specification: `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"` The **Grantee** element must also include one of these specifications of identifier type to indicate how the grantee is identified: • `xsi:type="CanonicalUser"` The grantee is identified by the **ID** and, optionally, the **DisplayName** element. If present, the **DisplayName** element is ignored. • `xsi:type="Group"` The grantee is identified by the **URI** element. • `xsi:type="AmazonCustomerByEmail"` The grantee is identified by the **EmailAddress** element.
ID	Child of the **Owner** element or of the **Grantee** element when the identifier type is CanonicalUser. The value of the **ID** element can be the user ID of an HCP user account or, for object ACLs only, the SID of an AD user account. The **ID** element is required in the context of the **Owner** element and in the context of the **Grantee** element when the identifier type in the **Grantee** element is CanonicalUser. To learn the ID or SID for a user account, see your tenant administrator.

(Continued)

Element	Description
Owner	Child of the **AccessControlPolicy** element and container for the owner identifier. The owner is identified by the **ID** and, optionally, **DisplayName** elements. The **Owner** element is optional in an ACL request body. If you omit it, the bucket or object owner does not change.
Permission	Child of the **Grant** element. Valid values for the **Permission** element are: • READ • READ_ACP • WRITE • WRITE_ACP • FULL_CONTROL These values are case sensitive. For more information on these values, see "ACL permissions" on page 23.
URI	Child of the **Owner** element or of the **Grantee** element when the identifier type is Group. Valid values for the URI element are the URI for the group of all authenticated users and the URI for the group of all users. For these URIs, see "ACL grantees" on page 24.

Here’s a sample ACL that sets the owner to the user named lgreen and grants read permission to all users and write permission to the user named pdgrey:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Owner>
        <ID>53344e3b-00de-494b-962e-827ac143fa84</ID>
        <DisplayName>lgreen</DisplayName>
    </Owner>
    <AccessControlList>
        <Grant>
            <Grantee xsi:type="Group"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
            </Grantee>
            <Permission>READ</Permission>
        </Grant>
        <Grant>
            <Grantee xsi:type="AmazonCustomerByEmail"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <EmailAddress>pdgrey</EmailAddress>
            </Grantee>
            <Permission>WRITE</Permission>
        </Grant>
    </AccessControlList>
</AccessControlPolicy>
```

Removing an ACL

You can remove an ACL from a bucket or object by removing all grants from it. To do this by using headers, use the x-amz-acl header with the canned ACL named private, like this:

```
x-amz-acl:private
```

To remove an ACL by using an ACL request body, omit all grants from the request the body, like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <AccessControlList>
    </AccessControlList>
</AccessControlPolicy>
```

Removing an ACL does not remove full control from the bucket or object owner.

Versioning

Versioning is an optional bucket feature that enables the creation and management of multiple versions of an object. With versioning enabled, when you store an object with the same name as an existing object, a new version of the object is created. If versioning is not enabled, you cannot store an object that has the same name as an existing object.

HCP does not create new versions of objects that are under retention or on hold. An attempt to store a new version of such an object results in an error.

You can use the HS3 API to enable, disable, and view the status of versioning for buckets you own. Tenant administrators can also take these actions on buckets you own. Whether versioning is initially enabled when you create a bucket is determined by a tenant-level configuration setting.

Note: Tenants can be configured not to allow versioning at all. If the tenant is configured this way, you cannot enable versioning for your buckets.

You cannot enable versioning for a bucket while the WebDAV, CIFS, NFS, or SMTP protocol or appendable objects are enabled for that bucket. You can disable versioning for a bucket at any time.

Disabling versioning does not cause old versions of objects to be deleted. However, you cannot store new versions while versioning is disabled. A request to retrieve an old version of an object while versioning is disabled returns the current version.

Each version of an object is an object in its own right. It has system metadata and can have custom metadata and an ACL. However, you can change metadata only on the current version of an object. Changing metadata, other than the owner or ACL, has no effect on old versions of the object. Changes to the object owner or ACL apply to all versions of the object.

All objects, including those created while versioning is not enabled, have version IDs. Version IDs are integers. Each time a new version of an existing object is created, that new version is assigned an ID that is greater than the ID of the previous version of the object. The IDs for multiple versions of an object are not necessarily consecutive numbers.

Version IDs are unique for the versions of a given object but are not necessarily unique across the versions of all objects.

HCP also assigns version IDs to folders. However, folders cannot have multiple versions.

When you store an object while versioning is enabled, HCP returns the version ID of the object in the x-amz-version-id response header. When you store an object while versioning is disabled, the response headers do not include x-amz-version-id.

When you delete an object while versioning is enabled, HCP:

- Retains a copy of the deleted object as an old version.
- Creates a special version of the object, called a **deleted version**, as the current version. This version serves as an indicator that the object has been deleted. A deleted version of an object has a version ID but does not have any data or metadata. The version ID of the deleted version is different from the version ID of the object you deleted.

If you delete an object with multiple versions while versioning is disabled, HCP, changes the current version of the object to a deleted version and does not change the version ID.

You can delete only the current version of an object (provided that it's not under retention or on hold). You cannot delete an old version or a deleted version.

If versioning is enabled for a namespace, **pruning** may also be enabled. Pruning is the automatic deletion of old versions of objects after a specific length of time since their creation. HCP does not prune old versions of objects that are on hold.

Whether pruning is initially enabled when you create a bucket is determined by a tenant-level configuration setting. You cannot use HS3 to change the pruning setting for a bucket. However, tenant administrators can use other HCP interfaces to do this.

Allocated space

When you create a bucket, the bucket is allocated a certain amount of space. This is the amount of space available for storing objects in the bucket.

The initial amount of space allocated to a bucket is determined by a tenant-level configuration setting. You cannot use HS3 to change the amount of space allocated to a bucket. However, tenant administrators can use other HCP interfaces to do this.

Search

Buckets can be enabled for search. If a bucket is search enabled, users with the search data access permission can use the HCP metadata query API or Search Console to search for objects in the bucket. If a bucket is not search enabled, users cannot search the bucket.

Whether search is initially enabled when you create a bucket is determined by a tenant-level configuration setting. You cannot use HS3 to enable or disable search for a bucket. However, tenant administrators can use other HCP interfaces to take these actions.

For information on the metadata query API, see “HCP metadata query API” on page 6. For information on the Search Console, see “HCP Search Console” on page 7.

Data access permission masks

A **data access permission mask** determines which operations on objects are allowed in a bucket. If the permission mask does not include the permission to perform a particular operation, you cannot perform that operation, regardless of your data access permissions for the bucket or target object.

Data access permission masks are set at the system, tenant, and bucket level. The effective permission mask for a bucket allows only the operations that are allowed at all three levels.

For example, for you to be able to delete an object in a bucket:

- The system-level permission mask must include the delete permission
- The tenant-level permission mask must include the delete permissions
- The permission mask for the bucket must include the delete permission
- Either of these must be true:
  - Your data access permissions for the bucket include delete.
  - You have delete permission for the target object either because you are the object owner or because the object has an ACL that grants you delete permission.

When you create a bucket, its data access permission mask allows all operations. Tenant administrators can change the data permission mask for the buckets you create. You cannot use the HS3 API to change the permission mask for a bucket.

Tenant administrators can also change the tenant-level permission mask, and HCP system administrators can change the system-level permission mask. Changes to the permission mask at any level may affect which operations you can perform with the HS3 API.

3

Access and authentication

HS3 is an HTTP-based API. This means that you use URLs to identify tenants, buckets, and objects. Every HS3 request for access to HCP must include the URL for the service point, bucket, or object that's the target of the request.

With every HS3 request, you also need to either provide credentials for the user account you're using or request anonymous access. If HCP determines that credentials you provide correspond to a valid user account and that the user account has the applicable permissions for the requested operation, you become an authenticated user.

HCP supports two methods of user authentication for HS3: Amazon Web Services™ (AWS™) and HCP. With AWS authentication, HCP follows the Amazon S3 method of authenticating users. With HCP authentication, HCP uses its own standard method. If you're using HS3 to develop S3-compatible applications, you should use AWS authentication.

This chapter describes the URLs you can use in HS3 requests when using AWS authentication. It also explains how to provide credentials or request anonymous access with that authentication method.

For information on HCP authentication and the URLs you can use with that authentication method, see Appendix B, "Alternative authentication method," on page 201.

Notes:

- All the examples of HS3 requests in this book assume AWS authentication.
- To use a recognized Active Directory user account for access to HCP through the HS3 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.

URLs for access to HCP

With the HS3 API, the **service point** (also called the **endpoint**) is a tenant. Every request you make to access HCP using HS3 is made within the context of a tenant.

The URL in a request identifies the target of the request. This target can be the tenant, a bucket within the tenant, or an object within a bucket within the tenant. The format you use for a URL depends on the target of the request and the authentication method being used.

An HCP system can have multiple tenants and, therefore, multiple service points. Each tenant has its own default configuration settings for new buckets. Buckets, therefore, can have different characteristics depending on the tenants within which you create them.

Each tenant also has its own set of user accounts. For access that requires you to be an authenticated user, you need to use a user account that's defined for the applicable tenant.

Targeting a tenant (the service point)

In an HS3 request that uses AWS authentication, if the target is a tenant, you use a URL in this format:

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

Here's an example in which the tenant name is europe:

https://europe.hcp.example.com

Targeting a bucket

In an HS3 request that uses AWS authentication, if the target is a bucket, you use a URL in either of these formats:

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

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

In the first format above, the bucket name is part of the hostname. In the second format, the bucket name follows the hostname.

Here's are examples in which the tenant name is europe and the bucket name is finance:

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

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

Targeting an object

In an HS3 request that uses AWS authentication, if the target is an object, you use a URL in either of these formats:

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

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

In the first format above, the bucket name is part of the hostname. In the second format, the bucket name follows the hostname.

Here's are examples in which the tenant name is europe, the bucket name is finance, and the object name is Q4_2012.ppt:

```
https://finance.europe.hcp.example.com/Q4_2012.ppt
```

```
https://europe.hcp.example.com/finance/Q4_2012.ppt
```

Using SSL security

When you create a bucket or list the buckets you own, HCP does not require the use of SSL security with the HS3 request; that is, you can start the URL in the request with either HTTP or HTTPS. Whether the use of SSL security with other HS3 requests is required, optional, or not supported depends on the configuration of the HS3 API for the target bucket.

By default, the use of SSL security is optional for requests for operations other than creating a bucket or listing the buckets you own. You cannot use HS3 to change this configuration. However, tenant administrators can use other HCP interfaces to reconfigure the HS3 API to require or not support the use of SSL security.

Using an IP address in a URL

Normally, you let HCP choose the node on which to process an HS3 API request. You can, however, use an IP address in the request 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 an HCP node.

When you use an IP address in a URL, you also need to specify the fully qualified hostname in an HTTP Host request header.

Here's an example of an s3curl command that uses a URL with an IP address and a Host header that identifies the finance bucket:

```
./s3curl --id=lgreen --put=Q4_2012.ppt -- -k
    -H "Host:finance.europe.hcp.exmple.com"
    "https://192.168.210.16/quarterly_rpts/Q4_2012.ppt"
```

Here's a command that does the same thing as the command above but uses the format in which the bucket name follows the URL:

```
./s3curl  --id=lgreen  --put=Q4_2012.ppt  --  -k
    -H  "Host:europe.hcp.exmple.com"
    "https://192.168.210.16/finance/quarterly_rpts/Q4_2012.ppt"
```

For information on HCP nodes, see "HCP nodes" on page 3. For information on when to use an IP address instead of a hostname, see "Hostname and IP address considerations" on page 184.

For the IP addresses of the HCP nodes, contact your tenant administrator.

Using a hosts file

Typically, the HCP system uses DNS for system addressing. If this is not the case, to access the system, you need to provide mappings of the hostname of each target tenant or bucket to one or more node IP addresses.

You specify hostname mappings in the hosts file on the client. The location of this 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/hosts

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.
- 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.

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 through the HS3 API.

Sample hosts file

Here's a sample hosts file that contains mappings at the tenant and bucket levels:

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

# bucket-level mappings
192.168.210.16      finance.europe.hcp.example.com  hr.europe.hcp.example.com
192.168.210.17      finance.europe.hcp.example.com  hr.europe.hcp.example.com
192.168.210.18      finance.europe.hcp.example.com  hr.europe.hcp.example.com
192.168.210.19      finance.europe.hcp.example.com  hr.europe.hcp.example.com
```

URL considerations

The following considerations apply to URLs in HS3 API requests.

Case sensitivity in URLs

In the URLs you specify in HS3 requests:

- HTTP and HTTPS are not case sensitive.
- Hostnames are not case sensitive.
- Bucket names are not case sensitive.
- Unlike the items above, object names are case sensitive.

Quotation marks with URLs in command lines

When using a command-line tool to access HCP through the HS3 API, you work in a Unix, Mac OS® X, or Windows shell. Some characters in the commands you enter may have special meaning to the shell. For example, the ampersand (&) used in URLs to join multiple query parameters also often indicates that a process should be put in the background.

To avoid the possibility of the Unix, Mac OS X, or Windows shell misinterpreting special characters in a URL, always enclose the entire URL, including any query parameters, in double quotation marks.

Disabling SSL certificate verification

If you're using HTTPS in your request URLs, check with your tenant administrator as to whether you need to disable SSL certificate verification. You would need to do this if the SSL server certificate presented by the HCP system is not signed by a trusted certificate authority.

With **s3curl**, you disable SSL certificate verification by including the **-k** or **--insecure** option in the request command line.

AWS authentication

To use the HS3 API as an authenticated user, you need to provide credentials that are based on the username and password for your user account. To provide credentials, you typically use the HTTP Authorization request header.

HCP also supports presigned URLs. A presigned URL uses query parameters to provide credentials. Presigned URLs allow you to temporarily share objects with other users without the need to grant those users permission to access your buckets or objects. Presigned URLs are compatible only with the AWS method of authentication.

If the HS3 API is configured to require user authentication, you need to provide credentials with every request. If HS3 is configured to allow anonymous access, you need to either provide credentials or request anonymous access with each request.

Authorization header

To provide credentials for AWS authentication using the Authorization header, you use this format:

```
Authorization: AWS access-key:signature
```

In this format:

- *access-key* is the Base64-encoded username for your user account.
- *signature* is a value calculated using your **secret key** and specific elements of the HS3 request, including the date and time of the request. Your secret key is the MD5-hashed password for your user account.

  Because the signature for an HS3 request is based on the request contents, it differs for different requests.

  For more information on signatures, see "Signatures" on page 44. For information on changing the password for your user account, see "Changing your password" on page 45.

Here's an example of an Authorization header for AWS authentication:

```
Authorization: AWS bGdyZWVu:vQ/rj3y0AUjWsht9M5aQw4+D0dA=
```

Presigned URLs

To provide credentials for AWS authentication using a presigned URL, you append these query parameters to the request URL:

AWSAccessKeyId=*access-key*
Expires=*expiration-time*
Signature=*signature*

In these parameters:

- *access-key* is the Base64-encoded username for your user account.
- *expiration-time* is the time the request expires, in seconds since January 1, 1970, at 00:00:00 UTC.
- *signature* is a value calculated using your secret key and specific elements of the HS3 request, including the date and time the request expires.

If any part of a presigned URL is changed, the URL becomes invalid.

Here's an example of a presigned URL:

```
https://finance.europe.hcp.example.com/quarterly_rpts/Q4_2012.ppt
    ?AWSAccessKeyId=bGdyZWVu&Expires=1363867332
    &Signature=eY9lpX2KXRuW%2FrzLYU%2F5jXfjHj7%3D
```

Signatures

Third-party tools that are compatible with HS3 typically calculate request signatures automatically. Some of these tools can also be used to generate presigned URLs.

If you're writing your own application, you can use an AWS SDK to calculate request signatures. For information on these SDKs see:

http://docs.aws.amazon.com/AmazonS3/latest/dev/AuthUsingAcctOrUserCredentials.html

For detailed information on calculating signatures for AWS authentication, see:

http://docs.aws.amazon.com/AmazonS3/latest/dev/RESTAuthentication.html

Anonymous access

To request anonymous access to a bucket or object, you use the Authorization request header. However, instead of providing credentials for a user account, you specify **all_users** in place of the access key and leave the signature blank, like this:

```
Authorization:  AWS  all_users:
```

The colon after **all_users** is required.

Invalid credentials

Regardless of how HS3 is configured, if you provide invalid credentials, HCP returns an InvalidAccessKeyId or SignatureDoesNotMatch error code and does not perform the requested operation:

- InvalidAccessKeyId indicates that the access key you provided does not correspond to a valid username.

- SignatureDoesNotMatch indicates that the signature does not match the content of the request. This can happen if the secret key used in the signature calculation does not correspond to the password for the account with the applicable username but can also be due to other errors.

Changing your password

If you have a user account that's defined in HCP, the password for the account is initially set by a tenant administrator. You can change this password at any time, as long as you have browse or read permission for at least one bucket. For security purposes, you should change your password as soon as possible after you receive your account information.

Your password is used to calculate your secret key for use with the HS3 protocol. When you change your password, your secret key also changes.

You can use a few different HCP interfaces to change your password, as long as you have access to those interfaces. When you change your password in one interface, it changes for all HCP interfaces to which you have access. The instructions below are for using the Namespace Browser to change your password.

To change your password:

1. Open a web browser window.

2. In the address field, enter the URL for a bucket for which the HS3 API requires user authentication and for which you have browse or read permission, in this format:

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

   For example:

   https://finance.europe.hcp.example.com/browser

   Either of these happens:

   – If the tenant doesn't support AD authentication of if the tenant supports AD authentication but the namespace doesn't support AD single sign-on, the Namespace Browser login page appears.

   – If the tenant supports AD authentication and the HS3 API is configured to support AD single sign-on, a message appears indicating that single sign-on was not possible.

     In this case, you need to click on the **Console login page** link in the message to display the Namespace Browser login page.

3. On the Namespace Browser login page:

   – In the **Username** field, type your username.

- In the **Password** field, type your case-sensitive password.
- In the **Domain** field, if present, select the domain name of the HCP system.

4. Click on the **Log In** button.
5. In the top right corner of the Namespace Browser window, click on the **Password** link.
6. On the **Change Password** page:
   - In the **Existing Password** field, type your current password.
   - In the **New Password** field, type your new password. Passwords must be from one through 64 characters long, are case sensitive, and can contain any valid UTF-8 characters, including white space.

     To be valid, a password must include at least one character from two of these three groups: alphabetic, numeric, and other.

     The minimum length for passwords is tenant specific. Typically, it's six or eight characters.

     When changing your password, you cannot reuse your current password.
   - In the **Confirm New Password** field, type your new password again.
7. Click on the **Update Password** button.

4

Requests and responses

With the HS3 API, you submit requests for operations to HCP, and HCP returns responses that indicate the outcome of the request and include any requested information or content.

This chapter contains general information about HS3 requests and HCP responses to those requests. For information about specific requests, see Chapter 5, “Working with buckets,” on page 59 and Chapter 6, “Working with objects,” on page 119.

Requests

HS3 is an HTTP-based API. As is standard for HTTP, the structure for an HS3 request consists of a request line, request headers that qualify the request, and, if applicable, a request body.

Request line

The request line for an HS3 request specifies the operation to be performed and the version of HTTP on which the HS3 API is based. Additionally:

- If the target of the request is a tenant or if the target is a bucket and the hostname includes the bucket name, the request line includes a forward slash by itself.

  For example, here are the request line and HOST header for a request to create a bucket named finance where the bucket name is included in the hostname:

  ```
  PUT  /  HTTP/1.1
  HOST:  finance.europe.hcp.example.com
  ```

- If the target of the request is a bucket and the hostname does not include the bucket name, the request line includes the a forward slash followed by the bucket name.

  For example, here are the request line and HOST header for a request to create a bucket named finance where the bucket name is not included in the hostname:

  ```
  PUT  /finance  HTTP/1.1
  HOST:  europe.hcp.example.com
  ```

- If the target of the operation is an object, the request line includes a forward slash followed by the object name either by itself or following the bucket name, depending on whether the hostnames includes the bucket name.

  For example, here are the request line and HOST header for a request to create an object named quarterly_rpts/Q4_2012 where the bucket name is included in the hostname:

  ```
  PUT  /quarterly_rpts/Q4_2012  HTTP/1.1
  HOST:  finance.europe.hcp.example.com
  ```

Here are the request line and HOST header for a request to create an object named quarterly_rpts/Q4_2012 in a bucket named finance where the bucket name is not included in the hostname:

```
PUT  /finance/quarterly_rpts/Q4_2012  HTTP/1.1
HOST:  europe.hcp.example.com
```

- If the request uses any query parameters, those parameters are appended to the last one of the items listed above.

  For example, here are the request line and HOST header for a request to add an ACL to an object named quarterly_rpts/Q4_2012 where the bucket name is included in the hostname:

```
PUT  /quarterly_rpts/Q4_2012?acl  HTTP/1.1
HOST:  finance.europe.hcp.example.com
```

  If the request includes invalid query parameters, those parameters are ignored.

Common request headers

Some request headers are common to all HS3 requests. The table below describes those headers. Request headers that are specific to certain requests are described in the sections for those requests in Chapter 5, “Working with buckets,” on page 59 and Chapter 6, “Working with objects,” on page 119.

Request header names are not case sensitive. Depending on the header, the header values may or may not be case sensitive.

Request header	Description
Authorization	Specifies user credentials or requests anonymous access. For valid values for this header, see “AWS authentication” on page 42.

(Continued)

Request header	Description
Date	Specifies the date and time at which the request is being made according to the requester. Normally, this is the current date and time. The date and time must always be specified using Greenwich Mean Time (GMT). To specify the date and time, use this format: `DDD, dd MMM yyyy HH:mm:ss` **`+0000|GMT`** In this format: • *DDD* is the three-letter abbreviation for the day of the week, with an uppercase first letter (for example, Mon). • *dd* is the two-digit day of the month. • *MMM* is the three-letter abbreviation for the month, with an uppercase first letter (for example, Feb). • *yyyy* is the four-digit year. • *HH* is the hour on a 24-hour clock. • *mm* is the number of minutes into the hour. • *ss* is the number of seconds into the minute. For example: Thu, 07 Mar 2013 14:27:05 +0000 All HS3 requests must include either a Date header or an x-amz-date header. If a request includes both headers, HCP uses the date and time in the x-amz-date header.
Host	Specifies the hostname for the request. The host name identifies either a tenant or a bucket. For a tenant, use this format: *tenant-name*.*hcp-domain name* For a bucket, use this format: *bucket-name*.*tenant-name*.*hcp-domain name*

(Continued)

Request header	Description
x-amz-date	Specifies the date and time at which the request is being made according to the requester. Normally, this is the current date and time. For the valid values for this header, see the description of the Date header above.

Note: HCP also accepts several standard HTTP request headers that are not described in this book. Among others, these include Cache-Control, Connection, Content-Disposition, Content-Encoding, Content-Language, and Pragma. For more information on HTTP request headers, see http://www.w3.org/Protocols/rfc2616/rfc2616.html.

Responses

The response to an HS3 request consists of a response status line, response headers that provide information about the execution and outcome of the request, and, if applicable, a response body. The response to an unsuccessful request (except for **HEAD** requests) includes an error response body.

Response status line

The status line returned in response to an HS3 request specifies the version of HTTP on which the HS3 API is based and an HTTP status code that indicates the outcome of the request. HTTP status codes in the 2*xx* range indicate that the requested operation was successful. HTTP status codes in the 3*xx*, 4*xx*, and 5*xx* ranges, typically indicate that an error occurred and HCP did not perform the requested operation.

Here's an example of a response status line that indicates that the requested operation was successful:

```
HTTP/1.1 200 OK
```

Here's an example of a response status line for an unsuccessful operation:

```
HTTP/1.1 409 Conflict
```

For the correspondence between HTTP status codes and errors that can be reported in an error request body, see "Error codes" on page 53. For the status codes that are specific to certain requests, see the individual request descriptions in Chapter 5, "Working with buckets," on page 59 and Chapter 6, "Working with objects," on page 119.

Common response headers

Some response headers are common to all HS3 requests. The table below describes those headers. Response headers that are specific to certain requests are described in the sections for those requests in Chapter 5, "Working with buckets," on page 59 and Chapter 6, "Working with objects," on page 119.

Response header	Description
Content-Length	The size, in bytes, of the response body if HCP can determine the size before formulating the response. If the response does not include a response body, the value of the Content-Length header is **0** (zero).
Content-Type	The Internet media type of the response body if HCP can determine the Internet media type. If HCP cannot determine the Internet media type, the value of this header is **application/octet-stream**. Because HCP returns error information in a response body, the response to any request can include a Content-Type header.
Date	The date and time at which HCP responded to the request, in Greenwich Mean Time (GMT). The date and time are returned in this format: *DDD dd MMM yyyy HH*:*mm*:*ss* GMT For example: Thu, 14 Mar 2013 14:27:05 GMT
Server	The system that responded to the request. The value of this header is always HCP followed by the HCP version number (for example, HCP V6.0.1.117).
Transfer-Encoding	Always **chunked**. This header is returned if the response includes a response body but HCP cannot determine the size of the response body before formulating the response. Because HCP returns error information in a response body, the response to any request can include a Transfer-Encoding header.

Note: HCP also returns several standard HTTP response headers that are not described in this book. Among others, these include Cache-Control, Connection, Content-Disposition, Content-Encoding, Content-Language, Expires, and Pragma. For more information on HTTP response headers, see http://www.w3.org/Protocols/rfc2616/rfc2616.html.

Error response body

When an HS3 requests results in an error, HCP returns information about the error in an error request body. An error request body contains XML in this format:

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
    <Code>error-code</Code>
    <Message>error-message</Message>
    <RequestId>request-id</RequestId>
    <HostId>host-id</HostId>
    <BucketName>specified-bucket-name</BucketName>
    <Key>object-name</Key>
</Error>
```

The table below describes XML elements in an error response body.

Element	Description
Error	Root element.
Code	The error code. For information on error codes, see "Error codes" below.
Message	Text that provides more information about the error.
RequestId	The HCP-internal ID assigned to the request.
HostId	The HCP-internal ID of the host responding to the request.
BucketName	The specified bucket name. This element is included in the response body only when the error code is NoSuchBucket.
Key	The specified object name. This element is included in the response body only when the error code is NoSuchKey.

Error codes

Every error response body contains an error code and a message that provides more information about the error. Error codes are returned in addition to HTTP status codes. A single HTTP status code can correspond to multiple error codes.

The table below describes the error codes that can be returned in response to HS3 requests. For information on the HTTP status codes returned in response to specific requests, see the individual request descriptions in Chapter 5, "Working with buckets," on page 59 and Chapter 6, "Working with objects," on page 119.

Error code	Description	HTTP status code
AccessDenied	One of: • You do not have permission to perform the requested operation. • The user account identified by the access key provided in the request is disabled. To have the account enabled, contact your tenant administrator. • The target bucket does not currently support the requested operation. • The tenant does not currently support use of the HS3 API. • You are using a presigned URL, and the time period during which the URL was valid has expired.	403 Forbidden
BadDigest	One of: • The value of the Content-MD5 header does not match the Base64-encoded 128-bit MD5 hash of the submitted data. • The value of the Content-MD5 header is not a valid hash value.	400 Bad Request
BucketAlreadyExists	You are trying to create a bucket, but a bucket with the specified name already exists and either is owned by a user other than you or has no owner.	409 Conflict
BucketAlreadyOwnedByYou	You are trying to create a bucket, but a bucket with the specified name already exists and is owned by you.	200 OK
BucketNotEmpty	You are trying to delete a bucket that is not empty.	409 Conflict
InvalidAccessKeyId	The access key provided in the request does not correspond to a valid user account.	403 Forbidden

(Continued)

Error code	Description	HTTP status code
InvalidArgument	One of: • A query parameter has an invalid value. • An ACL grant header specifies an invalid grantee. • The request includes conflicting conditional headers (for example, If-Match and If-None-Match).	400 Bad Request
InvalidBucketName	The specified bucket name is invalid.	400 Bad Request
InvalidBucketState	One of: • The target bucket does not currently support the requested operation. • The tenant does not currently support use of the HS3 API.	409 Conflict
InvalidRange	You are trying to retrieve part of an object, and one of these is true: • The specified start position is greater than the size of the requested data. • The size of the specified range is zero.	416 Requested Range Not Satisfiable
InvalidRequest	You are trying to create a bucket, and one of these is true: • You already own the maximum number buckets allowed per user. • The tenant does not have enough space for another bucket. • The tenant already has the maximum number of buckets it's allowed to have. • An unexpected error occurred.	400 Bad Request
InternalError	An internal error occurred. Try the request again, gradually increasing the delay between each successive attempt. If this error persists, contact your tenant administrator.	500 Internal Server Error
KeyTooLong	The specified object name is too long.	400 Bad Request

(Continued)

Error code	Description	HTTP status code
MalformedXML	The request is invalid. Possible reasons include: • The object name specified in the request is invalid. • A query parameter specifies an invalid value. • The request includes both the x-amz-acl header and an individual ACL grant header. • The XML in the request body is malformed or contains invalid values.	400 Bad Request
MetadataTooLarge	The custom metadata you are trying to store is larger than two KB.	400 Bad Request
MethodNotAllowed	The requested HTTP method is not supported for the target tenant, bucket, or object.	405 Method Not Allowed
NoSuchBucket	The specified bucket does not exist.	404 Not Found
NoSuchKey	HCP could not find an object with the specified name.	404 Not Found
NotImplemented	The requested operation is not supported.	501 Not Implemented
OperationAborted	One of: • HCP cannot perform the requested operation because a conflicting operation is already in progress. • You are trying to delete an object that is under retention. • You are trying to create an object with the same name as an existing object, and versioning is disabled. • You are trying to read, add an ACL to, or delete an object that is currently being written to the bucket.	409 Conflict
OutOfSpace	The object you are trying to store or the custom metadata you are trying to add is too big for the amount of space left in the bucket.	413 Request Entity Too Large
PreconditionFailed	HCP did not perform the requested operation because a specified precondition for the operation was not satisfied.	304 Not Modified *or* 412 Precondition Failed

(Continued)

Error code	Description	HTTP status code
ServiceUnavailable	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. If this error persists, contact your tenant administrator.	503 Service Unavailable
SignatureDoesNotMatch	The signature provided with the request does not match the request contents. Check that the secret key and signing method used are correct.	403 Forbidden

5

Working with buckets

With the HS3 API, you can perform operations on individual buckets. You can also list all the buckets you own.

For each bucket operation you can request, this chapter:

- Describes the operation
- Shows the request line for the operation
- Describes the request headers for the operation
- Describes the response headers returned for a successful execution of the requested operation
- Shows the format of the request or response body, where applicable
- Explains the HTTP status codes that can be returned in response to requests for the operation
- Presents one or more examples of requests for the operation

For general information about HS3 requests and HCP responses to those requests, see Chapter 4, “Requests and responses,” on page 47.

For information about the examples in this chapter, see “Examples in this book” on page 12.

Creating a bucket

You use the HTTP **PUT** method to create a bucket. You can create a bucket only if your user account is configured to let you do so.

When you create a bucket, you specify a name for it. For information on naming buckets, see "Bucket names" on page 14.

You can specify an ACL for a bucket in the same request as you use to create the bucket. To do this, you need to use ACL headers. You cannot use an ACL request body when creating a bucket. For information on ACLs, see "Access control lists" on page 22.

If the ACL you specify in a request to create a bucket is invalid, HCP returns a 400 (Bad Request) or 501 (Not Implemented) status code and does not create the bucket.

When you create a bucket, you become the bucket owner. You also get browse, read, read ACL, write, write ACL, and delete data access permissions for the bucket. If search is enabled for the bucket, you also get search permission.

For information on bucket ownership, see "Bucket owners" on page 21. For information on data access permissions, see "Data access permissions" on page 10. For information on search, see "Search" on page 34.

Request line

Depending on whether the bucket name is included in the hostname in the HS3 request, the request line for a request to create a bucket has either of these formats:

- With the bucket name included in the hostname:

  ```
  PUT / HTTP/1.1
  ```

- With the bucket name following the hostname:

  ```
  PUT /bucket-name HTTP/1.1
  ```

Request headers

The table below describes the headers you can use in a request to create a bucket.

Request header	Description	Required
Authorization	See "AWS authentication" on page 42. You cannot create a bucket as an anonymous user.	Yes
Content-Length	Specifies the size, in bytes, of the request body. For a request to create a bucket, the value of this header must be **0** (zero).	Yes
Date	See "Common request headers" on page 49.	Date or x-amz-date
Host	See "Common request headers" on page 49.	Yes
x-amz-acl	Adds a canned ACL to the bucket. This header is required for adding a canned ACL to a bucket. If you're using individual x-amz-grant- headers to add the ACL, the x-amz-acl header is invalid. For valid values for this header, see "Canned ACLs" on page 25.	No
x-amz-date	See "Common request headers" on page 49.	x-amz-date or Date
x-amz-grant-full-control	Grants full control over the bucket to one or more specified grantees. If you're using a canned ACL to add an ACL to a bucket, the x-amz-grant-full-control header is invalid. For valid values for this and the following x-amz-grant- headers, see "Using individual grant headers" on page 26.	No
x-amz-grant-read	Grants the browse and read data access permissions for the bucket to one or more specified grantees. If you're using a canned ACL to add an ACL to a bucket, the x-amz-grant-read header is invalid.	No
x-amz-grant-read-acp	Grants the read ACL data access permission for the bucket to one or more specified grantees. If you're using a canned ACL to add an ACL to a bucket, the x-amz-grant-read-acp header is invalid.	No

(Continued)

Request header	Description	Required
x-amz-grant-write	Grants the write and delete data access permissions for the bucket to one or more specified grantees. If you're using a canned ACL to add an ACL to a bucket, the x-amz-grant-write header is invalid.	No
x-amz-grant-write-acp	Grants the write ACL data access permission for the bucket to one or more specified grantees. If you're using a canned ACL to add an ACL to a bucket, the x-amz-grant-write-acp header is invalid.	No

Response headers

The table below describes the response headers returned in response to a successful request to create a bucket.

Response header	Description
Content-Length	Specifies the size, in bytes, of the response body. In response to a successful request to create a bucket, the value of this header is always **0** (zero).
Date	See "Common response headers" on page 52.
Location	Specifies the name of the bucket created by the request.
Server	See "Common response headers" on page 52.

HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to create a bucket. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 53.

Code	Meaning	Description
200	OK	One of: • The bucket was successfully created. • A bucket with the specified name already exists and is owned by you.

(Continued)

Code	Meaning	Description
400	Bad Request	One of: • The specified bucket name is invalid. • An ACL grant header specifies an invalid grantee. • You already own the maximum number buckets allowed per user. • The tenant does not have enough space for another bucket. • The tenant already has the maximum number of buckets it’s allowed to have.
403	Forbidden	One of: • The credentials provided with the request are invalid. • You do not have permission to create buckets. • The tenant does not currently support use of the HS3 API.
409	Conflict	A bucket with the specified name already exists and either is owned by a user other than you or has no owner.
500	Internal Server Error	An internal error occurred. Try the request again, gradually increasing the delay between each successive attempt. If this error persists, contact your tenant administrator.
501	Not Implemented	The request includes the x-amz-acl header with an invalid value.
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. If this error persists, contact your tenant administrator.

Examples

The following sections show sample **PUT** requests for creating buckets.

Example 1: Creating a bucket

Here's a sample **PUT** request that creates a bucket named finance in the context of the tenant named europe.

Request with s3curl command line

```
./s3curl.pl  --id=lgreen  --createBucket  --  -k  "https://finance.europe.hcp.example.com"
```

Request headers

```
PUT / HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Mon, 18 Mar 2013 12:59:11 +0000
Authorization: AWS bGdyZWVu:1RXSNMipYpv4IOpfNS9Odi1UBWM=
Content-Length: 0
```

Response headers

```
HTTP/1.1 200 OK
Date: Mon, 18 Mar 2013 12:59:11 GMT
Server: HCP V6.0.1.117
Location: /finance
Content-Length: 0
```

Example 2: Creating a bucket with an ACL

Here's a sample **PUT** request that creates a bucket named human-resources and adds an ACL to the bucket. The ACL grants read permission to all users and write permission to the users with usernames mwhite and pdgrey.

Request with s3curl command line

```
./s3curl.pl  --id=lgreen  --createBucket  --  -k
    "https://human-resources.europe.hcp.example.com"
    -H  "x-amz-grant-read:emailAddress=all_users"
    -H  "x-amz-grant-write:emailAddress=mwhite,  emailAddress=pdgrey"
```

Request headers

```
PUT / HTTP/1.1
Host: human-resources.europe.hcp.example.com
Date: Mon, 18 Mar 2013 19:46:03 +0000
Authorization: AWS bGdyZWVu:0WCfi79j2QtCczA6TGutnJWNRm4=
x-amz-grant-read: emailAddress=all_users
x-amz-grant-write: emailAddress=mwhite, emailAddress=pdgrey
Content-Length: 0
```

Response headers

```
HTTP/1.1 200 OK
Date: Mon, 18 Mar 2013 19:46:03 GMT
Server: HCP V6.0.1.117
Location: /human-resources
Content-Length: 0
```

Listing the buckets you own

You use the HTTP **GET** method to list the buckets you own. You need a user account to request this operation.

The target of a request to list the buckets you own is a tenant (that is, the service point). The list of buckets in the response contains only buckets created in the context of that tenant and only those that have the HS3 API enabled. The buckets are listed in alphanumeric order.

The list of buckets you own is returned in an XML response body. For information on the format of this response body, see "Response body" on page 67.

Request line

The request line for a request to list the buckets you own has this format:

```
GET / HTTP/1.1
```

Request headers

The table below describes the headers you can use in a request to list the buckets you own.

Request header	Description	Required
Authorization	See "AWS authentication" on page 42.	Yes
Date	See "Common request headers" on page 49.	Date or x-amz-date
Host	See "Common request headers" on page 49.	Yes
x-amz-date	See "Common request headers" on page 49.	x-amz-date or Date
x-hcp-pretty-print	Optionally, requests that the XML response body be formatted for readability. Valid values are: • **true** — Format the XML response body for readability. • **false** — Do not apply any special formatting to the XML response body. The default is **false**. The values **true** and **false** are not case sensitive.	No

Response headers

The table below describes the response headers returned in response to a successful request to list the buckets you own.

Response header	Description
Content-Type	Specifies the Internet media type of the response body. For a request to list the buckets you own, the value of this header is always **application/xml;charset=UTF-8**.
Date	See "Common response headers" on page 52.
Server	See "Common response headers" on page 52.
Transfer-Encoding	Indicates that HCP could not determine the size of the response body before formulating the response. For a request to list the buckets you own, the value of this header is always **chunked**.

Response body

HCP returns the list of buckets you own in an XML response body, in this format:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Owner>
        <ID>bucket-owner-user-id</ID>
        <DisplayName>bucket-owner-username</DisplayName>
    </Owner>
    <Buckets>
        <Bucket>
            <Name>bucket-name</Name>
            <CreationDate>bucket-creation-date-and-time</CreationDate>
        </Bucket>
        .
        .
        .
    </Buckets>
</ListAllMyBucketsResult>
```

The table below describes XML elements in this response body. The elements are listed in alphabetical order.

Element	Description
Bucket	Child of the **ListAllMyBucketsResult** element and container for the **Name** and **CreationDate** elements. The response body contains one **Bucket** element for each bucket you own.
CreationDate	Child of the **Bucket** element. The **CreationDate** element specifies the date and time at which the applicable bucket was created, in Greenwich Mean Time (GMT). The date and time are expressed in this format: *yyyy*-*MM*-*dd*T*HH*:*mm*:*ss*.*SSS*Z For example: 2013-03-18T19:46:03.856Z

(Continued)

Element	Description
DisplayName	Child of the **Owner** element. If the bucket owner is identified by an HCP user account, the value of the **DisplayName** element is the username for that account. If the bucket owner is identified by an AD user account, the value of the **DisplayName** element is the username of that account followed by an at sign (@) and the AD domain name.
ID	Child of the **Owner** element. If the bucket owner is identified by an HCP user account, the value of the **ID** element is the user ID for that account. If the bucket owner is identified by an AD user account, the value of the **ID** element is the SID for that account.
ListAllMyBuckets Result	Root element.
Name	Child of the **Bucket** element. The **Name** element specifies the name of a bucket.
Owner	Child of the **ListAllMyBucketsResult** element and container for the **ID** and **DisplayName** elements. The **Owner** element identifies the owner of the applicable bucket.

HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to list the buckets you own. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 53.

Code	Meaning	Description
200	OK	HCP successfully retrieved the list of buckets you own.
403	Forbidden	One of: • The credentials provided with the request are invalid. • The tenant does not currently support use of the HS3 API. • The specified tenant does not exist.

(Continued)

Code	Meaning	Description
500	Internal Server Error	An internal error occurred. Try the request again, gradually increasing the delay between each successive attempt. If this error persists, 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. If this error persists, contact your tenant administrator.

Example: Listing the buckets you own

Here's a sample **GET** request that returns a list of the buckets owned by user lgreen in the context of the europe tenant.

Request with s3curl command line

```
./s3curl.pl --id=lgreen -- -k "https://europe.hcp.example.com"
    -H "x-hcp-pretty-print:true"
```

Request headers

```
GET / HTTP/1.1
Host: europe.hcp.example.com
Date: Tue, 19 Mar 2013 00:06:34 +0000
Authorization: AWS bGdyZWVu:Gek+OrFpyg06Bufgg+TW6kH5ISA=
x-hcp-pretty-print: true
```

Response headers

```
HTTP/1.1 200 OK
Date: Tue, 19 Mar 2013 00:06:34 GMT
Server: HCP V6.0.1.117
Content-Type: application/xml;charset=UTF-8
Transfer-Encoding: chunked
```

Response body

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Owner>
        <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
        <DisplayName>lgreen</DisplayName>
    </Owner>
    <Buckets>
        <Bucket>
            <Name>finance</Name>
            <CreationDate>2013-03-18T12:59:11.898Z</CreationDate>
        </Bucket>
        <Bucket>
            <Name>human-resources</Name>
            <CreationDate>2013-03-18T19:46:03.856Z</CreationDate>
        </Bucket>
    </Buckets>
</ListAllMyBucketsResult>
```

Checking the existence of a bucket

You use the HTTP **HEAD** method to check the existence of a bucket. To successfully check the existence of a bucket, you need read permission for the bucket.

If the bucket you specify in the **HEAD** request does not exist, HCP returns a 404 (Not Found) status code. If the bucket exists but you do not have read permission for it, HCP returns a 403 (Forbidden) status code.

Request line

Depending on whether the bucket name is included in the hostname in the HS3 request, the request line for a request to check the existence of a bucket has either of these formats:

- With the bucket name included in the hostname:

  ```
  HEAD / HTTP/1.1
  ```

- With the bucket name following the hostname:

  ```
  HEAD /bucket-name HTTP/1.1
  ```

Request headers

The table below describes the headers you can use in a request to check the existence of a bucket.

Request header	Description	Required
Authorization	See "AWS authentication" on page 42.	Yes
Date	See "Common request headers" on page 49.	Date or x-amz-date
Host	See "Common request headers" on page 49.	Yes
x-amz-date	See "Common request headers" on page 49.	x-amz-date or Date

Response headers

The table below describes the response headers returned in response to a successful request to check the existence of a bucket.

Response header	Description
Content-Length	Specifies the size, in bytes, of the response body. In response to a successful request to check the existence of a bucket, the value of this header is always **0** (zero).
Date	See "Common response headers" on page 52.
Server	See "Common response headers" on page 52.

HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to check the existence of a bucket. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 53.

Code	Meaning	Description
200	OK	The specified bucket exists.

(Continued)

Code	Meaning	Description
403	Forbidden	One of: • The credentials provided with the request are invalid. • The specified bucket exists, but you do not have read permission for it. • The specified bucket does not currently support the requested operation. • The HS3 API is currently disabled for the specified bucket. • The tenant does not currently support use of the HS3 API.
404	Not Found	The specified bucket does not exist.
500	Internal Server Error	An internal error occurred. Try the request again, gradually increasing the delay between each successive attempt. If this error persists, 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. If this error persists, contact your tenant administrator.

Example: Checking the existence of a bucket

Here’s a sample **HEAD** request that checks the existence of a bucket named finance.

Request with s3curl command line

```
./s3curl.pl --id=lgreen --head -- -k "https://finance.hcp.example.com"
```

Request headers

```
HEAD / HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Tue, 19 Mar 2013 20:08:59 +0000
Authorization: AWS bGdyZWVu:N2UfjwTaydEqS45O5vgcoGerZKw=
```

Response headers

```
HTTP/1.1 200 OK
Date: Tue, 19 Mar 2013 20:08:59 GMT
Server: HCP V6.0.1.117
Content-Length: 0
```

Adding an ACL to a bucket

You use the HTTP **PUT** method with the **acl** query parameter to add an ACL to an existing bucket. Adding an ACL to a bucket replaces any existing ACL in its entirety. You cannot modify an existing ACL in place.

To add an ACL to a bucket, you need write ACL permission for the bucket.

To add an ACL to a bucket, you can use either request headers or an ACL request body. You cannot use ACL headers and an ACL request body in the same request.

With ACL headers, you can specify either a canned ACL or individual x-amz-grant- headers. You cannot specify both a canned ACL and an x-amz-grant- header in the same request.

You can use an ACL request body to change the owner of a bucket you own. You cannot use ACL headers to do this. To change the owner of a bucket, you need both write ACL and change owner permission for the bucket.

If you try to add an ACL that specifies a user account that does not exist, HCP returns a 400 (Bad Request) status code and does not add the ACL to the bucket.

For an introduction to ACLs and information on how to specify them, see "Access control lists" on page 22.

Request line

Depending on whether the bucket name is included in the hostname in the HS3 request, the request line for a request to add an ACL to a bucket has either of these formats:

- With the bucket name included in the hostname:

  ```
  PUT /?acl HTTP/1.1
  ```

- With the bucket name following the hostname:

  ```
  PUT /bucket-name?acl HTTP/1.1
  ```

The **acl** query parameter is not case sensitive.

Request headers

The table below describes the headers you can use in a request to add an ACL to a bucket.

Request header	Description	Required
Authorization	See "AWS authentication" on page 42.	Yes
Content-Length	Specifies the size, in bytes, of the ACL request body. This header is required when you're using an ACL request body to add an ACL to the bucket. If you're using ACL headers to add the ACL, the Content-Length header is invalid.	No
Content-Type	Specifies the Internet media type of the request body. This header is valid only when the ACL is specified in the request body. The only valid value is **application/xml.**	No
Date	See "Common request headers" on page 49.	Date or x-amz-date
Host	See "Common request headers" on page 49.	Yes
x-amz-acl	Adds a canned ACL to the bucket. This header is required for adding a canned ACL to a bucket. If you're using an ACL request body or individual x-amz-grant- headers to add the ACL, the x-amz-acl header is invalid. For valid values for this header, see "Canned ACLs" on page 25.	No
x-amz-date	See "Common request headers" on page 49.	x-amz-date or Date

(Continued)

Request header	Description	Required
x-amz-grant-full-control	Grants full control over the bucket to one or more specified grantees. If you're using an ACL request body or a canned ACL to add an ACL to the bucket, the x-amz-grant-full-control header is invalid. For valid values for this and the following x-amz-grant- headers, see "Using individual grant headers" on page 26.	No
x-amz-grant-read	Grants the browse and read data access permissions for the bucket to one or more specified grantees. If you're using an ACL request body or a canned ACL to add an ACL to the bucket, the x-amz-grant-read header is invalid.	No
x-amz-grant-read-acp	Grants the read ACL data access permission for the bucket to one or more specified grantees. If you're using an ACL request body or a canned ACL to add an ACL to the bucket, the x-amz-grant-read-acp header is invalid.	No
x-amz-grant-write	Grants the write and delete data access permissions for the bucket to one or more specified grantees. If you're using an ACL request body or a canned ACL to add an ACL to the bucket, the x-amz-grant-write header is invalid.	No
x-amz-grant-write-acp	Grants the write ACL data access permission for the bucket to one or more specified grantees. If you're using an ACL request body or a canned ACL to add an ACL to the bucket, the x-amz-grant-write-acp header is invalid.	No

Response headers

The table below describes the response headers returned in response to a successful request to add an ACL to a bucket.

Response header	Description
Content-Length	Specifies the size, in bytes, of the response body. In response to a successful request to add an ACL to a bucket, the value of this header is always **0** (zero).
Date	See "Common response headers" on page 52.
Server	See "Common response headers" on page 52.

HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to add an ACL to a bucket. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 53.

Code	Meaning	Description
200	OK	HCP successfully added the ACL to the bucket.
400	Bad Request	One of: • You are trying to add an ACL that contains more than one hundred permission grants. • A specified grantee does not exist. • The specified owner does not exist. • Two grants of the same permission specify the same grantee. • The x-amz-acl header specifies an invalid value. • An x-amz-grant- header specifies an invalid identifier type. • The XML in the ACL request body is malformed or contains an invalid value.
403	Forbidden	One of: • The credentials provided with the request are invalid. • You do not have permission to add an ACL to the bucket. • The specified bucket does not currently support the requested operation. • The HS3 API is currently disabled for the specified bucket. • The tenant does not currently support use of the HS3 API.
404	Not Found	The specified bucket does not exist.

(Continued)

Code	Meaning	Description
500	Internal Server Error	An internal error occurred. Try the request again, gradually increasing the delay between each successive attempt. If this error persists, 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. If this error persists, contact your tenant administrator.

Examples

The following sections show sample **PUT** requests for adding ACLs to buckets.

Example 1: Adding an ACL to a bucket by specifying individual grants

Here's a sample **PUT** request that adds an ACL to the finance bucket by using x-amz-grant- headers. The ACL grants read permission to all users, write permission to user pdgrey, and write, read ACL, and write ACL permission to user mwhite.

Request with s3curl command line

```
./s3curl.pl --id=lgreen --put ~ -- -k "https://europe.hcp.example.com/finance?acl"
    -H "x-amz-grant-read:emailAddress=all_users"
    -H "x-amz-grant-write:emailAddress=pdgrey, emailAddress=mwhite"
    -H "x-amz-grant-read-acp:emailAddress=mwhite"
    -H "x-amz-grant-write-acp:emailAddress=mwhite"
```

Request headers

```
PUT /finance?acl HTTP/1.1
Host: europe.hcp.example.com
 Date: Wed, 20 Mar 2013 15:29:09 +0000
Authorization: AWS bGdyZWVu:RuKD8rwRevmwLo+ZMhF5beGq7Qk=
x-amz-grant-read: emailAddress=all_users
x-amz-grant-write: emailAddress=pdgrey, emailAddress=mwhite
x-amz-grant-read-acp: emailAddress=mwhite
x-amz-grant-write-acp: emailAddress=mwhite
```

Response headers

```
HTTP/1.1 200 OK
Date: Wed, 20 Mar 2013 15:29:27 GMT
Server: HCP V6.0.1.117
Content-Length: 0
```

Example 2: Adding an ACL to a bucket by using an ACL request body

Here's a sample **PUT** request that uses an ACL request body to add an ACL to the finance bucket and, at the same time, change the owner of the bucket. The ACL grants read and write access to the bucket to user lgreen and changes the bucket owner to user mwhite. The ACL request body is in a file named acl-7.xml.

Request body

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Owner>
        <ID>b9d39144-a081-4760-b0e8-b8fb51e10192</ID>
        <DisplayName>mwhite</DisplayName>
    </Owner>
    <AccessControlList>
        <Grant>
            <Grantee xsi:type="CanonicalUser"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
                <DisplayName>lgreen</DisplayName>
            </Grantee>
            <Permission>READ</Permission>
        </Grant>
        <Grant>
            <Grantee xsi:type="CanonicalUser"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
                <DisplayName>lgreen</DisplayName>
            </Grantee>
            <Permission>WRITE</Permission>
        </Grant>
    </AccessControlList>
</AccessControlPolicy>
```

Request with s3curl command line

```
./s3curl.pl  --id=lgreen  --put  acl-7.xml  --  -k
    "https://europe.hcp.example.com/finance?acl"
```

Request headers

```
PUT /finance?acl HTTP/1.1
Host: europe.hcp.example.com
Date: Wed, 20 Mar 2013 16:03:02 +0000
Authorization: AWS bGdyZWVu:mVNu4YolbdvK+PVzmafhOvd1VgU=
Content-Length: 727
```

Response headers

```
HTTP/1.1 200 OK
Date: Wed, 20 Mar 2013 16:03:02 GMT
Server: HCP V6.0.1.117
Content-Length: 0
```

Retrieving the ACL for a bucket

You use the HTTP **GET** method with the **acl** query parameter to retrieve the ACL for a bucket. To retrieve the ACL for a bucket, you need read ACL permission for the bucket.

The bucket ACL is returned in an XML response body. The format of the response body is the same as the format you use for the ACL request body when you add an ACL to a bucket. For the format of the ACL request body, see "Specifying an ACL in the request body" on page 27.

For an introduction to ACLs, see "Access control lists" on page 22.

Request line

Depending on whether the bucket name is included in the hostname in the HS3 request, the request line for a request to retrieve a bucket ACL has either of these formats:

- With the bucket name included in the hostname:

  `GET /?acl HTTP/1.1`

- With the bucket name following the hostname:

  `GET` */bucket-name***`?acl HTTP/1.1`**

The **acl** query parameter is not case sensitive.

Request headers

The table below describes the headers you can use in a request to retrieve a bucket ACL.

Request header	Description	Required
Authorization	See "AWS authentication" on page 42.	Yes
Date	See "Common request headers" on page 49.	Date or x-amz-date
Host	See "Common request headers" on page 49.	Yes
x-amz-date	See "Common request headers" on page 49.	x-amz-date or Date
x-hcp-pretty-print	Optionally, requests that the XML response body be formatted for readability. Valid values are: • **true** — Format the XML response body for readability. • **false** — Do not apply any special formatting to the XML response body. The default is **false**. The values **true** and **false** are not case sensitive.	No

Response headers

The table below describes the response headers returned in response to a successful request to retrieve a bucket ACL.

Response header	Description
Content-Type	Specifies the Internet media type of the response body. For a request to retrieve a bucket ACL, the value of this header is always **application/xml;charset=UTF-8**.
Date	See "Common response headers" on page 52.
Server	See "Common response headers" on page 52.
Transfer-Encoding	Indicates that HCP could not determine the size of the response body before formulating the response. For a request to retrieve a bucket ACL, the value of this header is always **chunked**.

HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to retrieve a bucket ACL. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 53.

Code	Meaning	Description
200	OK	HCP successfully retrieved the bucket ACL.
403	Forbidden	One of: • The credentials provided with the request are invalid. • You do not have permission to read the bucket ACL. • The specified bucket does not currently support the requested operation. • The HS3 API is currently disabled for the specified bucket. • The tenant does not currently support use of the HS3 API.
404	Not Found	The specified bucket does not exist.
500	Internal Server Error	An internal error occurred. Try the request again, gradually increasing the delay between each successive attempt. If this error persists, 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. If this error persists, contact your tenant administrator.

Example: Retrieving a bucket ACL

Here's a sample **GET** request that retrieves the ACL for the bucket named finance.

Request with s3curl command line

```
./s3curl.pl  --id=lgreen  --  -k  "https://finance.europe.hcp.example.com?acl"
     -H  "x-hcp-pretty-print:true"
```

Request headers

```
GET /?acl HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Tue, 19 Mar 2013 21:57:45 +0000
Authorization: AWS bGdyZWVu:Msry4PBtztkM6FMvzdDblC5RoPE=
x-hcp-pretty-print: true
```

Response headers

```
HTTP/1.1 200 OK
Date: Tue, 19 Mar 2013 21:57:45 GMT
Server: HCP V6.0.1.117
Content-Type: application/xml;charset=UTF-8
Transfer-Encoding: chunked
```

Response body

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Owner>
        <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
        <DisplayName>lgreen</DisplayName>
    </Owner>
    <AccessControlList>
        <Grant>
            <Grantee xsi:type="Group"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <URI>http://acs.amazonaws.com/groups/global/AuthenticatedUsers
                </URI>
            </Grantee>
            <Permission>READ</Permission>
        </Grant>
        <Grant>
            <Grantee xsi:type="Group"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
            </Grantee>
            <Permission>READ</Permission>
        </Grant>
        <Grant>
            <Grantee xsi:type="CanonicalUser"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <ID>b9d39144-a081-4760-b0e8-b8fb51e10192</ID>
                <DisplayName>mwhite</DisplayName>
            </Grantee>
            <Permission>READ_ACP</Permission>
        </Grant>
        <Grant>
            <Grantee xsi:type="CanonicalUser"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <ID>b9d39144-a081-4760-b0e8-b8fb51e10192</ID>
                <DisplayName>mwhite</DisplayName>
            </Grantee>
            <Permission>WRITE_ACP</Permission>
        </Grant>
        <Grant>
            <Grantee xsi:type="CanonicalUser"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <ID>b9d39144-a081-4760-b0e8-b8fb51e10192</ID>
                <DisplayName>mwhite</DisplayName>
            </Grantee>
            <Permission>WRITE</Permission>
        </Grant>
        <Grant>
            <Grantee xsi:type="CanonicalUser"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
                <DisplayName>lgreen</DisplayName>
            </Grantee>
            <Permission>FULL_CONTROL</Permission>
        </Grant>
        <Grant>
            <Grantee xsi:type="CanonicalUser"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <ID>b9d39144-a081-4763-b0e8-b8fb51e10192</ID>
                <DisplayName>pdgrey</DisplayName>
            </Grantee>
            <Permission>WRITE</Permission>
        </Grant>
    </AccessControlList>
</AccessControlPolicy>
```

Enabling or disabling versioning for a bucket

You use the HTTP **PUT** method with the **versioning** query parameter to enable or disable versioning for a bucket. To enable or disable versioning for a bucket, you need to be the bucket owner.

While versioning is enabled for a bucket, the versioning status of the bucket is Enabled. If you disable versioning for a bucket after it has been enabled, the versioning status changes to Suspended. If versioning has never been enabled for a bucket, the bucket has no explicit versioning status.

To enable or disable versioning for a bucket, you specify the new versioning status in an XML request body. For the format of this request body, see "Request body" below.

For an introduction to versioning, see "Versioning" on page 32.

Request body

For the content of a versioning request body, you use XML in this format:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<VersioningConfiguration
  xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>versioning-status</Status>
</VersioningConfiguration>
```

The table below describes XML elements in a versioning request body. The elements are listed in alphabetical order.

Element	Description
Status	Child of the **VersioningConfiguration** element. Valid values for the **Status** element are: • **Enabled** — Enables versioning for the bucket • **Suspended** — Disables versioning for the bucket
Versioning Configuration	Root element. This must be the first element in the versioning request body. The **VersioningConfiguration** element must include this XML namespace specification: xmlns="http://s3.amazonaws.com/doc/2006-03-01/"> The **VersioningConfiguration** element is a container for the **Status** element.

Request line

Depending on whether the bucket name is included in the hostname in the HS3 request, the request line for a request to enable or disable versioning for a bucket has either of these formats:

- With the bucket name included in the hostname:

  ```
  PUT /?versioning HTTP/1.1
  ```

- With the bucket name following the hostname:

  ```
  PUT /bucket-name?versioning HTTP/1.1
  ```

The **versioning** query parameter is not case sensitive.

Request headers

The table below describes the headers you can use in a request to enable or disable versioning for a bucket.

Request header	Description	Required
Authorization	See "AWS authentication" on page 42.	Yes
Content-Length	Specifies the size, in bytes, of the versioning request body.	Yes
Content-Type	Specifies the Internet media type of the request body. The only valid value is **application/xml.**	No
Date	See "Common request headers" on page 49.	Date or x-amz-date
Host	See "Common request headers" on page 49.	Yes
x-amz-date	See "Common request headers" on page 49.	x-amz-date or Date

Response headers

The table below describes the response headers returned in response to a successful request to enable or disable versioning for a bucket.

Response header	Description
Content-Length	Specifies the size, in bytes, of the response body. In response to a successful request to enable or disable versioning for a bucket, the value of this header is always **0** (zero).

(Continued)

Response header	Description
Date	See "Common response headers" on page 52.
Location	Specifies the name of the target bucket.
Server	See "Common response headers" on page 52.

HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to enable or disable versioning for a bucket. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 53.

Code	Meaning	Description
200	OK	HCP successfully enabled or disabled versioning for the bucket.
400	Bad Request	The XML in the versioning request body is malformed or contains an invalid value.
403	Forbidden	One of: • The credentials provided with the request are invalid. • You are not the bucket owner. • The HS3 API is currently disabled for the specified bucket. • The tenant does not currently support use of the HS3 API.
404	Not Found	The specified bucket does not exist.
500	Internal Server Error	An internal error occurred. Try the request again, gradually increasing the delay between each successive attempt. If this error persists, 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. If this error persists, contact your tenant administrator.

Example: Enabling versioning for a bucket

Here’s a sample **PUT** request that enables versioning for the finance bucket. The versioning request body is in a file named versioning.xml.

Request body

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Status>Enabled</Status>
</VersioningConfiguration>
```

Request with s3curl command line

```
./s3curl.pl --id=lgreen --put versioning.xml -- -k
    "https://europe.hcp.example.com/finance?versioning"
```

Request headers

```
PUT /finance?versioning HTTP/1.1
Host: europe.hcp.example.com
Date: Wed, 20 Mar 2013 13:40:15 +0000
Authorization: AWS bGdyZWVu:ETdibK607bVlNdxb0aKCttJU33U=
Content-Length: 182
```

Response headers

```
HTTP/1.1 200 OK
Date: Wed, 20 Mar 2013 13:40:15 GMT
Server: HCP V6.0.1.117
Location: /finance
Content-Length: 0
```

Checking the versioning status of a bucket

You use the HTTP **GET** method with the **versioning** query parameter to check the versioning status of a bucket. To check the versioning status of a bucket, you need to be the bucket owner.

The versioning status is returned in an XML response body. The format of the response body is the same as the format you use for the versioning request body when you enable or disable versioning for a bucket. For the format of the versioning request body, see “Request body” on page 84.

For an introduction to versioning, see “Versioning” on page 32.

Request line

Depending on whether the bucket name is included in the hostname in the HS3 request, the request line for a request to check the versioning status of a bucket has either of these formats:

- With the bucket name included in the hostname:

  `GET /?versioning HTTP/1.1`

- With the bucket name following the hostname:

  `GET /`*`bucket-name`***`?versioning HTTP/1.1`**

The **versioning** query parameter is not case sensitive.

Request headers

The table below describes the headers you can use in a request to check the versioning status of a bucket.

Request header	Description	Required
Authorization	See "AWS authentication" on page 42.	Yes
Date	See "Common request headers" on page 49.	Date or x-amz-date
Host	See "Common request headers" on page 49.	Yes
x-amz-date	See "Common request headers" on page 49.	x-amz-date or Date
x-hcp-pretty-print	Optionally, requests that the XML response body be formatted for readability. Valid values are: • **true** — Format the XML response body for readability. • **false** — Do not apply any special formatting to the XML response body. The default is **false**. The values **true** and **false** are not case sensitive.	No

Response headers

The table below describes the response headers returned in response to a successful request to check the versioning status of a bucket.

Response header	Description
Content-Type	Specifies the Internet media type of the response body. For a request to check the versioning status of a bucket, the value of this header is always **application/ xml;charset=UTF-8**.
Date	See "Common response headers" on page 52.
Server	See "Common response headers" on page 52.
Transfer-Encoding	Indicates that HCP could not determine the size of the response body before formulating the response. For a request to check the versioning status of a bucket, the value of this header is always **chunked**.

HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to check the versioning status of a bucket. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 53.

Code	Meaning	Description
200	OK	HCP successfully retrieved the versioning status.
403	Forbidden	One of: • The credentials provided with the request are invalid. • You are not the bucket owner. • The HS3 API is currently disabled for the specified bucket. • The tenant does not currently support use of the HS3 API.
404	Not Found	The specified bucket does not exist.

(Continued)

Code	Meaning	Description
500	Internal Server Error	An internal error occurred. Try the request again, gradually increasing the delay between each successive attempt. If this error persists, 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. If this error persists, contact your tenant administrator.

Example: Checking the versioning status of a bucket

Here's a sample **GET** request that retrieves the versioning status of the bucket named finance.

Request with s3curl command line

```
./s3curl.pl --id=lgreen -- -k "https://finance.europe.hcp.example.com?versioning"
    -H "x-hcp-pretty-print:true"
```

Request headers

```
GET /?versioning HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Wed, 20 Mar 2013 18:31:50 +0000
Authorization: AWS bGdyZWVu:2rmMzjz+08PWDb/4Kd1nD43Wf1s=
x-hcp-pretty-print: true
```

Response headers

```
HTTP/1.1 200 OK
Date: Wed, 20 Mar 2013 18:31:50 GMT
Server: HCP V6.0.1.117
Content-Type: application/xml;charset=UTF-8
Transfer-Encoding: chunked
```

Response body

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Status>Enabled</Status>
</VersioningConfiguration>
```

Listing bucket contents

You use the HTTP **GET** method to list the contents of a bucket. To list the contents of a bucket, you need browse permission for the bucket.

For the purpose of a bucket listing, the bucket contents consist not only of the objects you stored in the bucket but also of the folders that you created in the bucket or that HCP created automatically from the object names. For example, by default, if a bucket contains an object named quarterly_rpts/Q4_2012, a list of the bucket contents includes these two items:

```
quarterly_rpts/
quarterly_rpts/Q4_2012
```

By default, a bucket listing includes only the current (or only) versions of objects. To request a listing that includes old versions of objects, you use the **versions** query parameter.

A listing that includes versions of objects may include deleted versions. A deleted version is a marker that indicates that the previous version of the object was deleted. For an introduction to object versions, see "Versioning" on page 32.

You can use query parameters to list only a subset of the items in a bucket. For information on using these parameters, see "Limiting the bucket listing" on page 92.

A bucket listing is returned in an XML response body. In the response body, items in the bucket listing occur in alphanumeric order. If the listing includes multiple versions of an object, those versions are listed in chronological order with the oldest version first.

The format of the bucket listing response body differs depending on whether the **GET** request includes the **versions** query parameter. For the formats of the bucket listing response body, see "Response body" on page 100.

By default, a bucket listing can include at most one thousand items. However, you can use the **max-keys** query parameter in a request to specify a smaller maximum number of items. If more than the maximum number of items satisfy the criteria for a request, you can resubmit the request using query parameters to retrieve the next part of the listing.

Limiting the bucket listing

You use query parameters to limit the items included in a bucket listing. The query parameters you can use depend on whether you're requesting a list of the current (or only) versions of objects or a list of object versions.

Limiting a listing of current items

You can use **delimiter**, **marker**, **max-keys**, and **prefix** query parameters, alone or in combination with each other, to limit the list of the current items in a bucket.

delimiter

You use the **delimiter** query parameter to request a bucket listing that includes a list of common prefixes, where a prefix is the name of an item up through the first occurrence of the character string specified by the **delimiter** parameter. Each common prefix is listed only once regardless of the number of items with matching names. The items with matching names are not included elsewhere in the listing.

The returned listing also contains items with names that do not include the character string specified by the **delimiter** parameter. In the listing, all the named items are listed first, followed by all the common prefixes.

For example, the bucket listing returned in response to a **GET** request with the **delimiter=budget_proposals** query parameter contains these items and common prefixes:

Items:

AcctgBestPractices.doc *(current version)*
acctg/
hum_res/
mktg/
mktg/campaign_GoGetEm_expenses.xls *(current version)*
mktg/campaign_LiveIt_expenses.xls
quarterly_rpts/
quarterly_rpts/Q2_2012.ppt
quarterly_rpts/Q3_2012.ppt
quarterly_rpts/Q4_2012.ppt
sales/
sales_quotas_2013.pdf

Common prefixes:

acctg/budget_proposals/
hum_res/budget_proposals/
mktg/budget_proposals/
sales/budget_proposals/

The lists of named items and common prefixes included in a listing are subject to any other criteria specified in the request.

Both named items and common prefixes count toward the maximum number of items that can be included in the bucket listing.

marker

You use the **marker** query parameter to start the returned bucket listing with the item following the first item with a name that starts with the character string specified by the **marker** parameter.

For example, the bucket listing returned in response to a **GET** request with the **marker=quarterly_rpts/** query parameter contains these items:

quarterly_rpts/Q2_2012.ppt
quarterly_rpts/Q3_2012.ppt
quarterly_rpts/Q4_2012.ppt
sales/
sales/budget_proposals/
sales/budget_proposals/BudgProp-2013
sales_quotas_2013.pdf

The **marker** query parameter is useful when more than the requested number of items satisfy the request criteria. In this case, the response body includes the **IsTruncated** element with a value of **true**. You can request the next part of the listing by including the **marker** query parameter in a new request. As the parameter value, you specify either the name of the last item in the returned listing or the last common prefix in the returned listing, whichever is alphanumerically greater.

If the string you specify as the value of the **marker** query parameter is the name of a folder and does not end with a forward slash (/), items that begin with that string followed by a forward slash are omitted from the listing.

max-keys

You use the **max-keys** query parameter to limit the number of items in the returned bucket listing to fewer than one thousand.

For example, the bucket listing returned in response to a **GET** request with the **max-keys=5** query parameter contains these items:

AcctgBestPractices.doc *(current version)*
acctg/
acctg/budget_proposals/
acctg/budget_proposals/BudgProp-2013
hum_res/

prefix

You use the **prefix** query parameter to request a bucket listing that contains only items with names that begin with a specified character string (the prefix) and, if applicable, common prefixes that begin that prefix.

For example, the bucket listing returned in response to a **GET** request with the **prefix=sales** query parameter contains only these items:

sales/
sales/budget_proposals/
sales/budget_proposals/BudgProp-2013
sales_quotas_2013.pdf

Limiting a version listing

You can use **delimiter**, **key-marker**, **max-keys**, **prefix**, and **version-id-marker** query parameters, alone or in combination with each other, to limit the items included in a version listing.

The **delimiter**, **max-keys**, and **prefix** parameters have the same functions as the **delimiter**, **max-keys**, and **prefix** parameters you can use with requests for listings of the current (or only) versions of objects in a bucket. The **key-marker** parameter has the same function as the **marker** parameter you can use with requests for listings of the current (or only) versions of objects. For information on these query parameters, see "Limiting a listing of current items" above.

You use the **version-id-marker** parameter in conjunction with the **key-marker** parameter to start the returned bucket listing with the first item with a name that's equal to the character string specified by the **key-marker** parameter and a version ID that's greater than the value specified by the **version-id-marker** parameter. If no such item exists, the returned bucket listing starts with the item following the first item with a name that starts with the character string specified by the **key-marker** parameter.

For example, the version ID of the first version of the object named mktg/campaign_GoGetEm_expenses.xls is 87288784288321. The bucket listing returned in response to a **GET** request with the **key-marker=mktg/campaign_GoGetEm_expenses.xls** and **version-id-marker=87288784288321** query parameters contains only these items:

```
mktg/campaign_GoGetEm_expenses.xls (second version)
mktg/campaign_LiveIt_expenses.xls
quarterly_rpts/
quarterly_rpts/Q2_2012.ppt
quarterly_rpts/Q3_2012.ppt
quarterly_rpts/Q4_2012.ppt
sales/
sales/budget_proposals/
sales/budget_proposals/BudgProp-2013
sales_quotas_2013.pdf
```

The **key-marker** and **version-id-marker** query parameters are useful when more than the requested number of items satisfy the request criteria. In this case, the response body includes the **IsTruncated** element with a value of **true**. It also includes the **NextKeyMarker** and **NextVersionIdMarker** elements:

- If the alphanumerically last item in the returned listing is a named item, the values of these elements are the name and version ID of that item, respectively.
- If the alphanumerically last item in the returned listing is a common prefix, the value the **NextKeyMarker** element is that prefix, and **NextVersionIdMarker** is an empty element.

To request the next part of the listing, you include the **key-marker** and **version-id-marker** query parameters in a new request. For the value of the **key-marker** parameter, you use the value of the **NextKeyMarker** element from the previously returned listing. For the value of the **version-id-marker** parameter, you use the value of the **NextVersionIdMarker** element from the previously returned listing.

Request line

Depending on whether the bucket name is included in the hostname in the HS3 request, the request line for a request to list the contents of a bucket of a bucket has either of these formats:

- With the bucket name included in the hostname:

  GET /[**?***query-parameters*] **HTTP/1.1**

- With the bucket name following the hostname:

  GET /*bucket-name*[**?***query-parameters*] **HTTP/1.1**

In these formats, *query-parameters* can be none, one, or more of the query parameters described in the table below.

Query parameter	Description
delimiter	Includes a list of common prefixes in the bucket listing, where a prefix is the name of an item up through the first occurrence of a specified character string. The specified character string can contain any valid UTF-8 characters. including white space. When specifying a character string, percent-encode non-ASCII characters and reserved special characters such as ampersands (&), commas (,) and equal signs (=). If a character string contains spaces, enclose it in quotation marks. The **delimiter** parameter name and the character string you specify are both case sensitive. For more information on the **delimiter** parameter, see “delimiter” on page 92.

(Continued)

Query parameter	Description
key-marker	Starts the returned bucket listing with the next item after the first item with a name that starts with a specified character string. The specified character string can contain any valid UTF-8 characters. including white space. When specifying a character string, percent-encode non-ASCII characters and reserved special characters such as ampersands (&), commas (,) and equal signs (=). If a character string contains spaces, enclose it in quotation marks. The **key-marker** parameter name and the character string you specify are both case sensitive. The **key-marker** parameter is ignored if specified in a **GET** request that does not include the **versions** parameter. For more information on the **key-marker** parameter, see "Limiting a version listing" above.
marker	Starts the returned bucket listing with the next item after the first item with a name that starts with a specified character string. The specified character string can contain any valid UTF-8 characters. including white space. When specifying a character string, percent-encode non-ASCII characters and reserved special characters such as ampersands (&), commas (,) and equal signs (=). If a character string contains spaces, enclose it in quotation marks. The **marker** parameter name and the character string you specify are both case sensitive. The **marker** parameter is ignored if specified in a **GET** request that includes the **versions** parameter. For more information on the **marker** parameter, see "marker" on page 93.

(Continued)

Query parameter	Description
max-keys	Specifies the maximum number of items to be included in the returned bucket listing. Valid values are integers in the range zero through 2,147,483,647. If you specify an integer greater than one thousand, the returned bucket listing includes only the first one thousand items that satisfy the request criteria (or fewer if fewer than one thousand items satisfy the criteria). The **max-keys** parameter name is case sensitive. For more information on the **max-keys** parameter, see "max-keys" on page 94.
prefix	Requests that only items with names that begin with a specified character string (the prefix) be included in the returned bucket listing. The specified character string can contain any valid UTF-8 characters. including white space. When specifying a character string, percent-encode non-ASCII characters and reserved special characters such as ampersands (&), commas (,) and equal signs (=). If a character string contains spaces, enclose it in quotation marks. The **prefix** parameter name and the character string you specify are both case sensitive. For more information on the **prefix** parameter, see "prefix" on page 94.
version-id-marker	When used in conjunction with the **key-marker** parameter, starts the returned bucket listing with the first item with a name that's equal to the character string specified by the **key-marker** parameter and a version ID that's greater than a specified value. Valid values are integers greater than or equal to zero. The **version-id-marker** parameter name is case sensitive. For more information on the **version-id-marker** parameter, see "Limiting a version listing" above.
versions	Includes old versions of objects in the listing and provides version information for each listed item. The **versions** parameter name is not case sensitive.

Request headers

The table below describes the headers you can use in a request to list the contents of a bucket.

Request header	Description	Required
Authorization	See "AWS authentication" on page 42.	Yes
Date	See "Common request headers" on page 49.	Date or x-amz-date
Host	See "Common request headers" on page 49.	Yes
x-amz-date	See "Common request headers" on page 49.	x-amz-date or Date
x-hcp-pretty-print	Optionally, requests that the XML response body be formatted for readability. Valid values are: • **true** — Format the XML response body for readability. • **false** — Do not apply any special formatting to the XML response body. The default is **false**. The values **true** and **false** are not case sensitive.	No

Response headers

The table below describes the response headers returned in response to a successful request to list the contents of a bucket.

Response header	Description
Content-Type	Specifies the Internet media type of the response body. For a request to list the contents of a bucket, the value of this header is always **application/xml;charset=UTF-8**.
Date	See "Common response headers" on page 52.
Server	See "Common response headers" on page 52.
Transfer-Encoding	Indicates that HCP could not determine the size of the response body before formulating the response. For a request to list the contents of a bucket, the value of this header is always **chunked**.

Response body

The response body returned in response to a request to list the contents of a bucket differs depending on whether request is for the current (or only) versions of objects or for object versions.

Response body for a listing of current items

In response to a request to list the current (or only) versions of objects in a bucket, HCP returns an XML response body, in this format:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Name>bucket-name</Name>
    <Prefix>prefix</Prefix>
    <Marker>marker</Marker>
    <MaxKeys>maximum-items-to-list</MaxKeys>
    <Delimiter>delimiter</Delimiter>
    <IsTruncated>true|false</IsTruncated>
Format for an item:
    <Contents>
        <Key>item-name</Key>
        <LastModified>date-and-time-last-modified</LastModified>
        <ETag>"etag"</ETag>
        <Size>size-in-bytes</Size>
        <Owner>
            <ID>owner-user-id</ID>
            <DisplayName>owner-username</DisplayName>
        </Owner>
        <StorageClass>STANDARD</StorageClass>
    </Contents>
Format for a common prefix:
    <CommonPrefixes>
        <Prefix>common-prefix</Prefix>
    </CommonPrefixes>
</ListBucketResult>
```

Response body for a version listing

In response to a request to list versions of objects in a bucket, HCP returns an XML response body, in this format:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListVersionsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Name>bucket-name</Name>
    <Prefix>prefix</Prefix>
    <KeyMarker>key-marker</KeyMarker>
    <VersionIdMarker>version-id-marker</VersionIdMarker>
    <NextKeyMarker>name-of-last-item-in-listing</NextKeyMarker>
    <NextVersionIdMarker>version-id-of-last-item-in-listing</NextVersionIdMarker>
    <MaxKeys>maximum-items-to-list</MaxKeys>
    <Delimiter>delimiter</Delimiter>
    <IsTruncated>true|false</IsTruncated>
Format for a version of an item other than a deleted version:
    <Version>
        <Key>item-name</Key>
        <VersionId>version-id</VersionId>
        <IsLatest>true|false</IsLatest>
        <LastModified>date-and-time-last-modified</LastModified>
        <ETag>"etag"</ETag>
        <Size>size-in-bytes</Size>
        <Owner>
            <ID>owner-user-id</ID>
            <DisplayName>owner-username</DisplayName>
        </Owner>
        <StorageClass>STANDARD</StorageClass>
    </Version>
Format for a deleted version of an item:
    <DeleteMarker>
        <Key>item-name</Key>
        <VersionId>version-id</VersionId>
        <IsLatest>true|false</IsLatest>
        <LastModified>date-and-time-last-modified</LastModified>
        <Owner>
            <ID>owner-user-id</ID>
            <DisplayName>owner-username</DisplayName>
        </Owner>
    </DeleteMarker>
Format for a common prefix:
    <CommonPrefixes>
        <Prefix>common-prefix</Prefix>
    </CommonPrefixes>
</ListVersionsResult>
```

Response body elements

The table below describes the XML elements in the two formats of the response body returned in response to a request to list the contents of a bucket. The elements are listed in alphabetical order.

Element	Description
CommonPrefixes	Child of the **ListBucketResult** or **ListVersionsResult** element and container for the **Prefix** element. The response body contains one **CommonPrefixes** element for each common prefix in the bucket listing.
Contents	Child of the **ListBucketResult** element and container for the elements that describe a current item in the bucket. The response body contains one **Contents** element for each item that satisfies the request criteria.
DeleteMarker	Child of the **ListVersionsResult** element and container for the elements that describe a deleted version of an item in the bucket. The response body contains one **DeleteMarker** element for each deleted version that satisfies the request criteria.
Delimiter	Child of the **ListBucketResult** or **ListVersionsResult** element. The **Delimiter** element specifies the value of the **delimiter** query parameter included in the request. If the request did not include the **delimiter** query parameter, the response body does not include the **Marker** element.
DisplayName	Child of the **Owner** element. If the item owner is identified by an HCP user account, the value of the **DisplayName** element is the username for that account. If the item owner is identified by an AD user account, the value of the **DisplayName** element is the username of that account followed by an at sign (@) and the AD domain name. If the item has no owner, the value of the **DisplayName** element is **nobody**. For folders, the value of the **DisplayName** element is always **nobody**.

(Continued)

Element	Description
ETag	Child of the **Contents** or **Version** element. The **ETag** element specifies the ETag of the applicable item or item version. The response body includes **ETag** elements for the listed folders. However, because they have no content, all folders have the same value for this element. For information on ETags, see "ETags" on page 120.
ID	Child of the **Owner** element. If the item owner is identified by an HCP user account, the value of the **ID** element is the user ID for that account. If the item owner is identified by an AD user account, the value of the **ID** element is the SID for that account. If the item has no owner, the value of the **ID** element is **nobody**. For folders, the value of the **ID** element is always **nobody**.
IsLatest	Child of the **Version** or **DeleteMarker** element. The **IsLatest** element indicates whether the item version is the current version of the item. Possible values are: • **true** — The item version is the current version of the item. • **false** — The item version is an old version of the item.
IsTruncated	Child of the **ListBucketResult** or **ListVersionsResult** element. The **IsTruncated** element indicates whether the returned bucket listing is complete. Possible values are: • **true** — The returned listing includes all items that satisfy the request. • **false** — The returned listing does not include all items that satisfy the request.
Key	Child of the **Contents**, **Version**, or **DeleteMarker** element. The **Key** element specifies the name of an item or item version in the bucket listing.

(Continued)

Element	Description
KeyMarker	Child of the **ListVersionsResult** element. The **KeyMarker** element specifies the value of the **key-marker** query parameter included in the request. If the request did not include the **key-marker** query parameter, the response body includes **KeyMarker** as an empty element.
LastModified	Child of the **Contents**, **Version**, or **DeleteMarker** element. The **LastModified** element specifies the date and time at which the applicable item or item version was last modified, in Greenwich Mean Time (GMT). For a deleted version, this is the date and time at which the deleted version was created. The date and time are expressed in this format: *yyyy*-*MM*-*dd*T*HH*:*mm*:*ss*.*SSS*Z For example: 2013-03-18T19:46:03.856Z Modifying an object means modifying its metadata. You cannot modify the content of an object.
ListBucketResult	Root element for a listing of current items.
ListVersionsResult	Root element for a version listing.
Marker	Child of the **ListBucketResult** element. The **Marker** element specifies the value of the **marker** query parameter included in the request. If the request did not include the **marker** query parameter, the response body includes **Marker** as an empty element.
MaxKeys	Child of the **ListBucketResult** or **ListVersionsResult** element. The **MaxKeys** element specifies the value of the **max-keys** query parameter included in the request. If the request did not include the **max-keys** query parameter, the response body includes **MaxKeys** as an empty element.
Name	Child of the **ListBucketResult** or **ListVersionsResult** element. The **Name** element specifies the name of the applicable bucket.

(Continued)

Element	Description
NextKeyMarker	Child of the **ListVersionsResult** element. The **NextKeyMarker** element specifies the name of the last item included in the returned bucket listing. This element is included in the response body only when the value of the **IsTruncated** element is **true**. If the returned listing is truncated, you can use the value of the **NextKeyMarker** element as the value of the **key-marker** query parameter in a new request to retrieve the next set of items that satisfy the request criteria.
NextVersionId Marker	Child of the **ListVersionsResult** element. The **NextVersionIdMarker** element specifies the version ID of the last item included in the returned bucket listing. This element is included in the response body only when the value of the **IsTruncated** element is **true**. If the returned listing is truncated, you can use the value of the **NextVersionIdMarker** element as the value of the **version-id-marker** query parameter in a new request to retrieve the next set of items that satisfy the request criteria.
Owner	Child of the **Contents**, **Version**, or **DeleteMarker** element and container for the **ID** and **DisplayName** elements. The **Owner** element identifies the owner of the applicable item.
Prefix	One of: • Child of the **ListBucketResult** or **ListVersionsResult** element. In this case, the **Prefix** element specifies the value of the **prefix** query parameter included in the request. If the request did not include the **prefix** query parameter, the response body includes **Prefix** as an empty element. • Child of the **CommonPrefixes** element. In this case, the **Prefix** element specifies a common prefix.
Size	Child of the **Contents** or **Version** element. The **Size** element specifies the size, in bytes, of the content of the item or item version. The response body includes **Size** elements for the listed folders. However, because folders have no content, the value of this element for a folder is always **0** (zero).

(Continued)

Element	Description
StorageClass	Child of the **Contents** or **Version** element. The value of the **StorageClass** element is always **STANDARD**.
Version	Child of the **ListVersionsResult** element and container for the elements that describe a version of an item in the bucket. The response body contains one **Version** element for each item version that satisfies the request criteria.
VersionId	Child of the **Version** or **DeleteMarker** element. The **VersionId** element specifies the version ID of an item version.
VersionIdMarker	Child of the **ListVersionsResult** element. The **VersionIdMarker** element specifies the value of the **version-id-marker** query parameter included in the request. If the request did not include the **version-id-marker** query parameter, the response body includes **VersionIdMarker** as an empty element.

HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to list the contents of a bucket. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 53.

Code	Meaning	Description
200	OK	HCP successfully retrieved the request bucket listing.
400	Bad Request	The **max-keys** query parameter specifies an invalid value.
403	Forbidden	One of: • The credentials provided with the request are invalid. • You are not the bucket owner. • The HS3 API is currently disabled for the specified bucket. • The tenant does not currently support use of the HS3 API.

(Continued)

Code	Meaning	Description
404	Not Found	The specified bucket does not exist.
500	Internal Server Error	An internal error occurred. Try the request again, gradually increasing the delay between each successive attempt. If this error persists, 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. If this error persists, contact your tenant administrator.

Examples

The following sections show sample **GET** requests for listing bucket contents.

Example 1: Listing the items in a folder

Here's a sample **GET** request for a bucket listing that lists the objects that are in the mktg folder and, separately, the subfolders that are in the mktg folder. The request uses these query parameters:

- **prefix=mktg/** — Lists only items that start with mktg/
- **marker=mktg/** — Starts the listing with the item that follows mktg/ by itself
- **delimiter=/** — Treats items that have a forward slash (/) after mktg/ as having a common prefix

Request with s3curl command line

```
./s3curl.pl --id=lgreen -- -k "https://finance.europe.hcp.example.com?prefix=mktg/
    &marker=mktg/&delimiter=/" -H "x-hcp-pretty-print:true"
```

Request headers

```
GET /?prefix=mktg/&delimiter=/&marker=mktg/ HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Fri, 29 Mar 2013 15:51:25 +0000
Authorization: AWS bGdyZWVu:ysiVYJ/cOr78z69BuDy6fy07ryo=
x-hcp-pretty-print: true
```

Response headers

```
HTTP/1.1 200 OK
Date: Fri, 29 Mar 2013 15:51:25 GMT
Server: HCP V6.0.1.117
Content-Type: application/xml;charset=UTF-8
Transfer-Encoding: chunked
```

Response body

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Name>finance</Name>
    <Prefix>mktg/</Prefix>
    <Marker>mktg/</Marker>
    <MaxKeys>1000</MaxKeys>
    <Delimiter>/</Delimiter>
    <IsTruncated>false</IsTruncated>
    <Contents>
        <Key>mktg/campaign_GoGetEm_expenses.xls</Key>
        <LastModified>2013-02-13T17:44:53.000Z</LastModified>
        <ETag>"6ed7faad1e0661c03ad65a4317d4a94c"</ETag>
        <Size>94328</Size>
        <Owner>
            <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
            <DisplayName>lgreen</DisplayName>
        </Owner>
        <StorageClass>STANDARD</StorageClass>
    </Contents>
    <Contents>
        <Key>mktg/campaign_LiveIt_expenses.xls</Key>
        <LastModified>2012-11-05T14:32:29.110Z</LastModified>
        <ETag>"7ad452af1e2f61b33a865c4362be5921"</ETag>
        <Size>81578</Size>
        <Owner>
            <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
            <DisplayName>lgreen</DisplayName>
        </Owner>
        <StorageClass>STANDARD</StorageClass>
    </Contents>
    <CommonPrefixes>
```

```
        <Prefix>mktg/budget_proposals/</Prefix>
    </CommonPrefixes>
</ListBucketResult>
```

Example 2: Listing items a few at a time

Here are two sample **GET** requests for bucket listings that list item versions. The first request returns a list of three items. The second request returns a list of the three items that follow the last item returned by the first request.

Request for the first three items

The request for the first three items uses these query parameters:

- **versions** — Lists item versions
- **key-marker=mktg/** — Starts the listing with the item that follows mktg/ by itself
- **max-keys=3** — Lists at most three items

Request with s3curl command line

```
./s3curl.pl --id=lgreen -- -k "https://finance.europe.hcp.example.com?versions
    &key-marker=mktg/&max-keys=3" -H "x-hcp-pretty-print:true"
```

Request headers

```
GET /?versions&key-marker=mktg/&max-keys=3 HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Fri, 29 Mar 2013 17:15:55 +0000
Authorization: AWS bGdyZWVu:sf4WfS+TzOj9zrHVRA5Z0i1KsFg=
x-hcp-pretty-print: true
```

Response headers

```
HTTP/1.1 200 OK
Date: Fri, 29 Mar 2013 17:15:54 GMT
Server: HCP V6.0.1.117
Content-Type: application/xml;charset=UTF-8
Transfer-Encoding: chunked
```

Response body

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListVersionsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Name>finance</Name>
    <Prefix></Prefix>
    <KeyMarker>mktg/</KeyMarker>
    <VersionIdMarker></VersionIdMarker>
    <NextKeyMarker>mktg/campaign_GoGetEm_expenses.xls</NextKeyMarker>
    <NextVersionIdMarker>87288784288321</NextVersionIdMarker>
    <MaxKeys>3</MaxKeys>
    <IsTruncated>true</IsTruncated>
    <Version>
        <Key>mktg/budget_proposals/</Key>
        <VersionId>87288779571521</VersionId>
        <IsLatest>true</IsLatest>
        <LastModified>2012-11-13T17:12:00.805Z</LastModified>
        <ETag>"d41d8cd98f00b204e9800998ecf8427e"</ETag>
        <Size>0</Size>
        <Owner>
            <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
            <DisplayName>lgreen</DisplayName>
        </Owner>
        <StorageClass>STANDARD</StorageClass>
    </Version>
    <Version>
        <Key>mktg/budget_proposals/BudgProp-2013</Key>
        <VersionId>87288779576769</VersionId>
        <IsLatest>true</IsLatest>
        <LastModified>2012-11-13T17:12:01.218Z</LastModified>
        <ETag>"af65fc4d3e90617b3ad65a83a946be11"</ETag>
        <Size>124591</Size>
        <Owner>
            <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
            <DisplayName>lgreen</DisplayName>
        </Owner>
        <StorageClass>STANDARD</StorageClass>
    </Version>
    <Version>
        <Key>mktg/campaign_GoGetEm_expenses.xls</Key>
        <VersionId>87288784288321</VersionId>
        <IsLatest>false</IsLatest>
        <LastModified>2013-02-12T20:14:06.519Z</LastModified>
        <ETag>"74d824cd5076a1361da128ee18e5a42b"</ETag>
        <Size>92127</Size>
        <Owner>
            <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
            <DisplayName>lgreen</DisplayName>
        </Owner>
        <StorageClass>STANDARD</StorageClass>
    </Version>
</ListVersionsResult>
```

Request for next three items

The request for the next three items uses these query parameters:

- **versions** — Lists item versions.
- **key-marker=mktg/campaign_GoGetEm_expenses.xls** and **version-id-marker=87288784288321** — Starts the listing with the item that follows version 87288784288321 of mktg/campaign_GoGetEm_expenses.xls. The version ID and item name are the values of the **NextVersionIdMarker** and **NextKeyMarker** elements from the previous response body, respectively.
- **max-keys=3** — Lists at most three items.

Request with s3curl command line

```
./s3curl.pl  --id=lgreen  --  -k  "https://finance.europe.hcp.example.com?versions
     &key-marker=mktg/campaign_GoGetEm_expenses.xls
     &version-id-marker=87288784288321&max-keys=3"
     -H  "x-hcp-pretty-print:true"
```

Request headers

```
GET /?versions&key-marker=mktg/campaign_GoGetEm_expenses.xls
     &version-id-marker=87288784288321&max-keys=3 HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Fri, 29 Mar 2013 18:03:33 +0000
Authorization: AWS bGdyZWVu:eYSJshWGcAf7i51jbpl/ZQqqJYM=
x-hcp-pretty-print: true
```

Response headers

```
HTTP/1.1 200 OK
Date: Fri, 29 Mar 2013 18:03:33 GMT
Server: HCP V6.0.1.117
Content-Type: application/xml;charset=UTF-8
Transfer-Encoding: chunked
```

Response body

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListVersionsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
     <Name>finance</Name>
     <Prefix></Prefix>
     <KeyMarker>mktg/campaign_GoGetEm_expenses.xls</KeyMarker>
     <VersionIdMarker>87288784288321</VersionIdMarker>
     <NextKeyMarker>quarterly_rpts/</NextKeyMarker>
     <NextVersionIdMarker>87288727467201</NextVersionIdMarker>
```

```
        <MaxKeys>3</MaxKeys>
        <IsTruncated>true</IsTruncated>
        <Version>
            <Key>mktg/campaign_GoGetEm_expenses.xls</Key>
            <VersionId>87288825190337</VersionId>
            <IsLatest>true</IsLatest>
            <LastModified>2013-02-13T17:44:53.000Z</LastModified>
            <ETag>"6ed7faad1e0661c03ad65a4317d4a94c"</ETag>
            <Size>94328</Size>
            <Owner>
                <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
                <DisplayName>lgreen</DisplayName>
            </Owner>
            <StorageClass>STANDARD</StorageClass>
        </Version>
        <Version>
            <Key>mktg/campaign_LiveIt_expenses.xls</Key>
            <VersionId>87288785222273</VersionId>
            <IsLatest>true</IsLatest>
            <LastModified>2012-11-05T14:32:29.110Z</LastModified>
            <ETag>"7ad452af1e2f61b33a865c4362be5921"</ETag>
            <Size>81578</Size>
            <Owner>
                <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
                <DisplayName>lgreen</DisplayName>
            </Owner>
            <StorageClass>STANDARD</StorageClass>
        </Version>
        <Version>
            <Key>quarterly_rpts/</Key>
            <VersionId>87288727467201</VersionId>
            <IsLatest>true</IsLatest>
            <LastModified>2012-07-23T18:26:24.675Z</LastModified>
            <ETag>"d41d8cd98f00b204e9800998ecf8427e"</ETag>
            <Size>0</Size>
            <Owner>
                <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
                <DisplayName>lgreen</DisplayName>
            </Owner>
            <StorageClass>STANDARD</StorageClass>
        </Version>
    </ListVersionsResult>
```

Example 3: Listing the versions of an individual object

Here's a sample **GET** request for a bucket listing that lists all the versions of the object named AcctgBestPractices.doc, including a deleted version. The request uses these query parameters:

- **versions** — Lists item versions
- **prefix=AcctgBestPractices.doc** — Lists only items that start with AcctgBestPractices.doc

Request with s3curl command line

```
./s3curl.pl  --id=lgreen  --  -k  "https://finance.europe.hcp.example.com?versions
    &prefix=AcctgBestPractices.doc"  -H  "x-hcp-pretty-print:true"
```

Request headers

```
GET /?versions&prefix=AcctgBestPractices.doc HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Fri, 29 Mar 2013 18:37:48 +0000
Authorization: AWS bGdyZWVu:zkfUJoEb+pOrtVpxxOHeamoD4cI=
x-hcp-pretty-print: true
```

Response headers

```
HTTP/1.1 200 OK
Date: Fri, 29 Mar 2013 18:37:48 GMT
Server: HCP V6.0.1.117
Content-Type: application/xml;charset=UTF-8
Transfer-Encoding: chunked
```

Response body

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListVersionsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Name>finance</Name>
    <Prefix>AcctgBestPractices.doc</Prefix>
    <KeyMarker></KeyMarker>
    <VersionIdMarker></VersionIdMarker>
    <MaxKeys>1000</MaxKeys>
    <IsTruncated>false</IsTruncated>
    <Version>
        <Key>AcctgBestPractices.doc</Key>
        <VersionId>87288758401473</VersionId>
        <IsLatest>false</IsLatest>
        <LastModified>2012-05-19T14:56:05.630Z</LastModified>
        <ETag>"26aa5129552e57fc64e10aa5b3911ee2"</ETag>
        <Size>3206178</Size>
        <Owner>
            <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
            <DisplayName>lgreen</DisplayName>
        </Owner>
        <StorageClass>STANDARD</StorageClass>
    </Version>
    <Version>
        <Key>AcctgBestPractices.doc</Key>
        <VersionId>87288800665537</VersionId>
        <IsLatest>false</IsLatest>
        <LastModified>2012-10-08T19:23:31.305Z</LastModified>
        <ETag>"b7235e841a2cc45a7e42a8a576d493b1"</ETag>
        <Size>3374982</Size>
        <Owner>
            <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
            <DisplayName>lgreen</DisplayName>
        </Owner>
        <StorageClass>STANDARD</StorageClass>
    </Version>
    <Version>
        <Key>AcctgBestPractices.doc</Key>
        <VersionId>87288808614529</VersionId>
        <IsLatest>false</IsLatest>
        <LastModified>2012-12-18T21:06:52.011Z</LastModified>
        <ETag>"5ab7542f753b09fdb73141a66c134b9"</ETag>
        <Size>3557448</Size>
        <Owner>
            <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
            <DisplayName>lgreen</DisplayName>
        </Owner>
        <StorageClass>STANDARD</StorageClass>
    </Version>
    <DeleteMarker>
        <Key>AcctgBestPractices.doc</Key>
        <VersionId>87288810855745</VersionId>
        <IsLatest>false</IsLatest>
        <LastModified>2012-12-20T13:10:04.902Z</LastModified>
        <Owner>
            <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
            <DisplayName>lgreen</DisplayName>
        </Owner>
    </DeleteMarker>
    <Version>
        <Key>AcctgBestPractices.doc</Key>
        <VersionId>87288815588289</VersionId>
        <IsLatest>true</IsLatest>
        <LastModified>2012-10-20T19:42:16.331Z</LastModified>
        <ETag>"764f38262c6e581f678e1ac9b0211ae8"</ETag>
```

```
            <Size>3552369</Size>
            <Owner>
                <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
                <DisplayName>lgreen</DisplayName>
            </Owner>
            <StorageClass>STANDARD</StorageClass>
        </Version>
    </ListVersionsResult>
```

Deleting a bucket

You use the HTTP **DELETE** method to delete a bucket. To delete a bucket, you need to be the bucket owner.

You can delete a bucket only while it's empty. If you try to delete a bucket that contains any objects, HCP returns a 409 (Conflict) status code and does not delete the bucket.

Request line

Depending on whether the bucket name is included in the hostname in the HS3 request, the request line for a request to delete a bucket has either of these formats:

- With the bucket name included in the hostname:

  `DELETE / HTTP/1.1`

- With the bucket name following the hostname:

  `DELETE` */bucket-name* **`HTTP/1.1`**

Request headers

The table below describes the headers you can use in a request to delete a bucket.

Request header	Description	Required
Authorization	See "AWS authentication" on page 42.	Yes
Date	See "Common request headers" on page 49.	Date or x-amz-date
Host	See "Common request headers" on page 49.	Yes
x-amz-date	See "Common request headers" on page 49.	x-amz-date or Date

Response headers

The table below describes the response headers returned in response to a successful request to delete a bucket.

Response header	Description
Date	See “Common response headers” on page 52.
Server	See “Common response headers” on page 52.

HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to delete a bucket. For more information on HTTP status codes and the error codes that can accompany them, see “Error codes” on page 53.

Code	Meaning	Description
204	No Content	HCP successfully deleted the bucket.
403	Forbidden	One of: • The credentials provided with the request are invalid. • You are not the bucket owner. • The HS3 API is currently disabled for the specified bucket. • The tenant does not currently support use of the HS3 API.
404	Not Found	The specified bucket does not exist.
409	Conflict	The specified bucket is not empty.
500	Internal Server Error	An internal error occurred. Try the request again, gradually increasing the delay between each successive attempt. If this error persists, contact your tenant administrator.

(Continued)

Code	Meaning	Description
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. If this error persists, contact your tenant administrator.

Example: Deleting a bucket

Here's a sample **DELETE** request that deletes the bucket named sales-mktg.

Request with s3curl command line

```
./s3curl.pl --id=lgreen --delete -- -k "https://sales-mktg.europe.hcp.example.com"
```

Request headers

```
DELETE / HTTP/1.1
Host: sales-mktg.europe.hcp.example.com
Date: Wed, 20 Mar 2013 21:50:17 +0000
Authorization: AWS bGdyZWVu:2nVpI9dSOakB9JZtEyx81RCZTKw=
```

Response headers

```
HTTP/1.1 204 No Content
Date: Wed, 20 Mar 2013 21:50:17 GMT
Server: HCP V6.0.1.117
```

6

Working with objects

With the HS3 API, you can perform operations on individual objects.

For each object operation you can request, this chapter:

- Describes the operation
- Shows the request line for the operation
- Describes the request headers for the operation
- Describes the response headers returned for a successful execution of the requested operation
- Explains the HTTP status codes that can be returned in response to requests for the operation
- Presents one or more examples of requests for the operation

For general information about HS3 requests and HCP responses to those requests, see Chapter 4, "Requests and responses," on page 47.

For information about the examples in this chapter, see "Examples in this book" on page 12.

Storing an object

You use the HTTP **PUT** method to store an object in a bucket. To store an object, you need write permission for the bucket.

For a request to store an object, the request body consists of the data in a specified file. This data becomes the object content.

When you store an object, you specify a name for it. The object name does not need to be the same as the name of the file containing the original data. For information on naming objects, see "Object names" on page 14.

If versioning is enabled and you try to store an object with the same name as an existing object, HCP creates a new version of the object. If versioning is disabled and you try to store an object with the same name as an existing object, HCP returns a 409 (Conflict) status code and does not store the object. For information on versioning, see "Versioning" on page 32.

You can add custom metadata to an object in the same request as you use to store the object. To do this, you use x-amz-meta- headers. For information on custom metadata, see "Custom metadata" on page 18.

You can specify an ACL for an object in the same request as you use to store the object. To do this, you need to use ACL headers. You cannot use an ACL request body when storing an object. For information on ACLs, see "Access control lists" on page 22.

If the ACL you specify in a request to store an object is invalid, HCP returns a 400 (Bad Request) or 501 (Not Implemented) status code and does not store the object.

If you're an authenticated user, when you store an object, you become the object owner. If you're accessing the bucket anonymously, the new object has no owner. For information on object ownership, see "Object owners" on page 22.

ETags

When you store an object, HCP returns the **ETag** for the object in the ETag response header. An ETag is an identifier for the content of an object. ETags are useful for making object-level operations conditional based on the object content. Operations that can be made conditional are checking the existence of an object (page 131), copying an object (page 152), and retrieving an object (page 166).

Ensuring object integrity

When you store an object, you can use the Content-MD5 request header to ensure the integrity of the object data. The value you specify for this header must be the Base64-encoded MD5 hash of the original file data.

When you include the Content-MD5 header in a request to store an object, HCP calculates the Base64-encoded MD5 hash of the data it receives and compares that to the header value. If the values don't match, HCP returns a 400 (Bad Request) status code and does not store the object.

Object encryption

When you store an object, you can use the x-amz-server-side-encryption request header to determine whether objects stored in HCP are encrypted. If stored objects are encrypted, the response headers include x-amz-server-side-encryption with a value representing the encryption algorithm and key length HCP is using. If stored objects are not encrypted, the value of the x-amz-server-side-encryption response header is **None**.

Saving network bandwidth

You can use the Expect request header in a request to store an object to tell the application not to send the request body (the data) to HCP if the request headers are rejected. This prevents unnecessary network bandwidth usage.

Request line

Depending on whether the bucket name is included in the hostname in the HS3 request, the request line for a request to store an object has either of these formats:

- With the bucket name included in the hostname:

  `PUT` *`/object-name`* **`HTTP/1.1`**

- With the bucket name following the hostname:

  `PUT` *`/bucket-name/object-name`* **`HTTP/1.1`**

Request headers

The table below describes the headers you can use in a request to store an object.

Request header	Description	Required
Authorization	See “AWS authentication” on page 42.	Yes
Content-Length	Specifies the size, in bytes, of the data being stored.	Yes
Content-MD5	Directs HCP to check the integrity of the data it receives by comparing a Base64-encoded MD5 hash of that data to the value specified by this header. The valid value for this header is the Base64-encoded MD5 hash of the data in the request body.	No
Content-Type	Specifies the Internet media type of the request body. Valid values are Internet media types (for example, **text/plain**, **application/xml**, or **image/jpeg**).	No
Date	See “Common request headers” on page 49.	Date or x-amz-date
Expect	Tells the application not to send the request body if the request headers are rejected. The only valid value is **100-continue**. This value is not case sensitive.	No
Host	See “Common request headers” on page 49.	Yes
x-amz-acl	Adds a canned ACL to the object. This header is required for adding a canned ACL to an object. If you’re using individual x-amz-grant- headers to add the ACL, the x-amz-acl header is invalid. For valid values for this header, see “Canned ACLs” on page 25.	No
x-amz-date	See “Common request headers” on page 49.	x-amz-date or Date
x-amz-grant-full-control	Grants full control over the object to one or more specified grantees. If you’re using a canned ACL to add an ACL to an object, the x-amz-grant-full-control header is invalid. For valid values for this and the following x-amz-grant- headers, see “Using individual grant headers” on page 26.	No
x-amz-grant-read	Grants the browse and read data access permissions for the object to one or more specified grantees. If you’re using a canned ACL to add an ACL to an object, the x-amz-grant-read header is invalid.	No

(Continued)

Request header	Description	Required
x-amz-grant-read-acp	Grants the read ACL data access permission for the object to one or more specified grantees. If you're using a canned ACL to add an ACL to an object, the x-amz-grant-read-acp header is invalid.	No
x-amz-grant-write	Grants the write and delete data access permissions for the object to one or more specified grantees. If you're using a canned ACL to add an ACL to an object, the x-amz-grant-write header is invalid.	No
x-amz-grant-write-acp	Grants the write ACL data access permission for the object to one or more specified grantees. If you're using a canned ACL to add an ACL to an object, the x-amz-grant-write-acp header is invalid.	No
x-amz-meta-	Adds custom metadata to the object. For information on using this header, see "Storing custom metadata with HS3" on page 18.	No
x-amz-server-side-encryption	Requests that the response headers include x-amz-server-side-encryption, which indicates whether objects stored in HCP are encrypted. Valid values consist of the name of an encryption algorithm concatenated with an encryption key length (for example, **AES256**).	No

Response headers

The table below describes the response headers returned in response to a successful request to store an object.

Response header	Description
Content-Length	Specifies the size, in bytes, of the response body. In response to a successful request to store an object, the value of this header is always **0** (zero).
Date	See "Common response headers" on page 52.
ETag	Specifies the ETag for the object.
Server	See "Common response headers" on page 52.

(Continued)

Response header	Description
x-amz-server-side-encryption	Specifies whether objects stored in HCP are encrypted. Possible values are: • If objects are encrypted, the name of the encryption algorithm concatenated with the encryption key length (for example, **AES256**) • If objects are not encrypted, **None** This header is returned only if the request headers include x-amz-server-side-encryption.
x-amz-version-id	Specifies the version ID of the object. This header is returned only while versioning is enabled.

HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to store an object. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 53.

Code	Meaning	Description
200	OK	HCP successfully stored the object.
400	Bad Request	One of: • An ACL grant header specifies an invalid grantee. • The value specified by the Content-MD5 header does not match the Base64-encoded MD5 hash of the data HCP received.
403	Forbidden	One of: • The credentials provided with the request are invalid. • You do not have permission to store objects in the specified bucket. • The HS3 API is currently disabled for the specified bucket.
404	Not Found	The specified bucket does not exist.

(Continued)

Code	Meaning	Description
409	Conflict	One of: • An object with the specified name already exists in the specified bucket, and versioning is disabled for that bucket. • A folder with the specified name already exists in the specified bucket.
413	Request Entity Too Large	The object you are trying to store is too big for the amount of space left in the bucket.
500	Internal Server Error	An internal error occurred. Try the request again, gradually increasing the delay between each successive attempt. If this error persists, contact your tenant administrator.
501	Not Implemented	The request includes the x-amz-acl header with an invalid value.
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. If this error persists, contact your tenant administrator.

Examples

The following sections show sample **PUT** requests for storing objects.

Example 1: Storing an object

Here's a sample **PUT** request that stores an object named quarterly_rpts/Q4_2012.ppt in the finance bucket. The example shows the response headers HCP returns while versioning is enabled for the bucket and while versioning is disabled for the bucket.

Request with s3curl command line

```
./s3curl.pl  --id=lgreen  --put=/quarterly_rpts/Q4_2012.ppt  --  -k
    "https://finance.europe.hcp.example.com/quarterly_rpts/Q4_2012.ppt"
```

Request headers

```
PUT /quarterly_rpts/Q4_2012.ppt HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Mon, 22 Jan 2013 17:19:26 +0000
Authorization: AWS bGdyZWVu:HbkRBWvyO3YQO55Bm0VS3RMatvg=
Content-Length: 235813
```

Response headers with versioning enabled

```
HTTP/1.1 200 OK
Date: Mon, 22 Jan 2013 17:19:26 GMT
Server: HCP V6.0.1.117
x-amz-version-id: 87288727469825
ETag: "617e8ef649d40cda1f7f3ca81c97a06a"
Content-Length: 0
```

Response headers with versioning disabled

```
HTTP/1.1 200 OK
Date: Mon, 22 Jan 2013 17:19:26 GMT
Server: HCP V6.0.1.117
ETag: "617e8ef649d40cda1f7f3ca81c97a06a"
Content-Length: 0
```

Example 2: Storing an object with custom metadata and an ACL

Here's a sample **PUT** request that stores an object named hum_res/budget_proposals/BudgProp-2013 in the finance bucket, for which versioning is disabled. The object is stored with custom metadata specified by three x-amz-meta- headers and a canned ACL specified by the x-amz-acl header.

Request with s3curl command line

```
./s3curl.pl  --id=lgreen  --put=/hum_res/budget_proposals/BudgProp-2013  --  -k
    "https://finance.europe.hcp.example.com/hum_res/budget_proposals/
    BudgProp-2013"  -H  "x-amz-meta-author:P.D. Grey"
    -H  "x-amz-meta-author:Morgan White"  -H  "x-amz-meta-auther:Paris Black"
    -H  "x-amz-acl:authenticated-read"
```

Request headers

```
PUT /hum_res/budget_proposals/BudgProp-2013 HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Fri, 16 Nov 2012 23:29:17 +0000
Authorization: AWS bGdyZWVu:FiWxioJHDGMLyP0WkCuqUpRqr2w=
x-amz-meta-author: P .D. Grey
x-amz-meta-author: Morgan White
x-amz-meta-author: Paris Black
x-amz-acl: authenticated-read
Content-Length: 881932
```

Response headers

```
HTTP/1.1 200 OK
Date: Fri, 16 Nov 2012 23:29:17 GMT
Server: HCP V6.0.1.117
ETag: "76216527ff2f6219f7c29251a619c8db"
Content-Length: 0
```

Creating a folder

You use the HTTP **PUT** method to create a folder in a bucket. To create a folder, you need write permission for the bucket.

To tell HCP to create a folder instead of an object in response to a **PUT** request, you can do either these:

- Include a forward slash (/) or the percent-encoded equivalent (%2F) after the folder name in the request.
- Include the Content-Type header in the request with a value of **x-directory**.

In either case, you also need to include the Content-Length header in the request. However, HCP ignores any request body.

When you create a folder, you specify a name for it. The rules for folder names are the same as the rules for object names. For more information on object names, see “Object names” on page 14.

Request line

Depending on whether the bucket name is included in the hostname in the HS3 request, the request line for a request to create a folder has either of these formats:

- With the bucket name included in the hostname:

  PUT /*folder-name*[(/|**%2F**)] **HTTP/1.1**

- With the bucket name following the hostname:

  PUT /*bucket-name*/*folder-name*[(/|**%2F**)] **HTTP/1.1**

Request headers

The table below describes the headers you can use in a request to create a folder.

Request header	Description	Required
Authorization	See "AWS authentication" on page 42.	Yes
Content-Length	Specifies the size, in bytes, of the data being stored. In a request to create a folder, the value of this header should be **0** (zero). Even if you specify a value greater than zero for this header, no request body is sent to HCP.	Yes
Date	See "Common request headers" on page 49.	Date or x-amz-date
Content-Type	Specifies the Internet media type of the request body. The only valid value is **x-directory**. This header is ignored if the folder name in the request ends with a forward slash (/) or the percent-encoded equivalent (%2F). Otherwise, this header is required.	No
Host	See "Common request headers" on page 49.	Yes
Transfer-Encoding	Specifies the encoding transformation for the request body. The only valid value is **chunked**. Even if you include this header in the request, no request body is sent to HCP.	No
x-amz-date	See "Common request headers" on page 49.	x-amz-date or Date

Response headers

The table below describes the response headers returned in response to a successful request to create a folder.

Response header	Description
Content-Length	Specifies the size, in bytes, of the response body. In response to a successful request to create a folder, the value of this header is always **0** (zero).
Date	See "Common response headers" on page 52.
ETag	Specifies the ETag for the folder. Even though a folder has no content, HCP returns an ETag for it. However, because folders have no content, all folders have the same ETag.
Server	See "Common response headers" on page 52.
x-amz-version-id	Specifies the version ID of the folder. This header is returned only while versioning is enabled.

HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to create a folder. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 53.

Code	Meaning	Description
200	OK	HCP successfully stored the object.
400	Bad Request	The request does not include a Content-Length header.
403	Forbidden	One of: • The credentials provided with the request are invalid. • You do not have permission to create folders in the specified bucket. • The HS3 API is currently disabled for the specified bucket.
404	Not Found	The specified bucket does not exist.
409	Conflict	An object or folder with the specified name already exists.

(Continued)

Code	Meaning	Description
500	Internal Server Error	An internal error occurred. Try the request again, gradually increasing the delay between each successive attempt. If this error persists, 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. If this error persists, contact your tenant administrator.

Example: Creating a folder

Here's a sample **PUT** request that creates a folder named budget_proposals in the r&d folder in the finance bucket. If the r&d folder doesn't already exist, this request also creates that folder. The request is being made while versioning is enabled for the bucket.

Request with s3curl command line

```
./s3curl.pl --id=lgreen --put ~ -- -k
    "https://finance.europe.hcp.example.com/r&d/budget_proposals%2F"
    -H "Content-Length:0"
```

Request headers

```
PUT /r&d/budget_proposals%2F HTTP/1.1
Host: finance.europe.hcp.example.com
Transfer-Encoding: chunked
Date: Tue, 04 Dec 2012 14:40:27 +0000
Authorization: AWS bGdyZWVu:VsSmWFRWwTPWPjH+QEs6Z2Qwf84=
Content-Length: 0
```

Response headers

```
HTTP/1.1 200 OK
Date: Tue, 04 Dec 2012 14:40:26 GMT
Server: HCP V6.0.1.117
x-amz-version-id: 87395113368065
ETag: "d41d8cd98f00b204e9800998ecf8427e"
Content-Length: 0
```

Checking the existence of an object or folder

You use the HTTP **HEAD** method to check the existence of an objector folder in a bucket. To check the existence of an object, you need read permission for the bucket containing the object or for the object itself. To check the existence of a folder, you need read permission for the bucket.

In response to a request to check the existence of an object or folder, HCP returns a 200 (OK) status code if the object or folder exists and a 404 (Not Found) status code if the object or folder doesn't exist. If you don't have read permission for the bucket or object, HCP returns a 403 (Forbidden) status code.

By default, a **HEAD** request to check the existence of an object checks the existence of the current version of the object. However, while versioning is enabled for the target bucket, you can use the **versionId** query parameter to check the existence of a specific version of an object. If the version identified by the **versionId** parameter does not exist or is a deleted version, HCP returns a 404 (Not Found) status code.

Object information

If the status code returned in response to a request to check the existence of an object, object version, or folder is 200 (OK), the response headers include this information:

- The ETag for the specified item (ETag header). For information on ETags, see "ETags" on page 120.
- While versioning is enabled for the target bucket, the version ID of the specified item (x-amz-version-id header).
- The date the specified item was last modified (Last-Modified header).
- The Internet media type of the specified item (Content-Type header).
- The size of the specified item, in bytes (and Content-Length header).
- For an object or object version, any applicable custom metadata stored in the .metapairs annotation for that item (x-amz-meta- headers). For information on x-amz-meta- headers, see "Custom metadata" on page 18.

Conditionally checking the existence of an object, object version, or folder

You can use the If-Match, If-None-Match, If-Modified-Since, and If-Unmodified-Since request headers to make **HEAD** requests conditional:

- The If-Match and If-None-Match headers compare the ETag for the specified item to one or more values that you specify. Typically, each value is the ETag for an object or object version of interest.

- The If-Modified-Since and If-Unmodified-Since headers compare the date and time the specified item was last modified to a date and time that you specify. The comparison is at the level of seconds, so you cannot use these headers to differentiate between changes that happened at different milliseconds within the same second.

If the specified item meets all the conditions specified by the conditional headers included in the request, HCP returns a 200 (OK) status code. If the specified item does not meet the condition specified by:

- An If-Match or If-Unmodified-Since header, HCP returns a 412 (Precondition Failed) status code

- An If-None-Match or If-Modified-Since header, HCP returns a 304 (Not Modified) status code

If a request includes multiple different conditional headers, HCP processes any If-Match and If-None-Match headers before any If-Modified-Since or If-Unmodified-Since headers. If a request includes more than one of any given header, HCP processes only the first one of those headers and ignores the others.

Request line

Depending on whether the bucket name is included in the hostname in the HS3 request, the request line for a request to check the existence of an object, object version, or folder has either of these formats:

- With the bucket name included in the hostname:

```
HEAD /(object-name[(?versionId=version-id])|(folder-name(/|%2F))
  HTTP/1.1
```

- With the bucket name following the hostname:

```
HEAD /bucket-name/(object-name[(?versionId=version-id])|
  (folder-name(/|%2F)) HTTP/1.1
```

Valid values for *version-id* are integers greater than or equal to zero. Normally, the value of *version-id* is the version ID of a version of the object specified in the request.

The versionId query parameter is not case sensitive.

If you include the **versionId** query parameter in a **HEAD** request while versioning is disabled, the parameter is ignored.

Request headers

The table below describes the headers you can use in a request to check the existence of an object, object version, or folder.

Request header	Description	Required
Authorization	See "AWS authentication" on page 42.	Yes
Date	See "Common request headers" on page 49.	Date or x-amz-date
Host	See "Common request headers" on page 49.	Yes
If-Match	Specifies one or more values for comparison with the ETag of the specified item. If the ETag matches one of the specified values, HCP continues processing the request. If the ETag doesn't match any of the specified values, HCP returns a 412 (Precondition Failed) status code. To specify the values for this header, use this format: `"value"[, "value"]...` In this format, each value can be any string of one or more characters and must be enclosed in double quotation marks ("). Alternatively, you can specify a single asterisk (*) as the value for the If-Match header. All ETags match an asterisk in an If-Match header.	No

(Continued)

Request header	Description	Required
If-Modified-Since	Specifies a date and time, in Greenwich Mean Time (GMT), for comparison with the date and time the specified item was last modified. If the modification date and time is later than the specified date and time, HCP continues processing the request. If the modification date and time is equal to or earlier than the specified date and time, HCP returns a 304 (Not Modified) status code. To specify the date and time for this header, use one of these formats: • *DDD*, *dd* *MMM* **yyyy** *HH*:*mm*:*ss* **+0000|GMT** For example: Thu, 07 Mar 2013 14:27:05 +0000 • *DDDD*, *dd*-*MMM*-*yy* *HH*:*mm*:*ss* **+0000|GMT** For example: Thursday, 07-Mar-13 14:27:05 +0000 • *DDD MMM d HH*:*mm*:*ss yyyy* For example: Thu Mar 7 14:27:05 2013 In these formats: • *DDD* is the three-letter abbreviation for the day of the week, with an uppercase first letter (for example, Mon). • *DDDD* is the day of the week fully spelled out, with an uppercase first letter (for example, Monday). • *d* is the one- or two-digit day of the month, as applicable. • *dd* is the two-digit day of the month. • *MMM* is the three-letter abbreviation for the month, with an uppercase first letter (for example, Feb). • *yy* is the last two digits of the year. HCP assumes that the intended year is within 80 years before or 20 years after the current year. • *yyyy* is the four-digit year. • *HH* is the hour on a 24-hour clock. • *mm* is the number of minutes into the hour. • *ss* is the number of seconds into the minute. If the value specified by the If-Modified-Since header doesn't conform to one of the formats, shown above, the header is ignored.	No

(Continued)

Request header	Description	Required
If-None-Match	Specifies one or more values for comparison with the ETag of the specified item. If the ETag doesn't match any of the specified values, HCP continues processing the request. If the ETag matches any of the specified values, HCP returns a 304 (Not Modified) status code. To specify the values for this header, use this format: `"`*`value`*`"[, "`*`value`*`"]...` In this format, each value can be any string of one or more characters and must be enclosed in double quotation marks ("). Alternatively, you can specify a single asterisk (*) as the value for the If-None-Match header. No ETags match an asterisk in an If-None-Match header.	No
If-Unmodified-Since	Specifies a date and time, in Greenwich Mean Time (GMT), for comparison with the date and time the specified item was last modified. If the modification date and time is equal to or earlier than the specified date and time, HCP continues processing the request. If the modification date and time is later than the specified date and time, HCP returns a 412 (Precondition Failed) status code. For valid values for this header, see the description of the If-Modified-Since header above.	No
x-amz-date	See "Common request headers" on page 49.	x-amz-date or Date

Response headers

The table below describes the response headers returned in response to a successful request to check the existence an object, object version, or folder.

Response header	Description
Content-Length	Specifies the size, in bytes, of the object, object version, or folder.

(Continued)

Response header	Description
Content-Type	Specifies the Internet media type of the object or object version, or folder. This is one of, in order of precedence: • The Internet media type specified in the Content-Type header when the object was stored. • An Internet media type determined by HCP based on the object name. • **application/octet-stream** if HCP cannot determine the Internet media type. For a folder, the value of the Content-Type header is always **application/octet-stream**.
Date	See "Common response headers" on page 52.
ETag	Specifies the ETag for the specified object, object version, or folder. For a deleted version, this is the ETag of the version of the object that was deleted.
Last-Modified	Specifies the date and time at which the object, object version, or folder was last modified, in Greenwich Mean Time (GMT). For a deleted version, this is the date and time at which the deleted version was created. The date and time are expressed in this format: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* GMT For example: Mon, 18 Mar 2013 19:46:03 GMT Modifying an object means modifying its metadata. You cannot modify the content of an object.
Server	See "Common response headers" on page 52.
x-amz-delete-marker	For object versions only, indicates whether the object version is a deleted version. This header is returned only if versioning is enabled for the bucket and the object version is a deleted version. The value is always **true**.
x-amz-meta-	For objects and object versions only, specifies a custom metadata property/value pair. The response headers include one x-amz-meta- header for each applicable pair. For more information on this header, see "Retrieving custom metadata with HS3" on page 20.

(Continued)

Response header	Description
x-amz-missing-meta	For objects and object versions only, indicates that the .metapairs annotation for the object or object version doesn't contain valid XML or that the first line in the annotation doesn't begin with the **metapairs** element. This header is returned only when applicable. The value is always **1** (one).
x-amz-version-id	Specifies the version ID of the object version or folder. This header is returned only while versioning is enabled for the bucket.

HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to check the existence of an object, object version, or folder. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 53.

Code	Meaning	Description
200	OK	The specified object or object version exists.
304	Not Modified	One of: • The request included an If-None-Match header, and the ETag for the specified object or object version matched a value specified by the header. • The request included an If-Modified-Since header, and the specified object or object version was not modified after the date and time specified by the header.
403	Forbidden	One of: • The credentials provided with the request are invalid. • You do not have read permission for the specified bucket or object. • The HS3 API is currently disabled for the specified bucket.

(Continued)

Code	Meaning	Description
404	Not Found	One of: • The specified object, object version, or folder does not exist. • The specified object version is a deleted version. • The specified bucket does not exist.
412	Precondition Failed	One of: • The request included an If-Match header, and the ETag for the specified object or object version does not match any of the values specified by the header. • The request included an If-Unmodified-Since header, and the specified object or object version was modified after the date and time specified by the header.
500	Internal Server Error	An internal error occurred. Try the request again, gradually increasing the delay between each successive attempt. If this error persists, 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. If this error persists, contact your tenant administrator.

Examples

The following sections show sample **HEAD** requests for checking the existence of objects, object versions, and folders.

Example 1: Checking the existence of an object

Here’s a sample **HEAD** request that checks the existence of an object named sales_quotas_2013.pdf in the finance bucket. The request is being made while versioning is disabled for the bucket.

Request with s3curl command line

```
./s3curl.pl --id=lgreen --head -- -k
    "https://finance.europe.hcp.example.com/sales_quotas_2013.pdf"
```

Request headers

```
HEAD /sales_quotas_2013.pdf HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Thu, 04 Apr 2013 14:54:29 +0000
Authorization: AWS bGdyZWVu:LDa147ALd+O5Q02LjkGRrXwSGAc=
```

Response headers

```
HTTP/1.1 200 OK
Date: Thu, 04 Apr 2013 14:54:28 GMT
Server: HCP V6.0.1.117
ETag: "62e82b6d3ef16070a8d75ab55c42b80d"
Last-Modified: Mon, 03 Dec 2012 14:28:48 GMT
Content-Type: application/pdf
Content-Length: 23166
```

Example 2: Retrieving custom metadata for an object version

Here’s a sample **HEAD** request that checks the existence of a version of the object named AcctgBestPractices.doc, which has custom metadata. The request is being made while versioning is enabled for the bucket.

Request with s3curl command line

```
./s3curl.pl --id=lgreen --head -- -k
    "https://finance.europe.hcp.example.com/AcctgBestPractices.doc
    ?versionId=87288758401473"
```

Request headers

```
HEAD /AcctgBestPractices.doc?versionId=87288815588289 HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Thu, 04 Apr 2013 15:31:19 +0000
Authorization: AWS bGdyZWVu:EUa2evHNUtlw8Xu/1iFdGgWSDc8=
```

Response headers

```
HTTP/1.1 200 OK
Date: Thu, 04 Apr 2013 15:31:19 GMT
Server: HCP V6.0.1.117
ETag: "26aa5129552e57fc64e10aa5b3911ee2"
x-amz-version-id: 87288758401473
Last-Modified: Wed, 19 May 2012 14:56:05 GMT
x-amz-meta-author: P.D. Grey,Morgan White,Paris Black
Content-Type: application/msword
Content-Length: 3206178
```

Example 3: Checking whether an object has been modified

Here's a sample **HEAD** request that checks whether the object named AcctgBestPractices.doc has been modified since December 20, 2012, at 19:42:16 GMT. The request is being made while versioning is enabled for the bucket.

Request with s3curl command line

```
./s3curl.pl --id=lgreen --head -- -k
    "https://finance.europe.hcp.example.com/AcctgBestPractices.doc"
    -H "If-Modified-Since:Thu, 20 Dec 2012 19:42:16 +0000"
```

Request headers

```
HEAD /AcctgBestPractices.doc HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Thu, 04 Apr 2013 15:44:36 +0000
Authorization: AWS bGdyZWVu:Yn7gcia/yqzGRKRC04HaOuD3aH0=
If-Modified-Since: Thu, 20 Dec 2012 19:42:16 +0000
```

Response headers

```
HTTP/1.1 200 OK
Date: Thu, 04 Apr 2013 15:31:19 GMT
Server: HCP V6.0.1.117
ETag: "764f38262c6e581f678e1ac9b0211ae8"
x-amz-version-id: 87288815588289
Last-Modified: Thu, 20 Dec 2012 19:42:16 GMT
x-amz-meta-author: Morgan White
Content-Type: application/msword
Content-Length: 3552369
```

Example 4: Checking the existence of a folder

Here’s a sample **HEAD** request that checks the existence of a folder named r&d/budget_proposals in the finance bucket. The request is being made while versioning is enabled for the bucket.

Request with s3curl command line

```
./s3curl.pl --id=lgreen --head -- -k
    "https://finance.europe.hcp.example.com/r&d/budget_proposals/"
```

Request headers

```
HEAD /r&d/budget_proposals/ HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Tue, 09 Apr 2013 23:12:14 +0000
Authorization: AWS bGdyZWVu:4u9bhK0DQF2FLZtT0VVwMKdvm+I=
```

Response headers

```
HTTP/1.1 200 OK
Date: Tue, 09 Apr 2013 23:12:13 GMT
Server: HCP V6.0.1.117
ETag: "d41d8cd98f00b204e9800998ecf8427e"
x-amz-version-id: 87395113368065
Last-Modified: Tue, 04 Dec 2012 14:40:26 GMT
Content-Type: application/octet-stream
Content-Length: 0
```

Adding an ACL to an object

You use the HTTP **PUT** method with the **acl** query parameter to add an ACL to an existing object. Adding an ACL to an object replaces any existing ACL in its entirety. You cannot modify an existing ACL in place.

To add an ACL to an object, you need write ACL permission for the bucket containing the object or for the object itself.

You can add an ACL only to the current version of an object. However, the ACL you add applies to all versions of the object.

To add an ACL to an object, you can use either request headers or an ACL request body. You cannot use ACL headers and an ACL request body in the same request.

With ACL headers, you can specify either a canned ACL or individual x-amz-grant- headers. You cannot specify both a canned ACL and an x-amz-grant- header in the same request.

You can use an ACL request body to change the owner of an object. You cannot use ACL headers to do this. To change the owner of an object, you need both write ACL permission for the bucket or object and change owner permission for the bucket.

If you try to add an ACL that specifies a user account that does not exist, HCP returns a 400 (Bad Request) status code and does not add the ACL to the object.

For an introduction to ACLs and information on how to specify them, see "Access control lists" on page 22.

Request line

Depending on whether the bucket name is included in the hostname in the HS3 request, the request line for a request to add an ACL to an object has either of these formats:

- With the bucket name included in the hostname:

  PUT */object-name***?acl HTTP/1.1**

- With the bucket name following the hostname:

  PUT */bucket-name/object-name***?acl HTTP/1.1**

The ACL query parameter is not case sensitive.

Request headers

The table below describes the headers you can use in a request to add an ACL to an object.

Request header	Description	Required
Authorization	See "AWS authentication" on page 42.	Yes
Content-Length	Specifies the size, in bytes, of the ACL request body. This header is required when you're using an ACL request body to add an ACL to an object. If you're using ACL headers to add the ACL, the Content-Length header is invalid.	No

(Continued)

Request header	Description	Required
Content-Type	Specifies the Internet media type of the request body. This header is valid only when the ACL is specified in the request body. The only valid value is **application/xml.**	No
Date	See "Common request headers" on page 49.	Date or x-amz-date
Host	See "Common request headers" on page 49.	Yes
x-amz-acl	Adds a canned ACL to the object. This header is required for adding a canned ACL to an object. If you're using an ACL request body or individual x-amz-grant- headers to add the ACL, the x-amz-acl header is invalid. For valid values for this header, see "Canned ACLs" on page 25.	No
x-amz-date	See "Common request headers" on page 49.	x-amz-date or Date
x-amz-grant-full-control	Grants full control over the object to one or more specified grantees. If you're using an ACL request body or a canned ACL to add an ACL to an object, the x-amz-grant-full-control header is invalid. For valid values for this and the following x-amz-grant-headers, see "Using individual grant headers" on page 26.	No
x-amz-grant-read	Grants the browse and read data access permissions for the object to one or more specified grantees. If you're using an ACL request body or a canned ACL to add an ACL to an object, the x-amz-grant-read header is invalid.	No
x-amz-grant-read-acp	Grants the read ACL data access permission for the object to one or more specified grantees. If you're using an ACL request body or a canned ACL to add an ACL to an object, the x-amz-grant-read-acp header is invalid.	No
x-amz-grant-write	Grants the write and delete data access permissions for the object to one or more specified grantees. If you're using an ACL request body or a canned ACL to add an ACL to an object, the x-amz-grant-write header is invalid.	No

(Continued)

Request header	Description	Required
x-amz-grant-write-acp	Grants the write ACL data access permission for the object to one or more specified grantees. If you're using an ACL request body or a canned ACL to add an ACL to an object, the x-amz-grant-write-acp header is invalid.	No

Response headers

The table below describes the response headers returned in response to a successful request to add an ACL to an object.

Response header	Description
Content-Length	Specifies the size, in bytes, of the response body. In response to a successful request to add an ACL to an object, the value of this header is always **0** (zero).
Date	See "Common response headers" on page 52.
Server	See "Common response headers" on page 52.

HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to add an ACL to an object. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 53.

Code	Meaning	Description
200	OK	HCP successfully added the ACL to the object.

(Continued)

Code	Meaning	Description
400	Bad Request	One of: • You are trying to add an ACL that contains more than one hundred permission grants. • A specified grantee does not exist. • The specified owner does not exist. • Two grants of the same permission specify the same grantee. • The x-amz-acl header specifies an invalid value. • An x-amz-grant- header specifies an invalid identifier type. • The XML in the ACL request body is malformed or contains an invalid value.
403	Forbidden	One of: • The credentials provided with the request are invalid. • You do not have permission to add an ACL to the object. • The specified bucket does not currently support the requested operation. • The HS3 API is currently disabled for the specified bucket.
404	Not Found	One of: • The specified object does not exist. • The specified bucket does not exist.
500	Internal Server Error	An internal error occurred. Try the request again, gradually increasing the delay between each successive attempt. If this error persists, contact your tenant administrator.

(Continued)

Code	Meaning	Description
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. If this error persists, contact your tenant administrator.

Examples

The following sections show sample **PUT** requests for adding ACLs to objects.

Example 1: Adding an ACL to an object by using a canned ACL

Here's a sample **PUT** request that adds a canned ACL to the object named AcctgBestPractices.doc. The ACL grants read permission to all users.

Request with s3curl command line

```
./s3curl.pl --id=lgreen --put ~ -- -k
    "https:// finance.europe.hcp.example.com/AcctgBestPractices.doc?acl"
    -H "x-amz-acl:public-read"
```

Request headers

```
PUT /AcctgBestPractices.doc?acl HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Fri, 05 Apr 2013 12:20:37 +0000
Authorization: AWS bGdyZWVu:C8KpGuK62B7j2US0kN0Bl0Wx48k=
x-amz-acl: public-read
```

Response headers

```
HTTP/1.1 200 OK
Date: Fri, 05 Apr 2013 12:20:37 GMT
Server: HCP V6.0.1.117
Content-Length: 0
```

Example 2: Adding an ACL to an object by using an ACL request body

Here’s a sample **PUT** request that uses an ACL request body to add an ACL to the object named mktg/budget_proposals/BudgProp-2013. The ACL grants read access to all authenticated users and full control user pblack. It makes no change to the object owner. The ACL request body is in a file named acl-3.xml.

Request body

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <AccessControlList>
        <Grant>
            <Grantee xsi:type="Group"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <URI>http://acs.amazonaws.com/groups/global/
                AuthenticatedUsers</URI>
            </Grantee>
            <Permission>READ</Permission>
        </Grant>
        <Grant>
            <Grantee xsi:type="CanonicalUser"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <ID>b9d39144-a081-4761-b0e8-b8fb51e10192</ID>
                <DisplayName>pblack</DisplayName>
            </Grantee><Permission>FULL_CONTROL</Permission>
        </Grant>
    </AccessControlList>
</AccessControlPolicy>
```

Request with s3curl command line

```
./s3curl.pl  --id=lgreen  --put=acl-3.xml  --  -k
    "https://finance.europe.hcp.example.com/mktg/budget_proposals/
    BudgProp-2013?acl"
```

Request headers

```
PUT /mktg/budget_proposals/BudgProp-2013?acl HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Fri, 05 Apr 2013 13:39:05 +0000
Authorization: AWS bGdyZWVu:mRqVstjwWxvfEFr8JajjrPcd8eY=
Content-Length: 632
```

Response headers

```
HTTP/1.1 200 OK
Date: Fri, 05 Apr 2013 13:39:04 GMT
Server: HCP V6.0.1.117
Content-Length: 0
```

Retrieving the ACL for an object

You use the HTTP **GET** method with the **acl** query parameter to retrieve the ACL for an object. To retrieve the ACL for an object, you need read ACL permission for the bucket containing the object or for the object itself.

The object ACL is returned in an XML response body. The format of the response body is the same as the format you use for the ACL request body when you add an ACL to a bucket. For the format of the ACL request body, see “Specifying an ACL in the request body” on page 27.

For an introduction to ACLs, see “Access control lists” on page 22.

Request line

Depending on whether the bucket name is included in the hostname in the HS3 request, the request line for a request to retrieve an object ACL has either of these formats:

- With the bucket name included in the hostname:

  GET */object-name***?acl HTTP/1.1**

- With the bucket name following the hostname:

  GET */bucket-name/object-name***?acl HTTP/1.1**

The ACL query parameter is not case sensitive.

Request headers

The table below describes the headers you can use in a request to retrieve an object ACL.

Request header	Description	Required
Authorization	See "AWS authentication" on page 42.	Yes
Date	See "Common request headers" on page 49.	Date or x-amz-date
Host	See "Common request headers" on page 49.	Yes
x-amz-date	See "Common request headers" on page 49.	x-amz-date or Date
x-hcp-pretty-print	Optionally, requests that the XML response body be formatted for readability. Valid values are: • **true** — Format the XML response body for readability. • **false** — Do not apply any special formatting to the XML response body. The default is **false**. The values **true** and **false** are not case sensitive.	No

Response headers

The table below describes the response headers returned in response to a successful request to retrieve an object ACL.

Response header	Description
Content-Type	Specifies the Internet media type of the response body. For a request to retrieve an object ACL, the value of this header is always **application/xml;charset=UTF-8**.
Date	See "Common response headers" on page 52.
Server	See "Common response headers" on page 52.
Transfer-Encoding	Indicates that HCP could not determine the size of the response body before formulating the response. For a request to retrieve an object ACL, the value of this header is always **chunked**.

HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to retrieve an object ACL. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 53.

Code	Meaning	Description
200	OK	HCP successfully retrieved the object ACL.
403	Forbidden	One of: • The credentials provided with the request are invalid. • You do not have permission to read the object ACL. • The specified bucket does not currently support the requested operation. • The HS3 API is currently disabled for the specified bucket.
404	Not Found	One of: • The specified object does not exist. • The specified bucket does not exist.
500	Internal Server Error	An internal error occurred. Try the request again, gradually increasing the delay between each successive attempt. If this error persists, 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. If this error persists, contact your tenant administrator.

Example: Retrieving an object ACL

Here's a sample **GET** request that retrieves the ACL for the object named mktg/budget_proposals/BudgProp-2013.

Request with s3curl command line

```
./s3curl.pl  --id=lgreen  --  -k
     "https://finance.europe.hcp.example.com/mktg/budget_proposals/
     BudgProp-2013?acl"  -H "x-hcp-pretty-print:true"
```

Request headers

```
GET /mktg/budget_proposals/BudgProp-2013?acl HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Fri, 05 Apr 2013 14:15:44 +0000
Authorization: AWS bGdyZWVu:3pB7HQ8LNIHaFzHToefVgDD9hVo=
x-hcp-pretty-print: true
```

Response headers

```
HTTP/1.1 200 OK
Date: Fri, 05 Apr 2013 14:15:43 GMT
Server: HCP V6.0.1.117
Content-Type: application/xml;charset=UTF-8
Transfer-Encoding: chunked
```

Response body

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
     <Owner>
          <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
          <DisplayName>lgreen</DisplayName>
     </Owner>
     <AccessControlList>
          <Grant>
               <Grantee xsi:type="CanonicalUser"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                    <ID>b9d39144-a081-4762-b0e8-b8fb51e10192</ID>
                    <DisplayName>lgreen</DisplayName>
               </Grantee>
               <Permission>FULL_CONTROL</Permission>
          </Grant>
          <Grant>
               <Grantee xsi:type="Group"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                    <URI>http://acs.amazonaws.com/groups/global/AuthenticatedUsers
                    </URI>
               </Grantee>
               <Permission>READ</Permission>
          </Grant>
          <Grant>
               <Grantee xsi:type="CanonicalUser"
```

```
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                        <ID>b9d39144-a081-4761-b0e8-b8fb51e10192</ID>
                        <DisplayName>pblack</DisplayName>
                    </Grantee>
                    <Permission>FULL_CONTROL</Permission>
                </Grant>
            </AccessControlList>
        </AccessControlPolicy>
```

Copying an object

You use the HTTP **PUT** method with the x-amz-copy-source header to copy an object from one location to another. The source and target locations can be two different buckets within the same tenant, or they can be the same bucket. The source object is the object you're copying. The target object is the object that results from the copy operation.

To copy an object, you need read permission for the bucket containing the source object or for the source object itself and write permission for the target bucket.

When copying an object, you can specify a name for the target object that's different from the name of the source object.

By default, a copy operation copies the current version of the source object specified in the request. However, while versioning is enabled for the source bucket, you can use the **versionId** query parameter with the source object specification to copy a specific version of the object. If the version identified by the **versionId** parameter does not exist or is a deleted version, HCP returns a 404 (Not Found) status code.

HCP does not copy version IDs with objects. The object created by a copy operation has its own version ID.

By default, HCP copies any custom metadata for the source object to the target object. However, in the copy request, you can specify replacement custom metadata to be used for the target object. To apply this custom metadata to the target object, you need to include the x-amz-metadata-directive header with a value of **REPLACE** in the copy request.

HCP does not copy ACLs with objects. However, in the copy request, you can specify an ACL for the target object. To do this, you need to use ACL headers. You cannot use an ACL request body when copying an object. For information on ACLs, see "Access control lists" on page 22.

If the ACL you specify in a request to copy an object is invalid, HCP returns a 400 (Bad Request) or 501 (Not Implemented) status code and does not copy the object.

If you're an authenticated user, when you copy an object, you become the owner of the target object. If you're accessing the bucket anonymously, the target object has no owner. For information on object ownership, see "Object owners" on page 22.

In response to a request to copy an object, HCP returns an XML response body containing the ETag and last modification date of the target object. For the format of this response body, see "Response body" on page 160.

Copying an object to itself

If the target object you specify in a copy request identifies an existing object and versioning is enabled, HCP creates a new version of the existing object.

If the target object you specify in a copy request identifies an existing object and versioning is disabled:

- If the value of the x-amz-metadata-directive header is **REPLACE**, HCP replaces all custom metadata for the existing object with the custom metadata specified in the copy request. If the request doesn't specify any custom metadata, HCP removes all custom metadata from the existing object.

  When you replace or remove the custom metadata for an object in this way, the version ID of the object does not change.

- If the value of the x-amz-metadata-directive header is **COPY** or if the request does not include the x-amz-metadata-directive header, HCP returns a 400 (Bad Request) status code and does not replace the custom metadata for the object.

In no case does HCP replace the content of an existing object.

Conditionally copying an object

You can use the x-amz-copy-source-if-match, x-amz-copy-source-if-none-match, x-amz-copy-source-if-modified-since, and x-amz-copy-source-if-unmodified-since request headers to make copy requests conditional:

- The x-amz-copy-source-if-match and x-amz-copy-source-if-none-match headers compare the ETag of the source object or object version to one or more values that you specify. Typically, each value is the ETag for an object or object version of interest.

- The x-amz-copy-source-if-modified-since and x-amz-copy-source-if-unmodified-since headers compare the date and time the source object or object version was last modified to a date and time that you specify.

If the source object or object version:

- Meets all the conditions specified by the conditional headers included in the request, HCP performs the copy operation
- Does not meet all the conditions specified by the conditional headers included in the request, HCP returns a 412 (Precondition Failed) status code

If a request includes multiple different conditional headers, HCP processes any x-amz-copy-source-if-match and x-amz-copy-source-if-none-match headers before any x-amz-copy-source-if-modified-since or x-amz-copy-source-if-unmodified-since headers. If a request includes more than one of any given header, HCP processes only the first one of those headers and ignores the others.

Request line

Depending on whether the bucket name is included in the hostname in the HS3 request, the request line for a request to copy an object has either of these formats:

- With the bucket name included in the hostname:

  PUT */target-object-name* **HTTP/1.1**

- With the bucket name following the hostname:

  PUT */target-bucket-name/target-object-name* **HTTP/1.1**

Request headers

The table below describes the headers you can use in a request to copy an object.

Request header	Description	Required
Authorization	See "AWS authentication" on page 42.	Yes
Date	See "Common request headers" on page 49.	Date or x-amz-date
Host	See "Common request headers" on page 49.	Yes

(Continued)

Request header	Description	Required
x-amz-acl	Adds a canned ACL to the target object. This header is required for adding a canned ACL to an object. If you're using individual x-amz-grant- headers to add the ACL, the x-amz-acl header is invalid. For valid values for this header, see "Canned ACLs" on page 25.	No
x-amz-copy-source	Specifies the source bucket and object or object version, in this format: `/`*`bucket-name`*`/`*`source-object-name`* **`[?`***`versionId=source-object-version-id`***`]`** The initial forward slash (/) is required. Valid values for *`source-object-version-id`* are the IDs of versions of the source object specified in the request. The versionId query parameter is not case sensitive. If you include the **versionId** query parameter in the x-amz-copy-source header with an invalid value while versioning is enabled, HCP returns a 404 (Not Found) status code and does not copy the object. If you include the **versionId** query parameter in the x-amz-copy-source header while versioning is disabled, the parameter is ignored, and the current version of the specified object is used as the source for the copy operation.	Yes
x-amz-copy-source-if-match	Specifies one or more values for comparison with the ETag of the specified source object or object version. If the ETag matches one of the specified values, HCP continues processing the request. If the ETag doesn't match any of the specified values, HCP returns a 412 (Precondition Failed) status code. To specify the values for this header, use this format: **`"`***`value`***`"`**`[`**`, "`***`value`***`"`**`]...` In this format, each value can be any string of one or more characters and must be enclosed in double quotation marks ("). Alternatively, you can specify a single asterisk (*) as the value for the x-amz-copy-source-if-match header. All ETags match an asterisk in an x-amz-copy-source-if-match header.	No

(Continued)

Request header	Description	Required
x-amz-copy-source-if-modified-since	Specifies a date and time, in Greenwich Mean Time (GMT), for comparison with the date and time the specified source object or object version was last modified. If the modification date and time is later than the specified date and time, HCP continues processing the request. If the modification date and time is equal to or earlier than the specified date and time, HCP returns a 412 (Precondition Failed) status code. To specify the date and time for this header, use one of these formats: • *DDD*, *dd* *MMM* **yyyy** *HH*:*mm*:*ss* **+0000|GMT** For example: Thu, 07 Mar 2013 14:27:05 +0000 • *DDDD*, *dd*-*MMM*-*yy* *HH*:*mm*:*ss* **+0000|GMT** For example: Thursday, 07-Mar-13 14:27:05 +0000 • *DDD MMM d HH*:*mm*:*ss yyyy* For example: Thu Mar 7 14:27:05 2013 In these formats: • *DDD* is the three-letter abbreviation for the day of the week, with an uppercase first letter (for example, Mon). • *DDDD* is the day of the week fully spelled out, with an uppercase first letter (for example, Monday). • *d* is the one- or two-digit day of the month, as applicable. • *dd* is the two-digit day of the month. • *MMM* is the three-letter abbreviation for the month, with an uppercase first letter (for example, Feb). • *yy* is the last two digits of the year. HCP assumes that the intended year is within 80 years before or 20 years after the current year. • *yyyy* is the four-digit year. • *HH* is the hour on a 24-hour clock. • *mm* is the number of minutes into the hour. • *ss* is the number of seconds into the minute. If the value specified by the x-amz-copy-source-if-modified-since header doesn't conform to one of the formats, shown above, the header is ignored.	No

(Continued)

Request header	Description	Required
x-amz-copy-source-if-none-match	Specifies one or more values for comparison with the ETag of the specified source object or object version. If the ETag doesn't match any of the specified values, HCP continues processing the request. If the ETag matches any of the specified values, HCP returns a 412 (Precondition Failed) status code. To specify the values for this header, use this format: `"`*`value`*`"[, "`*`value`*`"]...` In this format, each value can be any string of one or more characters and must be enclosed in double quotation marks ("). Alternatively, you can specify a single asterisk (*) as the value for the x-amz-copy-source-if-match header. No ETags match an asterisk in an x-amz-copy-source-if-match header.	No
x-amz-copy-source-if-unmodified-since	Specifies a date and time, in Greenwich Mean Time (GMT), for comparison with the date and time the specified source object or object version was last modified. If the modification date and time is equal to or earlier than the specified date and time, HCP continues processing the request. If the modification date and time is later than the specified date and time, HCP returns a 412 (Precondition Failed) status code. For valid values for this header, see the description of the x-amz-copy-source-if-modified-since header above.	No
x-amz-date	See "Common request headers" on page 49.	x-amz-date or Date
x-amz-grant-full-control	Grants full control over the target object to one or more specified grantees. If you're using a canned ACL to add an ACL to an object, the x-amz-grant-full-control header is invalid. For valid values for this and the following x-amz-grant- headers, see "Using individual grant headers" on page 26.	No
x-amz-grant-read	Grants the browse and read data access permissions for the target object to one or more specified grantees. If you're using a canned ACL to add an ACL to an object, the x-amz-grant-read header is invalid.	No

(Continued)

Request header	Description	Required
x-amz-grant-read-acp	Grants the read ACL data access permission for the target object to one or more specified grantees. If you're using a canned ACL to add an ACL to an object, the x-amz-grant-read-acp header is invalid.	No
x-amz-grant-write	Grants the write and delete data access permissions for the target object to one or more specified grantees. If you're using a canned ACL to add an ACL to an object, the x-amz-grant-write header is invalid.	No
x-amz-grant-write-acp	Grants the write ACL data access permission for the target object to one or more specified grantees. If you're using a canned ACL to add an ACL to an object, the x-amz-grant-write-acp header is invalid.	No
x-amz-meta-	Adds custom metadata to the target object. For information on using this header, see "Storing custom metadata with HS3" on page 18.	No
x-amz-metadata-directive	Tells HCP whether to use the custom metadata, if any, for the source object or object version as the custom metadata for the target object or to use the custom metadata, if any, specified in the copy request. Valid values are: • **COPY** — Use the custom metadata for the source object or object version. • **REPLACE** — Use the custom metadata specified in the copy request. These values are case sensitive. The default is **COPY**.	No
x-amz-server-side-encryption	Requests that the response headers include x-amz-server-side-encryption, which indicates whether objects stored in HCP are encrypted. Valid values consist of the name of an encryption algorithm concatenated with an encryption key length (for example, **AES256**).	No

(Continued)

Request header	Description	Required
x-hcp-pretty-print	Optionally, requests that the XML response body be formatted for readability. Valid values are: • **true** — Format the XML response body for readability. • **false** — Do not apply any special formatting to the XML response body. The default is **false**. The values **true** and **false** are not case sensitive.	No

Response headers

The table below describes the response headers returned in response to a successful request to copy an object.

Response header	Description
Content-Type	Specifies the Internet media type of the response body. For a request to copy an object, the value of this header is always **application/xml;charset=UTF-8**.
Date	See "Common response headers" on page 52.
ETag	Specifies the ETag for the target object.
Server	See "Common response headers" on page 52.
Transfer-Encoding	Indicates that HCP could not determine the size of the response body before formulating the response. For a request to copy an object, the value of this header is always **chunked**.
x-amz-copy-source-version-id	Specifies the version ID of the source object. This header is returned only while versioning is enabled on the source bucket.
x-amz-server-side-encryption	Specifies whether objects stored in HCP are encrypted. Possible values are: • If objects are encrypted, the name of the encryption algorithm concatenated with the encryption key length (for example, **AES256**) • If objects are not encrypted, **None** This header is returned only if the request headers include x-amz-server-side-encryption.

(Continued)

Response header	Description
x-amz-version-id	Specifies the version ID of the target object. This header is returned only while versioning is enabled on the target bucket.

Response body

HCP returns information about the target object that results from a copy request in an XML response body, in this format:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CopyObjectResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <LastModified>date-and-time-last-modified</LastModified>
    <ETag>"etag"</ETag>
</CopyObjectResult>
```

The table below describes the XML elements in the format of the response body returned in response to a request to copy an object. The elements are listed in alphabetical order.

Element	Description
CopyObjectResult	Root element.
ETag	Child of the **CopyObjectResult** element. The **ETag** element specifies the ETag for the target object.
LastModified	Child of the **CopyObjectResult** element. The LastModified element specifies the date and time at which the target object was last modified, in Greenwich Mean Time (GMT). The date and time are expressed in this format: *yyyy-MM-dd*T*HH*:*mm*:*ss*.*SSS*Z For example: 2013-03-18T19:46:03.856Z Modifying an object means modifying its metadata. You cannot modify the content of an object.

HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to copy an object. For more information on HTTP status codes and the error codes that can accompany them, see “Error codes” on page 53.

Code	Meaning	Description
200	OK	HCP successfully copied the object.
400	Bad Request	An ACL grant header specifies an invalid grantee.
403	Forbidden	One of: • The credentials provided with the request are invalid. • You do not have permission to store objects in the target bucket. • The HS3 API is currently disabled for the source or target bucket.
404	Not Found	One of: • The specified source object or object version does not exist, or the specified object version is a deleted version. • The specified source bucket does not exist.
409	Conflict	One of: • An object with the specified target object name already exists in the target bucket, versioning is disabled for that bucket, and the copy request does not include an x-amz-metadata-directive header with a value of **REPLACE**. • A folder with the specified target object name already exists in the target bucket.

(Continued)

Code	Meaning	Description
412	Precondition Failed	One of: • The request included an x-amz-copy-source-if-match header, and the ETag for the specified source object or object version does not match any of the values specified by the header. • The request included an x-amz-copy-source-if-none-match header, and the ETag for the specified source object or object version matched a value specified by the header. • The request included an x-amz-copy-source-if-modified-since header, and the specified source object or object version was not modified after the date and time specified by the header. • The request included an x-amz-copy-source-if-unmodified-since header, and the specified source object or object version was modified after the date and time specified by the header.
413	Request Entity Too Large	The source object you are trying to copy is too big for the amount of space left in the target bucket.
500	Internal Server Error	An internal error occurred. Try the request again, gradually increasing the delay between each successive attempt. If this error persists, contact your tenant administrator.
501	Not Implemented	The request includes the x-amz-acl header with an invalid value.
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. If this error persists, contact your tenant administrator.

Examples

The following sections show sample **PUT** requests for copying objects.

Example 1: Conditionally copying an object from one bucket to another

Here’s a sample **PUT** request that conditionally copies the current version of an object named campaigns/GoGetEm.xls from the sales-mktg bucket to the finance bucket, where the target object that results from the copy operation is named mktg/campaign_GoGetEm_expenses.xls. The request is being made while versioning is enabled for both the source and target buckets.

For the purpose of this example, assume that mktg/campaign_GoGetEm_expenses.xls already exists in the target bucket with a last-modified date and time of Tuesday, February 12, 2013, at 10:14:06 GMT. The request says to perform the copy operation only if the last-modified date and time of the source object is later than the last-modified date and time of the target object.

Request with s3curl command line

```
./s3curl.pl  --id=lgreen  --copysrc=/sales-mktg/campaigns/GoGetEm.xls -- -k
    "https://finance.europe.hcp.example/mktg/campaign_GoGetEm_expenses.xls"
    -H  "x-amz-copy-source-if-modified-since:Tue,  12  Feb  2013  10:14:06  +0000"
    -H  "x-hcp-pretty-print:true"
```

Request headers

```
PUT /mktg/campaign_GoGetEm_expenses.xls HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Wed, 13 Feb 2013 17:44:53 +0000
Authorization: AWS bGdyZWVu:oBVRqkcjktavqo6z1m+chHhRmmI=
x-amz-copy-source: /sales-mktg/campaigns/GoGetEm.xls
x-amz-copy-source-if-modified-since: Tue, 12 Feb 2013 10:14:06 +0000
x-hcp-pretty-print: true
```

Response headers

```
HTTP/1.1 200 OK
Date: Wed, 13 Feb 2013 17:44:53 GMT
Server: HCP V6.0.1.117
x-amz-version-id: 87288825190337
ETag: "6ed7faad1e0661c03ad65a4317d4a94c"
x-amz-copy-source-version-id: 87388217426433
Content-Type: application/xml;charset=UTF-8
Transfer-Encoding: chunked
```

Response body

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CopyObjectResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <LastModified>2013-02-13T17:44:53.000Z</LastModified>
    <ETag>"6ed7faad1e0661c03ad65a4317d4a94c"</ETag>
</CopyObjectResult>
```

Example 2: Recovering an old version of an object

Here's a sample **PUT** request that copies an old version of an object to the same object, thereby creating a new current version from the old version. The object in question is named AcctgBestPractices.doc and is in the finance bucket. The version ID of the version being copied is 87288808614529. The request is being made while versioning is enabled for the bucket.

Request with s3curl command line

```
./s3curl.pl  --id=lgreen
    --copysrc=/finance/ AcctgBestPractices.doc?versionId=87288808614529  --  -k
    "https://finance.europe.hcp.example.com/AcctgBestPractices.doc"
    -H  "x-hcp-pretty-print:true"
```

Request headers

```
PUT /AcctgBestPractices.doc HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Thu, 20 Dec 2012 19:42:16 +0000
Authorization: AWS bGdyZWVu:AZ/GOgJJXFh7K1pr59bIlwRUrc0=
x-amz-copy-source: /finance/AcctgBestPractices.doc?versionId=87288808614529
x-hcp-pretty-print: true
```

Response headers

```
HTTP/1.1 200 OK
Date: Thu, 20 Dec 2012 19:42:16 GMT
Server: HCP V6.0.1.117
x-amz-version-id: 87288815588289
ETag: "764f38262c6e581f678e1ac9b0211ae8"
x-amz-copy-source-version-id: 87288808614529
Content-Type: application/xml;charset=UTF-8
Transfer-Encoding: chunked
```

Response body

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CopyObjectResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <LastModified>2012-02-20T19:42:16.000Z</LastModified>
    <ETag>"764f38262c6e581f678e1ac9b0211ae8"</ETag>
</CopyObjectResult>
```

Example 3: Replacing custom metadata for an existing object

Here's a sample **PUT** request that replaces the existing custom metadata for the object named hum_res/budget_proposals/BudgProp-2013 in the finance bucket with new custom metadata. The request is being made while versioning is disabled for the bucket, so the custom metadata is replaced on the current version of the object. No new version is created.

Request with s3curl command line

```
./s3curl.pl  --id=lgreen
    --copysrc=/finance/hum_res/budget_proposals/BudgProp-2013  --  -k
    "https://finance.europe.hcp.example.com/hum_res/budget_proposals/
    BudgProp-2013"  -H  "x-amz-meta-author:Robin Silver"
    -H  "x-amz-meta-department:Human Resources"
    -H  "x-amz-metadata-directive:REPLACE"  -H  "x-hcp-pretty-print:true"
```

Request headers

```
PUT /hum_res/budget_proposals/BudgProp-2013 HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Wed, 10 Apr 2013 18:29:34 +0000
Authorization: AWS bGdyZWVu:WAamEr9PkL76M/kWkFu5K2rY9Bs=
x-amz-copy-source: /finance/hum_res/budget_proposals/BudgProp-2013
x-amz-meta-author: Robin Silver
x-amz-meta-department: Human Resources
x-amz-metadata-directive: REPLACE
x-hcp-pretty-print: true
```

Response headers

```
HTTP/1.1 200 OK
Date: Wed, 10 Apr 2013 18:29:33 GMT
Server: HCP V6.0.1.117
Content-Type: application/xml;charset=UTF-8
Transfer-Encoding: chunked
```

Response body

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CopyObjectResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <LastModified>2012-11-16T23:29:17.062Z</LastModified>
    <ETag>"76216527ff2f6219f7c29251a619c8db"</ETag>
</CopyObjectResult>
```

Retrieving an object

You use the HTTP **GET** method to retrieve an object from a bucket. Retrieving an object means retrieving the object data. To retrieve an object, you need read permission for the bucket containing the object or for the object itself.

By default, a request to retrieve an object retrieves the current version of the object. However, while versioning is enabled for the target bucket, you can use the **versionId** query parameter to retrieve a specific version of an object.

If a retrieved object or object version has custom metadata, the headers returned in response to the request include the applicable x-amz-meta- headers. For information on x-amz-meta- headers, see "Custom metadata" on page 18.

Retrieving part of an object

You can use the Range header in a **GET** request to retrieve only part of an object or object version. By using the Range header, you can limit the amount of data returned, even when you don't know the size of the object.

The value of the Range header is the range of bytes you want to retrieve. The first byte of the data for an object is in position 0 (zero), so a range of 1-5 specifies the second through sixth bytes, not the first through fifth.

To specify a byte range in a range header, you use this format:

Range: bytes=*byte-range*

The table below shows the valid values for *byte-range*.

Value	Description	Example
start-position–end-position	Bytes in *start-position* through *end-position*, inclusive. If *end-position* is greater than the size of the object data, HCP returns the bytes in *start-position* through the end of the data. Valid values for *start-position* and *end-position* are integers greater than or equal to zero. For the specified range to be valid, *end-position* must be greater than or equal to *start-position*.	Five hundred bytes beginning with the two-hundred-first: bytes=200-699
start-position–	Bytes in *start-position* through the end of the object data. Valid values for *start-position* are integers greater than or equal to zero.	All the bytes beginning with the seventy-sixth and continuing through the end of the object data: bytes=75-
–offset-from-end	Bytes in the *offset-from-end* position, counted back from the last position in the object data, through the end of the object data. If offset-from-end is greater than the size of the object data, HCP returns the complete object data. Valid values for *offset-from-end* are integers greater than or equal to zero.	The last 25 bytes of the object data: bytes=-25

These considerations apply to Range header values:

- If you specify a valid range in which the start position is less than the size of the object data, HCP returns the requested range of data with a 206 (Partial Content) status code.

- If you specify a valid range in which the start position is greater than or equal to the size of the object data, HCP returns a 416 (Requested Range Not Satisfiable) status code and does not return any data.

- If you specify an offset of zero, HCP returns a 416 (Requested Range Not Satisfiable) status code and does not return any data.

- If you specify an invalid value (for example, a value in which the start position is greater than the end position, HCP ignores the Range header and returns the complete object data with a status code of 200 (OK).

Conditionally retrieving an object

You can choose to retrieve an object or object version only if its ETag and/or last modification date and time meet certain criteria. You might do this, for example, in an application that maintains a local cache of frequently used objects. With such an application, you can reduce the load on HCP and the network by retrieving objects only if they have changed in some way since they were cached.

You use the If-Match, If-None-Match, If-Modified-Since, and If-Unmodified-Since request headers to make **GET** requests conditional:

- The If-Match and If-None-Match headers compare the ETag for the requested object or object version to one or more values that you specify. Typically, each value is the ETag for an object or object version of interest.

- The If-Modified-Since and If-Unmodified-Since headers compare the date and time the requested object or object version was last modified to a date and time that you specify.

If the requested object or object version meets all the conditions specified by the conditional headers included in the request, HCP returns the object data. If the specified item does not meet the condition specified by:

- An If-Match or If-Unmodified-Since header, HCP returns a 412 (Precondition Failed) status code and does not return the object data

- An If-None-Match or If-Modified-Since header, HCP returns a 304 (Not Modified) status code and does not return the object data

If a request includes multiple different conditional headers, HCP processes any If-Match and If-None-Match headers before any If-Modified-Since or If-Unmodified-Since headers. If a request includes more than one of any given header, HCP processes only the first one of those headers and ignores the others.

Overriding response headers

In a request to retrieve an object or object version, you can specify values to be returned in certain response headers. The values you specify in the request override any values that might otherwise be returned for those headers. The headers you can override are returned only in response to a successful request.

To specify response header values, you can use the request headers listed in the table below. The valid values for each request header are the valid values for the corresponding response header.

Request header	Response header
response-cache-control	Cache-Control
response-content-disposition	Content-Disposition
response-content-encoding	Content-Encoding
response content-language	Content-Language
response-content-type	Content-Type
response-expires	Expires

This book does not describe response headers listed above, with the exception of Content-Type. For information on the other response headers, see http://www.w3.org/Protocols/rfc2616/rfc2616.html.

Request line

Depending on whether the bucket name is included in the hostname in the HS3 request, the request line for a request to retrieve an object has either of these formats:

- With the bucket name included in the hostname:

  HEAD */object-name*[**?versionId=***version-id*] **HTTP/1.1**

- With the bucket name following the hostname:

  HEAD */bucket-name/object-name*[**?versionId=***version-id*] **HTTP/1.1**

Valid values for *version-id* are integers greater than or equal to zero. Normally, the value of *version-id* is the version ID of a version of the object specified in the request.

The versionId query parameter is not case sensitive.

If you include the **versionId** query parameter in a **GET** request while versioning is disabled, the parameter is ignored.

Request headers

The table below describes the headers you can use in a request to retrieve an object.

Request header	Description	Required
Authorization	See "AWS authentication" on page 42.	Yes
Date	See "Common request headers" on page 49.	Date or x-amz-date
Host	See "Common request headers" on page 49.	Yes
If-Match	Specifies one or more values for comparison with the ETag of the specified object or object version. If the ETag matches one of the specified values, HCP continues processing the request. If the ETag doesn't match any of the specified values, HCP returns a 412 (Precondition Failed) status code. To specify the values for this header, use this format: `"`*`value`*`"[, "`*`value`*`"]...` In this format, each value can be any string of one or more characters and must be enclosed in double quotation marks ("). Alternatively, you can specify a single asterisk (*) as the value for the If-Match header. All ETags match an asterisk in an If-Match header.	No

(Continued)

Request header	Description	Required
If-Modified-Since	Specifies a date and time, in Greenwich Mean Time (GMT), for comparison with the date and time the specified object or object version was last modified. If the modification date and time is later than the specified date and time, HCP continues processing the request. If the modification date and time is equal to or earlier than the specified date and time, HCP returns a 304 (Not Modified) status code. To specify the date and time for this header, use one of these formats: • *DDD*, *dd MMM* **yyyy** *HH*:*mm*:*ss* **+0000|GMT** For example: Thu, 07 Mar 2013 14:27:05 +0000 • *DDDD*, *dd*-*MMM*-*yy HH*:*mm*:*ss* **+0000|GMT** For example: Thursday, 07-Mar-13 14:27:05 +0000 • *DDD MMM d HH*:*mm*:*ss yyyy* For example: Thu Mar 7 14:27:05 2013 In these formats: • *DDD* is the three-letter abbreviation for the day of the week, with an uppercase first letter (for example, Mon). • *DDDD* is the day of the week fully spelled out, with an uppercase first letter (for example, Monday). • *d* is the one- or two-digit day of the month, as applicable. • *dd* is the two-digit day of the month. • *MMM* is the three-letter abbreviation for the month, with an uppercase first letter (for example, Feb). • *yy* is the last two digits of the year. HCP assumes that the intended year is within 80 years before or 20 years after the current year. • *yyyy* is the four-digit year. • *HH* is the hour on a 24-hour clock. • *mm* is the number of minutes into the hour. • *ss* is the number of seconds into the minute. If the value specified by the If-Modified-Since header doesn't conform to one of the formats, shown above, the header is ignored.	No

(Continued)

Request header	Description	Required
If-None-Match	Specifies one or more values for comparison with the ETag of the specified object or object version. If the ETag doesn't match any of the specified values, HCP continues processing the request. If the ETag matches any of the specified values, HCP returns a 304 (Not Modified) status code. To specify the values for this header, use this format: `"value"[, "value"]...` In this format, each value can be any string of one or more characters and must be enclosed in double quotation marks ("). Alternatively, you can specify a single asterisk (*) as the value for the If-None-Match header. No ETags match an asterisk in an If-None-Match header.	No
If-Unmodified-Since	Specifies a date and time, in Greenwich Mean Time (GMT), for comparison with the date and time the specified object or object version was last modified. If the modification date and time is equal to or earlier than the specified date and time, HCP continues processing the request. If the modification date and time is later than the specified date and time, HCP returns a 412 (Precondition Failed) status code. For valid values for this header, see the description of the If-Modified-Since header above.	No
Range	Retrieves part of an object. For valid values for this header, see "Retrieving part of an object" on page 166.	No
response-cache-control	Specifies a value to be returned in the Cache-Control response header.	No
response-content-disposition	Specifies a value to be returned in the Content-Disposition response header.	No
response-content-encoding	Specifies a value to be returned in the Content-Encoding response header.	No
response-content-language	Specifies a value to be returned in the Content-Language response header.	No
response-content-type	Specifies a value to be returned in the Content-Type response header.	No
response-expires	Specifies a value to be returned in the Expires response header.	No
x-amz-date	See "Common request headers" on page 49.	x-amz-date or Date

Response headers

The table below describes the response headers returned in response to a successful request to retrieve an object.

Response header	Description
Content-Length	Specifies the size, in bytes, of the returned data.
Content-Range	Specifies the range of bytes retrieved out of the total size of the object data. This header is returned only if the request headers include Range.
Content-Type	Specifies the Internet media type of the object or object version. This is one of, in order of precedence: • The Internet media type specified by the response-content-type request header. • The Internet media type specified in the Content-Type header when the object was stored. • An Internet media type determined by HCP based on the object name. • **application/octet-stream** if HCP cannot determine the Internet media type.
Date	See “Common response headers” on page 52.
ETag	Specifies the ETag for the object or object version.
Last-Modified	Specifies the date and time at which the object or object version was last modified, in Greenwich Mean Time (GMT). For a deleted version, this is the date and time at which the deleted version was created. The date and time are expressed in this format: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* GMT For example: Mon, 18 Mar 2013 19:46:03 GMT Modifying an object means modifying its metadata. You cannot modify the content of an object.
Server	See “Common response headers” on page 52.
x-amz-meta-	Specifies a custom metadata property/value pair. The response headers include one x-amz-meta- header for each applicable pair. For more information on this header, see “Retrieving custom metadata with HS3” on page 20.

(Continued)

Response header	Description
x-amz-missing-meta	Indicates that the .metapairs annotation for the object or object version doesn't contain valid XML or that the first line in the annotation doesn't begin with the **metapairs** element. This header is returned only when applicable. The value is always **1** (one).
x-amz-version-id	Specifies the version ID of the object version. This header is returned only while versioning is enabled for the bucket.

HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to retrieve an object. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 53.

Code	Meaning	Description
200	OK	HCP successfully retrieved the complete data for the specified object or object version.
206	Partial Content	HCP successfully retrieved the data in the byte range specified by the Range header.
304	Not Modified	One of: • The request included an If-None-Match header, and the ETag for the specified object or object version matched a value specified by the header. • The request included an If-Modified-Since header, and the specified object or object version was not modified after the date and time specified by the header.
403	Forbidden	One of: • The credentials provided with the request are invalid. • You do not have read permission for the specified bucket or object. • The HS3 API is currently disabled for the specified bucket.

(Continued)

Code	Meaning	Description
404	Not Found	One of: • The specified object or object version does not exist • The specified object version is a deleted version. • The specified item is a folder. • The specified bucket does not exist.
412	Precondition Failed	One of: • The request included an If-Match header, and the ETag for the specified object or object version does not match any of the values specified by the header. • The request included an If-Unmodified-Since header, and the specified object or object version was modified after the date and time specified by the header.
416	Requested Range Not Satisfiable	The request included a Range header that specified either: • A range in which the starting value is greater than or equal to the size of the object data • An offset of zero For information on the Range header, see "Retrieving part of an object" on page 166.
500	Internal Server Error	An internal error occurred. Try the request again, gradually increasing the delay between each successive attempt. If this error persists, 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. If this error persists, contact your tenant administrator.

Examples

The following sections show sample **PUT** requests for retrieving objects.

Example 1: Conditionally retrieving an object

Here's a sample **PUT** request that retrieves the current version of the object named mktg/campaign_GoGetEm_expenses.xls in the finance bucket only if the ETag for the object doesn't match a specified value. In this example, the specified value is the ETag of the first version of mktg/campaign_GoGetEm_expenses.xls. The request writes the object data to a file named mktg_GoGetEm.xls. The request is being made while versioning is enabled for the bucket.

Request with s3curl command line

```
./s3curl.pl --id=lgreen -- -k "https://finance.europe.hcp.example.com/mktg/
    campaign_GoGetEm_expenses.xls"
    -H 'If-None-Match:"74d824cd5076a1361da128ee18e5a42b"' >
    mktg_GoGetEm.xls
```

Request headers

```
GET /mktg/campaign_GoGetEm_expenses.xls HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Fri, 12 Apr 2013 13:35:31 +0000
Authorization: AWS bGdyZWVu:3ymfU6KeNWnFEvpphFxYvJ881Wg=
If-None-Match: "74d824cd5076a1361da128ee18e5a42b"
```

Response headers

```
HTTP/1.1 200 OK
Date: Fri, 12 Apr 2013 13:35:30 GMT
Server: HCP V6.0.1.117
ETag: "6ed7faad1e0661c03ad65a4317d4a94c"
x-amz-version-id: 87288825190337
Last-Modified: Wed, 13 Feb 2013 17:44:53 GMT
Cache-Control: no-cache,no-store
Pragma: no-cache
Content-Type: application/vnd.ms-excel
Content-Length: 94328
```

Example 2: Retrieving an old version of an object

Here's a sample **PUT** request that retrieves an old version of the object named AcctgBestPractices.doc in the finance bucket. The request writes the object data to a file named AcctgBestPractices.doc-10-08-12.doc. The request is being made while versioning is enabled for the bucket.

Request with s3curl command line

```
./s3curl.pl  --id=lgreen  --  -k  "https://finance.europe.hcp.example.com/
    AcctgBestPractices.doc?versionId=87288808614529"  >
    AcctgBestPractices.doc-10-08-12.doc
```

Request headers

```
GET /AcctgBestPractices.doc?versionId=87288808614529 HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Fri, 12 Apr 2013 13:49:37 +0000
Authorization: AWS bGdyZWVu:6Am3akZkcfc4fD3WXSBFr+dV7DE=
```

Response headers

```
HTTP/1.1 200 OK
Date: Fri, 12 Apr 2013 13:49:35 GMT
Server: HCP V6.0.1.117
ETag: "5ab7542f753b09fdb73141a66c134b9"
x-amz-version-id: 87288808614529
Last-Modified: Tue, 18 Dec 2012 21:06:52 GMT
Cache-Control: no-cache,no-store
Pragma: no-cache
Content-Type: application/msword
Content-Length: 3557448
```

Example 3: Retrieving part of an object

Here’s a sample **PUT** request that retrieves the first hundred thousand bytes of the object named quarterly_rpts/Q4_2012.ppt in the finance bucket. The request writes the object data to a file named Q4_2012_Rpt_Part-1. The request is being made while versioning is disabled for the bucket.

Request with s3curl command line

```
./s3curl.pl  --id=lgreen  --  -k
    "https://finance.europe.hcp.example.com/quarterly_rpts/Q4_2012.ppt"
    -H "Range:bytes=0-99999"  >  Q4_2012_Rpt_Part-1
```

Request headers

```
GET /quarterly_rpts/Q4_2012.ppt HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Fri, 12 Apr 2013 14:52:03 +0000
Authorization: AWS bGdyZWVu:f6RKgLahMlrfc7de89aJ0Xt8wKM=
Range: bytes=0-99999
```

Response headers

```
HTTP/1.1 206 Partial Content
Date: Fri, 12 Apr 2013 14:52:02 GMT
Server: HCP V6.0.1.117
ETag: "617e8ef649d40cda1f7f3ca81c97a06a"
Last-Modified: Mon, 22 Jan 2013 17:19:26 GMT
Cache-Control: no-cache,no-store
Pragma: no-cache
Content-Type: application/vnd.ms-powerpoint
Content-Range: bytes 0-99999/235813
Content-Length: 100000
```

Deleting an object or folder

You use the HTTP **DELETE** method to delete an object or folder in a bucket. To check the delete an object, you need delete permission for the bucket containing the object or for the object itself. To delete a folder, you need delete permission for the bucket.

You cannot delete an old version of an object. You also cannot delete a deleted version, even if it is the current version.

Request line

Depending on whether the bucket name is included in the hostname in the HS3 request, the request line for a request to delete an object or folder has either of these formats:

- With the bucket name included in the hostname:

  HEAD /(*object-name*)|(*folder-name*(**/**|**%2F**)) **HTTP/1.1**

- With the bucket name following the hostname:

  HEAD /*bucket-name***/**(*object-name*)| (*folder-name*(**/**|**%2F**)) **HTTP/1.1**

Request headers

The table below describes the headers you can use in a request to check the existence of an object or folder.

Request header	Description	Required
Authorization	See "AWS authentication" on page 42.	Yes
Date	See "Common request headers" on page 49.	Date or x-amz-date
Host	See "Common request headers" on page 49.	Yes
x-amz-date	See "Common request headers" on page 49.	x-amz-date or Date

Response headers

The table below describes the response headers returned in response to a successful request to delete an object or folder.

Response header	Description
Date	See "Common response headers" on page 52.
Server	See "Common response headers" on page 52.
x-amz-delete-marker	Indicates that the object version has been deleted. The value of this header is always **true**. This header is returned only while versioning is enabled.
x-amz-version-id	Specifies the version ID of the object or folder that was deleted. This header is returned only while versioning is enabled.

HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to delete an object or folder. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 53.

Code	Meaning	Description
204	No Content	One of: • HCP successfully deleted the object or folder. • The specified object or folder does not exist.

(Continued)

Code	Meaning	Description
403	Forbidden	One of: • The credentials provided with the request are invalid. • You do not have delete permission for the specified bucket or object. • The request includes a **versionId** query parameter that identifies the current version of the object, and the current version of the object is a deleted version. • The HS3 API is currently disabled for the specified bucket.
404	Not Found	The specified bucket does not exist.
409	Conflict	The specified object is under retention.
500	Internal Server Error	An internal error occurred. Try the request again, gradually increasing the delay between each successive attempt. If this error persists, contact your tenant administrator.
501	Not Implemented	The request includes a **versionId** query parameter that identifies an old version of the object or a deleted version that is not the current version.
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. If this error persists, contact your tenant administrator.

Examples

The following sections show sample **DELETE** requests for deleting objects and folders.

Example 1: Deleting an object

Here's a sample **DELETE** request that deletes an object named hum_res/budget_proposals/BudgProp-2013 from the finance bucket. The example shows the response headers HCP returns while versioning is enabled for the bucket and while versioning is disabled for the bucket.

Request with s3curl command line

```
./s3curl.pl --id=lgreen --delete -- -k "https://finance.europe.hcp.example.com/
    hum_res/budget_proposals/BudgProp-2013"
```

Request headers

```
DELETE /hum_res/budget_proposals/BudgProp-2013 HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Fri, 12 Apr 2013 18:20:55 +0000
Authorization: AWS bGdyZWVu:i9bRonH4gi1SrymsF0Fw84mWUeQ=
```

Response headers with versioning enabled

```
HTTP/1.1 204 No Content
Date: Fri, 12 Apr 2013 18:20:54 GMT
Server: HCP V6.0.1.117
x-amz-version-id: 87288781940929
x-amz-delete-marker: true
```

Response headers with versioning disabled

```
HTTP/1.1 204 No Content
Date: Fri, 12 Apr 2013 18:20:54 GMT
Server: HCP V6.0.1.117
```

Example 2: Deleting a folder

Here's a sample **DELETE** request that deletes a folder named hum_res/budget_proposals from the finance bucket. The request is being made while versioning is disabled for the bucket.

Request with s3curl command line

```
./s3curl.pl --id=lgreen --delete -- -k
    "https://finance.europe.hcp.example.com/hum_res/budget_proposals/"
```

Request headers

```
DELETE /hum_res/budget_proposals/ HTTP/1.1
Host: finance.europe.hcp.example.com
Date: Fri, 12 Apr 2013 18:28:52 +0000
Authorization: AWS bGdyZWVu:6iZRgTyBmsiLxX37WDtIS8X7vT8=
```

Response headers

```
HTTP/1.1 204 No Content
Date: Fri, 12 Apr 2013 18:28:51 GMT
Server: HCP V6.0.1.117
```

7

Usage considerations

This chapter contains considerations that apply to using the HS3 API.

Hostname and IP address considerations

In the URL you use to access HCP, you can specify either a hostname or an IP address. If the HCP system supports DNS and you specify a hostname, HCP selects the IP address for you from the currently available nodes. HCP uses a round-robin method to ensure that it doesn't always select the same address.

When you specify IP addresses, your application must take responsibility for balancing the load among nodes. Also, you risk trying to connect (or reconnect) to a node that is not available. However, in several cases using explicit IP addresses to connect to specific nodes can have advantages over using hostnames.

These considerations apply when deciding which technique to use:

- If your client uses a hosts file to map HCP hostnames to IP addresses, the client system has full responsibility for converting any hostnames to IP addresses. Therefore, HCP cannot spread the load or prevent attempts to connect to an unavailable node. For more information on using a hosts file, see "Using a hosts file" on page 40.

- If your client caches DNS information, connecting by hostname may result in the same node being used repeatedly.

- When you access the HCP system by hostname, HCP ensures that requests are distributed among nodes, but it does not ensure that the resulting loads on the nodes are evenly balanced.

- When multiple applications access the HCP system by hostname concurrently, HCP is less likely to spread the load evenly across the nodes than with a single application.

Tip: When using hostnames, you can ping the HCP system periodically to check whether you're getting connections to different nodes.

Folder structures

Because of the way HCP stores objects, the folders you create and the way you store objects in them can have an impact on performance. Here are some guidelines for creating effective folder structures:

- Plan the folder structure for a bucket before storing objects in the bucket. Make sure all users of the bucket are aware of these plans.

- Avoid structures that result in a single folder getting a large amount of traffic in a short time. For example, if you ingest objects rapidly, use a structure that does not store objects by date and time.

- If you do store objects by date and time, consider the number of objects ingested during a given period of time when planning the folder structure. For example, if you ingest several hundred objects per second, you might use a folder structure such as year/month/day/hour/minute/second. If you ingest only a few folders per second, a less fine-grained structure would be better.

- Follow these guidelines on the size of the folder structure:

  – Try to balance the width and depth of the folder structure.

  – Do not create folder structures that are more than 20 levels deep. Instead, create flatter folder structures.

  – Avoid placing a large number of objects (greater than 100,000) in a single folder. Instead, create multiple folders and evenly distribute the objects among them.

Concurrent writes of the same object

If two or more clients try to store an object with the same name at the same time, what happens depends on whether versioning is enabled for the target bucket:

- If versioning is enabled, HCP creates one version of the object for each **PUT** request. The versions are numbered in the order in which HCP received the requests, regardless of the order in which HCP finished processing the requests.

- If versioning is disabled and the bucket doesn't already contain an object with the specified name, HCP creates the object for the first **PUT** request. In response to each subsequent **PUT** request, HCP returns a 409 (Conflict) status code and does not create an object. This happens regardless of whether HCP has finished processing the first request.

Failed PUT requests to store objects

A **PUT** request to store an object fails if either of these happens:

- The target node fails while the object is open for write.

- The TCP connection breaks while the object is open for write (for example, due to a network failure or the abnormal termination of the client application).

Also, in some circumstances, a **PUT** request fails if HCP system hardware fails while HCP is processing the request.

When a **PUT** request fails, HCP does not create a new object or object version.

Tip: If a **PUT** request fails, retry the request.

Empty objects

When you use a **PUT** request to write a zero-sized file to HCP, the result is an empty object (that is, an object that has no data). Empty objects are WORM and are treated like any other object.

Deleting objects under repair

HCP regularly checks the health of the objects stored in the repository. If an object is found to be unhealthy, HCP tries to repair it.

If you try to delete an object while it is under repair, HCP returns a 409 (Conflict) status code and does not delete the object. In response to such an error, you should wait a few minutes and then try the request again.

Multithreading

HCP lets multiple threads access a bucket concurrently. Using multiple threads can enhance performance, especially when accessing many small objects across multiple folders.

Here are some guidelines for the effective use of multithreading:

- Concurrent threads, both reads and writes, should be directed against different folders. If that's not possible, multiple threads working against a single folder is still better than a single thread.

- To the extent possible, concurrent threads should work against different IP addresses. If that's not possible, multiple threads working against a single IP address is still better than a single thread.

- Only one client can write to a given object at one time. Similarly, a multithreaded client cannot have multiple threads writing to the same object at the same time. However, a multithreaded client can write to multiple objects at the same time.

- Multiple clients can read the same object concurrently. Similarly, a multithreaded client can use multiple threads to read a single object. However, because the reads can occur out of order, you generally get better performance by using one thread per object.

The HS3 API shares a connection pool with the HTTP and WebDAV protocols. HCP has a limit of 255 concurrent connections from this pool, with another 20 queued.

Tip: For better performance, consider limiting the number of concurrent read threads per node to 200 and concurrent write threads per node to 50 for small objects. For large objects, consider using fewer threads.

Persistent connections

HCP supports persistent connections. Following a request for an operation, HCP keeps the connection open for 60 seconds, so a subsequent request can use the same connection.

Persistent connections enhance performance because they avoid the overhead of opening and closing multiple connections. In conjunction with persistent connections, using multiple threads so that operations can run concurrently provides still better performance.

If the persistent connection timeout period is too short, tell your tenant administrator.

Note: With persistent connections, if a single IP address has more than 254 concurrent open connections, those above the first 254 may have to wait as long as ten minutes to be serviced. This includes connections where the request explicitly targeted the IP address, as well as connections where the HCP hostname resolved to the target IP address.

To avoid this issue, either don’t use persistent connections or ensure that no more than 254 threads are working against a single node at any time.

Connection failure handling

You should retry an HS3 request if either of these happens:

- The client cannot establish an HS3 connection to the HCP system.
- The connection breaks while HCP is processing a request. In this case, the most likely cause is that the node processing the request became unavailable.

When retrying the request:

- If the original request used the hostname of the HCP system in the URL, repeat the request in the same way.
- If the original request used an IP address, retry the request using either a different IP address or the hostname of the system.

If the connection breaks while HCP is processing a **GET** request, you may not know whether the returned data is all or only some of the object data. In this case, you can check the number of returned bytes against the content length returned in the HS3 Content-Length response header. If the numbers match, the returned data is complete.

A

Quick reference

This appendix contains a quick reference to the HS3 API methods. For each method, the appendix shows:

- The query parameters you can use in requests that use the method. Required parameters are listed first followed by optional parameters enclosed in square brackets ([]).
- The request headers for requests that use the method. Required headers are listed first followed by optional headers enclosed in square brackets.

The methods are grouped by the level of access (service (that is, tenant), bucket, or object). Within each grouping, the methods are presented in alphabetical order.

For general information on HS3 requests, see "Requests" on page 48.

Description and permission	Query parameters	Request headers
Service-level operation		
GET service		
Description: Returns a list of buckets owned by the requester **Permission:** None; requester must be an authenticated user **More information:** "Listing the buckets you own" on page 65	N/A	Authorization: AWS *access-key*:*signature* Date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT OR x-amz-date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT Host: *hostname*.*hcp-domain-name* [x-hcp-pretty-print: true|false]
Bucket-level operations		
DELETE bucket		
Description: Deletes a bucket **Permission:** None; requester must be the bucket owner **More information:** "Deleting a bucket" on page 115	N/A	Authorization: AWS *access-key*:*signature* Date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT OR x-amz-date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT Host: *hostname*.*hcp-domain-name*
GET bucket		
Description: Lists objects and folders in a bucket **Permission:** Browse **More information:** "Listing bucket contents" on page 91	[delimiter=*string*] [marker=*string*] [max-keys=*integer*] [prefix=*string*]	Authorization: AWS *access-key*:*signature* Date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT OR x-amz-date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT Host: *hostname*.*hcp-domain-name* [x-hcp-pretty-print: true|false]

(Continued)

Description and permission	Query parameters	Request headers
GET bucket acl		
Description: Retrieves the ACL for a bucket **Permission:** Read ACL **More information:** "Retrieving the ACL for a bucket" on page 79	acl	Authorization: AWS *access-key*:*signature* Date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT OR x-amz-date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT Host: *hostname*.*hcp-domain-name* [x-hcp-pretty-print: true|false]
GET bucket versioning		
Description: Checks the versioning status of a bucket **Permission:** None; requester must be the bucket owner **More information:** "Checking the versioning status of a bucket" on page 87	versioning	Authorization: AWS *access-key*:*signature* Date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT OR x-amz-date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT Host: *hostname*.*hcp-domain-name* [x-hcp-pretty-print: true|false]
GET bucket versions		
Description: Lists versions of objects in a bucket **Permission:** Browse **More information:** "Listing bucket contents" on page 91	versions [delimiter=*string*] [key-marker=*string*] [max-keys=*integer*] [prefix=*string*] [version-id-marker= *integer*]	Authorization: AWS *access-key*:*signature* Date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT OR x-amz-date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT Host: *hostname*.*hcp-domain-name* [x-hcp-pretty-print: true|false]

(Continued)

Description and permission	Query parameters	Request headers
HEAD bucket		
Description: Checks the existence of a bucket **Permission:** Read **More information:** "Checking the existence of a bucket" on page 70	N/A	Authorization: AWS *access-key*:*signature* Date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT OR x-amz-date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT Host: *hostname*.*hcp-domain-name*
PUT bucket		
Description: Creates a bucket and optionally adds an ACL to it **Permission:** None; requester must be an authenticated user **More information:** "Creating a bucket" on page 60	N/A	Authorization: AWS *access-key*:*signature* Content-Length: 0 Date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT OR x-amz-date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT Host: *hostname*.*hcp-domain-name* [x-amz-acl: *canned-acl-name*] [x-amz-grant-full-control: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-grant-read: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-grant-read-acp: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-grant-write: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-grant-write-acp: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...]

(Continued)

Description and permission	Query parameters	Request headers
PUT bucket acl		
Description: Adds an ACL to a bucket; removes an ACL from a bucket; changes the bucket owner **Permission:** To add or remove an ACL, write ACL; to change the bucket owner, write ACl and change owner **More information:** "Adding an ACL to a bucket" on page 73	acl	Authorization: AWS *access-key*:*signature* Date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT OR x-amz-date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT Host: *hostname*.*hcp-domain-name* [Content-Length: *acl-request-body-size-in-bytes*] *(Required with an ACL request body)* [Content-Type: application/xml] [x-amz-acl: *canned-acl-name*] [x-amz-grant-full-control: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-grant-read: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-grant-read-acp: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-grant-write: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-grant-write-acp: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...]

(Continued)

Description and permission	Query parameters	Request headers
PUT bucket versioning		
Description: Enables or disables versioning for a bucket **Permission:** None; requester must be the bucket owner **More information:** “Enabling or disabling versioning for a bucket” on page 83	versioning	Authorization: AWS *access-key*:*signature* Content-Length: *versioning-request-body-size-in-bytes* Date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT OR x-amz-date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT Host: *hostname*.*hcp-domain-name* [Content-Type: application/xml]
Object-level operations		
DELETE object		
Description: Deletes an object or folder **Permission:** Delete **More information:** “Deleting an object or folder” on page 178	N/A	Authorization: AWS *access-key*:*signature* Date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT OR x-amz-date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT Host: *hostname*.*hcp-domain-name*

(Continued)

Description and permission	Query parameters	Request headers
GET object		
Description: Retrieves an object or object version **Permission:** Read **More information:** "Retrieving an object" on page 166	[versionId= *version-id*]	Authorization: AWS *access-key*:*signature* Date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT OR x-amz-date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT Host: *hostname*.*hcp-domain-name* [If-Match: "*value*"[, "*value*"]...] [If-Modified-Since: *datetime-value*] *(For formats for datetime-value, see the description of If-Modified-Since on page 171)* [If-None-Match: "*value*"[, "*value*"]...] [If-Unmodified-Since: *datetime-value*] [response-cache-control: *cache-control-header-value*] [response-content-disposition: *content-disposition-header-value*] [response-content-encoding: *content-encoding-header-value*] [response-content-language: *content-language-header-value*] [response-content-type: *content-type-header-value*] [response-expires: *expires-header-value*]

(Continued)

Description and permission	Query parameters	Request headers
GET object acl		
Description: Retrieves the ACL for an object **Permission:** Read ACL **More information:** "Retrieving the ACL for an object" on page 148	acl	Authorization: AWS *access-key*:*signature* Date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT OR x-amz-date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT Host: *hostname*.*hcp-domain-name* [x-hcp-pretty-print: true|false]
HEAD object		
Description: Checks the existence of an object, object version, or folder **Permission:** Read **More information:** "Checking the existence of an object or folder" on page 131	[versionId= *version-id*]	Authorization: AWS *access-key*:*signature* Date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT OR x-amz-date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT Host: *hostname*.*hcp-domain-name* [If-Match: "*value*"[, "*value*"]...] [If-Modified-Since: *datetime-value*] *(For formats for datetime-value, see the description of If-Modified-Since on page 134)* [If-None-Match: "*value*"[, "*value*"]...] [If-Unmodified-Since: *datetime-value*]

(Continued)

Description and permission	Query parameters	Request headers
PUT object		
Description: Stores an object or new version of an object **Permission:** Write **More information:** "Storing an object" on page 120	N/A	Authorization: AWS *access-key*:*signature* Content-Length: *data-size-in-bytes* Date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT OR x-amz-date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT Host: *hostname*.*hcp-domain-name* [Content-MD5: *base64-encoded-md5-hash-of-data*] [Content-Type: *internet-media-type*] [Expect: 100-continue] [x-amz-acl: *canned-acl-name*] [x-amz-grant-full-control: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-grant-read: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-grant-read-acp: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-grant-write: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-grant-write-acp: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-meta-*property-name*: *value*] [x-amz-server-side-encryption: *encryption-algorithm-name-and-key-length*]

(Continued)

Description and permission	Query parameters	Request headers
PUT object acl		
Description: Adds an ACL to an object; removes an ACL from an object; changes the object owner **Permission:** To add or remove an ACL, write ACL; to change the object owner, write ACl and change owner **More information:** "Adding an ACL to an object" on page 141	acl	Authorization: AWS *access-key*:*signature* Date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT OR x-amz-date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT Host: *hostname*.*hcp-domain-name* [Content-Length: *acl-request-body-size-in-bytes*] *(Required with an ACL request body)* [Content-Type: application/xml] [x-amz-acl: *canned-acl-name*] [x-amz-grant-full-control: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-grant-read: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-grant-read-acp: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-grant-write: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-grant-write-acp: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...]

(Continued)

Description and permission	Query parameters	Request headers
PUT object copy		
Description: Copies an object; replaces custom metadata for an object **Permission:** Read for the source object; write for the target bucket **More information:** "Copying an object" on page 152	N/A	Authorization: AWS *access-key*:*signature* Date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT OR x-amz-date: *DDD*, *dd MMM yyyy HH*:*mm*:*ss* +0000|GMT Host: *hostname*.*hcp-domain-name* x-amz-copy-source: /*bucket-name*/*source-object-name* [?versionId=*source-object-version-id*] [x-amz-acl: *canned-acl-name*] [x-amz-copy-source-if-match: "*value*"[, "*value*"]...] [x-amz-copy-source-if-modified-since: *datetime-value*] *(For formats for datetime-value, see the description of x-amz-copy-source-if-modified-since on page 156)* [x-amz-copy-source-if-none-match: "*value*"[, "*value*"]...] [x-amz-copy-source-if-unmodified-since: *datetime-value*] [x-amz-server-side-encryption: *encryption-algorithm-name-and-key-length*] [x-amz-grant-full-control: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-grant-read: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-grant-read-acp: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...]

(Continued)

Description and permission	Query parameters	Request headers
PUT object copy (continued)		
		[x-amz-grant-write: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-grant-write-acp: *identifier-type*=*grantee-identifier* [, *identifier-type*=*grantee-identifier*]...] [x-amz-meta-*property-name*: *value*] [x-amz-metadata-directive: COPY|REPLACE]

B

Alternative authentication method

As an alternative to AWS method of authentication, you can use the HCP method. You might choose to do this, for example, if you're writing an application that uses both the HS3 API and the HTTP protocol (the HCP REST API). With the HTTP protocol, HCP supports only its own authentication method.

With HCP authentication, the format of the URLs you use in HS3 requests differs from the format used with AWS authentication.

This appendix describes the URLs you can use in HS3 requests when using HCP authentication. It also explains how to provide credentials or request anonymous access with that authentication method.

For information on AWS authentication and the URLs you can use with that authentication method, see Chapter 3, "Access and authentication," on page 37.

URLs with HCP authentication

The format you use for the URL in an HS3 request depends on whether the target of the request and the authentication method being used. In an HS3 request that uses HCP authentication:

- If the target of the request is a tenant, you use a URL in this format:

  http[**s**]**://***tenant-name***.***hcp-domain name***/hs3**

  Here's an example in which the tenant name is europe:

  https://europe.hcp.example.com/hs3

- If the target of the request is a bucket, you use a URL in either of these formats:

  http[**s**]**://***bucket-name***.***tenant-name***.***hcp-domain name***/hs3**

  http[**s**]**://***tenant-name***.***hcp-domain name***/hs3/***bucket-name*

  In the first format above, the bucket name is part of the hostname. In the second format, the bucket name follows the hostname and *hs3* interface identifier.

  Here's are examples in which the tenant name is europe and the bucket name is finance:

  https://finance.europe.hcp.example.com/hs3

  https://europe.hcp.example.com/hs3/finance

- If the target of the request is an object, you use a URL in either of these formats:

  http[**s**]**://***bucket-name***.***tenant-name***.***hcp-domain name***/hs3/***object-name*

  http[**s**]**://***tenant-name***.***hcp-domain name***/hs3/***bucket-name***/***object-name*

  In the first format above, the bucket name is part of the hostname. In the second format, the bucket name follows the hostname and *hs3* interface identifier.

Here's are examples in which the tenant name is europe, the bucket name is finance, and the object name is Q4_2012.ppt:

```
https://finance.europe.hcp.example.com/hs3/Q4_2012.ppt

https://europe.hcp.example.com/hs3/finance/Q4_2012.ppt
```

In these formats, the *hs3* interface identifier is case sensitive and must be all lowercase.

Note: Do not use the *hs3* interface identifier when using AWS authentication.

For additional information about URLs in HS3 requests, see "URLs for access to HCP" on page 38.

HCP authentication

In an HS3 request that uses HCP authentication, you use the HTTP Authorization header to provide credentials, in this format:

Authorization: HCP *access-key*:*secret-key*

In this format:

- *access-key* is the Base64-encoded username for your user account.
- *secret-key* is the MD5-hashed password for your user account.

Here's an example of an Authorization header for HCP authentication:

```
Authorization: AWS bGdyZWVu:35dc4c4aa08fe0deab7e292e00eb8e97
```

With HCP authentication, the Authorization header for requesting anonymous access looks like this:

```
Authorization: HCP all_users:
```

For more information on user authentication, see "AWS authentication" on page 42. For information on changing the password for your user account, see, "Changing your password" on page 45.

C

Using third-party tools with HS3

HS3 is also compatible with many third-party tools that support Amazon S3. These tools generally require some setup before you can use them.

This appendix describes the information you need to set up third-party tools to work with HS3. It also contains specific instructions for setting up s3curl to work with HS3.

General setup information for third-party tools

Typically, when configuring a third-party tool to work with HS3, you need to provide this information:

- **Service point** — This is the fully qualified hostname of the tenant in the context of which you plan to create and manage buckets and the objects in them. For example, if the tenant name is europe and the HCP system domain name is hcp.example.com, you specify the service point like this:

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

- **Access key** — This is the Base64-encoded username for the HCP user account you want to use for authenticated access to HCP.

- **Secret key** — This is the MD5-hashed password for the above HCP user account.

s3curl setup

The examples in this book use **s3curl**, which is a freely available open-source tool that you can install on your HS3 client computer. You can download **s3curl** from http://aws.amazon.com/code/128.

After installing **s3curl**, you need to configure it to work in your environment by providing the information described in "General setup information for third-party tools" above.

Specifying a access key and secret key

To specify the access key and secret key for an HCP user account, create a a file named .s3curl in your home directory. In this file, provide the necessary information in this format:

```
%awsSecretAccessKeys = (
  # HCP accounts
  account-name => {
    id => 'access-key',
    key => 'secret-key',
  },
);
```

account-name is the name to use in the **id** parameter in your **s3curl** commands.

For example:

```
%awsSecretAccessKeys = (
    # HCP accounts
    lgreen => {
        id => 'bGdyZWVu',
        key => '35dc4c4aa08fe0deab7e292e00eb8e97',
    },
);
```

To specify the access key and secret key for additional user accounts, repeat this portion of the file for each account:

```
account-name => {
  id => 'access-key',
  key => 'secret-key',
},
```

Here’s a sample .s3curl file that specifies the access key and secret key for each of two user accounts:

```
%awsSecretAccessKeys = (
    # HDS account
    lgreen => {
        id => 'bGdyZWVu',
        key => '35dc4c4aa08fe0deab7e292e00eb8e97',
    },
    mwhite => {
        id => 'bXdoaXRl',
        key => 'ad49ce36d0cec9634ef63b24151be0fb',
    },
);
```

Specifying a tenant

You specify the tenant you’re using as the HS3 service point in the s3curl.pl file, which is in the s3curl directory. In this file, find the line that begins like this:

```
my @endpoints = (
```

Between the opening parenthesis shown above and the first closing parenthesis that follows it, add the tenant URL in this format:

```
'tenant-name.hcp-domain-name',
```

For example:

```
my @endpoints = ('europe.hcp.example.com',);
```

To specify additional tenants, follow this example:

```
my @endpoints = ('europe.hcp.example.com',
                 'america.hcp.example.com',
                 'africa.hcp.example.com',);
```

D

Sample Java application

This appendix contains a sample Java application that uses the HCP HS3 API and the Amazon S3 SDK to perform a series of operations in HCP.

Assumptions

The application makes these assumptions:

- The HCP system has a tenant named europe.
- The tenant has a user account with username lgreen and password p4ssw0rd. The sample application uses the credentials for this account to access HCP.
- By default, versioning is disabled for new buckets.
- The local file system has folders named input and output that are located in the current working folder for the application.
- The input folder contains two files, Q4_2012.ppt and Q3_2012.ppt.

What the application does

The sample application shown in this appendix uses the HCP HS3 API to:

1. Create a bucket named finance in the context of the tenant named europe (the service point)
2. List the buckets for the europe tenant that are owned by the user lgreen
3. Add an ACL to the finance bucket

4. Store an object named quarterly_rpts/Q4_2012.ppt in the finance bucket, associating custom metadata with the object in the process
5. Store an object named quarterly_rpts/Q3_2012.ppt in the finance bucket
6. Retrieve the object named quarterly_rpts/Q4_2012.ppt and write its content to a new file on the local file system
7. Add an ACL to the object named quarterly_rpts/Q4_2012.ppt
8. Check whether the content of the object named quarterly_rpts/Q3_2012.ppt has changed and, if it has, retrieve the object and write its content to a new file on the local file system
9. Delete the quarterly_rpts/Q4_2012.ppt and quarterly_rpts/Q3_2012.ppt objects from the finance bucket
10. Delete the quarterly_rpts folder from the finance bucket (HCP created this folder automatically when the first object was stored)
11. Delete the finance bucket

Required libraries

To run the sample application presented in this appendix, you need to have installed these Java libraries:

- Amazon Java SDK 1.4.1, downloadable from:

  http://aws.amazon.com/sdkforjava/

- Apache Commons Codec 1.7, downloadable from:

  http://commons.apache.org/proper/commons-codec/download_codec.cgi

- Apache Commons Logging 1.1.2, downloadable from:

  http://commons.apache.org/proper/commons-logging/download_logging.cgi

- Apache HttpClient 4.2.3, downloadable from:

  http://hc.apache.org/downloads.cgi

- Apache HttpCore 4.2.4, downloadable from:

  http://hc.apache.org/downloads.cgi

Java application

Here’s the sample Java application.

```
/**
 * This sample Java application shows how to use the HCP HS3 API, which is
 * compatible with Amazon S3. The application uses the Amazon S3 SDK.
 */
package com.hds.hcp.examples;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.ClientConfiguration;
import com.amazonaws.Protocol;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.AccessControlList;
import com.amazonaws.services.s3.model.CanonicalGrantee;
import com.amazonaws.services.s3.model.EmailAddressGrantee;
import com.amazonaws.services.s3.model.GetObjectRequest;
import com.amazonaws.services.s3.model.ListObjectsRequest;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.Permission;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.model.Bucket;
import com.amazonaws.services.s3.model.PutObjectResult;
import com.amazonaws.services.s3.model.S3Object;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.s3.model.S3ObjectInputStream;
import com.amazonaws.services.s3.model.S3ObjectSummary;

public class HS3SampleApp {

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

        /*
         * Initialize access credentials for the HS3 client.
         */
```

```
// base64 of HCP user name: "lgreen"
String accessKey = "bGdyZWVu";
// md5 of HCP user password: "p4ssw0rd"
String secretKey = "2a9d119df47ff993b662a8ef36f9ea20";

/*
 * Set up the client configuration to allow for 200 max HTTP
 * connections, as this is an HCP best practice.
 */
ClientConfiguration myClientConfig = new ClientConfiguration();
myClientConfig.setMaxConnections(200);

/*
 * Use the HTTP protocol to communicate with HCP.  For you to use HTTPS,
 * HCP may need to have a trusted SSL server certificate.
 */
myClientConfig.setProtocol(Protocol.HTTP);

/*
 * Build the hs3Client to be used for communication with HCP.
 */
AmazonS3 hs3Client = new AmazonS3Client(
                             new BasicAWSCredentials(accessKey, secretKey),
                             myClientConfig);

// Set up the service point to be the tenant in HCP.
hs3Client.setEndpoint("europe.hcp.example.com");

/*
 * Now that the hs3Client is created for HCP usage, proceed with some
 * operations.
 */
String bucketName = "finance";

try {
    /*
     * Create a new bucket. With HCP, the bucket name does not need
     * to be globally unique. It needs to be unique only within the HCP
     * service point (that is, the HCP tenant).
     */
    System.out.println("Creating bucket " + bucketName + "\n");
    hs3Client.createBucket(bucketName);

    /*
     * List the buckets you own at the service point.
     */
    System.out.println("Buckets:");
    for (Bucket bucket : hs3Client.listBuckets()) {
        System.out.println(" * " + bucket.getName());
    }
    System.out.println();
```

```
/*
 * Add an ACL to the bucket to give read to a user with the
 * specified user ID.
 */
AccessControlList bucketACL = hs3Client.getBucketAcl(bucketName);
bucketACL.grantPermission(
        new CanonicalGrantee("7370bb2d-033c-4f05-863e-35a4eaf1d739"),
                             Permission.Read );
hs3Client.setBucketAcl(bucketName, bucketACL);

/*
 * Upload a couple of objects to the bucket from files on the local file
 * system.
 */
String objectNamePrefix = "quarterly_rpts/";

// Setup metadata for first object
String firstFileName = "input/Q4_2012.ppt";
ObjectMetadata metadata = new ObjectMetadata();
metadata.addUserMetadata("Author", "P.D. Gray");
metadata.addUserMetadata("Audit_Date", "2013-02-23");
// Content-Length must be set because the application will use an
//  InputStream during the PUT. Otherwise, the whole file would be
//  will be read into memory, which could cause the application to
//  run out of memory.
metadata.setContentLength(new File(firstFileName).length());

System.out.println("Uploading first object to HCP from a file\n");
String firstObjectName = objectNamePrefix + "Q4_2012.ppt";
hs3Client.putObject(new PutObjectRequest(
                                      bucketName,
                                      firstObjectName,
                                      new FileInputStream(firstFileName),
                                      metadata));

// Write a second object without metadata. Also collect its ETag for
// later usage.
System.out.println("Uploading second object to HCP from a file\n");
String secondObjectName = objectNamePrefix + "Q3_2012.ppt";
PutObjectResult result = hs3Client.putObject(
                                new PutObjectRequest(
                                      bucketName,
                                      secondObjectName,
                                      new File("input/Q3_2012.ppt")));
String secondObjectEtag = result.getETag();

/*
 * List objects in the bucket with prefix quarterly_rpts/Q.
 * The bucket listing is limited to 1,000 items per request.
 * Be sure to check whether the returned listing has been
 * truncated. If it has, retrieve additional results by using
 * the AmazonS3.listNextBatchOfObjects(...) operation.
 */
```

```
System.out.println("Objects:");
ObjectListing objectListing = hs3Client.listObjects(
                                new ListObjectsRequest()
                                      .withBucketName(bucketName)
                                      .withPrefix(objectNamePrefix
                                                  + "Q"));
for (S3ObjectSummary objectSummary
      : objectListing.getObjectSummaries()) {
    System.out.println(" * " + objectSummary.getKey() + "  " +
                       "(size = " + objectSummary.getSize() + ")");
}
System.out.println();

/*
 * Download an object. When you download an object, you get all
 * the object metadata and a stream from which to read the object
 * content.
 */
System.out.println("Downloading the first object\n");

S3Object firstObject = hs3Client.getObject(
                                  new GetObjectRequest(bucketName,
                                                       firstObjectName));

// Write the content to a file named Q4_2012.ppt in the output folder.
S3ObjectInputStream responseStream = firstObject.getObjectContent();
FileOutputStream dataFile = new FileOutputStream("output/Q4_2012.ppt");

// Keep reading bytes until the end of stream is reached.
byte buffer[] = new byte[2048];
int readSize;
while (-1 != (readSize = responseStream.read(buffer))) {
    dataFile.write(buffer, 0, readSize);
}

dataFile.close();

/*
 * Add an ACL to the first object to give full control to the user with
 * the username rsilver. HCP will look up the user ID based on the
 * username.
 */
AccessControlList objectACL = hs3Client.getObjectAcl(bucketName,
                                                     firstObjectName);
objectACL.grantPermission(new EmailAddressGrantee("rsilver"),
                          Permission.FullControl);
hs3Client.setObjectAcl(bucketName, firstObjectName, objectACL);

/*
 * Perform a conditional download of object. This will get the object only
 * if it doesn't match the ETag we received when storing the object.
 */
System.out.println("Checking the second object");
```

```
        GetObjectRequest conditionalRequest
             = new GetObjectRequest(bucketName, secondObjectName)
                         .withNonmatchingETagConstraint(secondObjectEtag);
        S3Object conditionalObject = hs3Client.getObject(conditionalRequest);
        if (null == conditionalObject) {
            System.out.println("  The object did not change; not downloaded.\n");
        } else {
            // The object has changed, download it to a new file.

            System.out.println(
                "  The object changed; downloading new revision\n");

            S3ObjectInputStream refreshResponseStream
                                    = conditionalObject.getObjectContent();
            FileOutputStream dataFile2
                               = new FileOutputStream("output/Q3_2012_Rev2.ppt");

            // Keep reading bytes until the end of stream is reached.
            byte readBuffer[] = new byte[2048];
            int conditionalReadSize;
            while (-1 != (conditionalReadSize
                                = refreshResponseStream.read(readBuffer))) {
                dataFile2.write(readBuffer, 0, conditionalReadSize);
            }
            dataFile2.close();
        }

        /*
         * Delete the objects.
         */
        System.out.println(
            "Deleting the objects created by this sample application\n");
        hs3Client.deleteObject(bucketName, firstObjectName);
        hs3Client.deleteObject(bucketName, secondObjectName);

        /*
         * Delete the folder.
         */
        System.out.println(
            "Deleting the folder created when the first object was stored\n");
        hs3Client.deleteObject(bucketName, objectNamePrefix);

        /*
         * Delete the bucket.
         */
        System.out.println("Deleting the finance bucket\n");
        hs3Client.deleteBucket(bucketName);

    } catch (AmazonServiceException ase) {
        System.out.println(
            "Caught an AmazonServiceException, which means the request made it "
                + "to HCP but was rejected for some reason.");
        System.out.println("Error Message:    " + ase.getMessage());
```

```
            System.out.println("HTTP Status Code: " + ase.getStatusCode());
            System.out.println("AWS Error Code:   " + ase.getErrorCode());
            System.out.println("Error Type:       " + ase.getErrorType());
            System.out.println("Request ID:       " + ase.getRequestId());
        } catch (AmazonClientException ace) {
            System.out.println(
                "Caught an AmazonClientException, which means the client encountered "
                    + "a serious internal problem while trying to communicate with "
                    + "HCP via HS3, such as not being able to access the network.");
            System.out.println("Error Message: " + ace.getMessage());
        } catch (IOException ioe) {
            System.out.println(
                "Caught an IOException while trying to create an object or read "
                    + "from an internal buffer.");
            System.out.println("Error Message: " + ioe.getMessage());
        }
    }
}
```

E

HCP HS3 XML schema

This appendix contains the version of the Amazon S3 schema that HCP uses for validating XML in HS3 requests and responses.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
 xmlns:tns="http://s3.amazonaws.com/doc/2006-03-01/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 targetNamespace="http://s3.amazonaws.com/doc/2006-03-01/">

    <xsd:element name="CreateBucket">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Bucket" type="xsd:string"/>
                <xsd:element name="AccessControlList" type="tns:AccessControlList" minOccurs="0"/>
                <xsd:element name="AWSAccessKeyId" type="xsd:string" minOccurs="0"/>
                <xsd:element name="Timestamp" type="xsd:dateTime" minOccurs="0"/>
                <xsd:element name="Signature" type="xsd:string" minOccurs="0"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="MetadataEntry">
        <xsd:sequence>
            <xsd:element name="Name" type="xsd:string"/>
            <xsd:element name="Value" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:element name="CreateBucketResponse">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="CreateBucketReturn" type="tns:CreateBucketResult"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="Status">
        <xsd:sequence>
            <xsd:element name="Code" type="xsd:int"/>
            <xsd:element name="Description" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="Result">
        <xsd:sequence>
            <xsd:element name="Status" type="tns:Status"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="CreateBucketResult">
        <xsd:sequence>
        <xsd:element name="BucketName" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:element name="DeleteBucket">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Bucket" type="xsd:string"/>
            <xsd:element name="AWSAccessKeyId" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Timestamp" type="xsd:dateTime" minOccurs="0"/>
            <xsd:element name="Signature" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Credential" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="DeleteBucketResponse">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="DeleteBucketResponse" type="tns:Status"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="BucketLoggingStatus">
    <xsd:sequence>
        <xsd:element name="LoggingEnabled" type="tns:LoggingSettings" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="LoggingSettings">
    <xsd:sequence>
        <xsd:element name="TargetBucket" type="xsd:string"/>
        <xsd:element name="TargetPrefix" type="xsd:string"/>
        <xsd:element name="TargetGrants" type="tns:AccessControlList" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:element name="GetBucketLoggingStatus">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Bucket" type="xsd:string"/>
            <xsd:element name="AWSAccessKeyId" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Timestamp" type="xsd:dateTime" minOccurs="0"/>
            <xsd:element name="Signature" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Credential" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="GetBucketLoggingStatusResponse">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="GetBucketLoggingStatusResponse"
              type="tns:BucketLoggingStatus"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="SetBucketLoggingStatus">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Bucket" type="xsd:string"/>
            <xsd:element name="AWSAccessKeyId" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Timestamp" type="xsd:dateTime" minOccurs="0"/>
            <xsd:element name="Signature" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Credential" type="xsd:string" minOccurs="0"/>
            <xsd:element name="BucketLoggingStatus" type="tns:BucketLoggingStatus"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="SetBucketLoggingStatusResponse">
    <xsd:complexType>
        <xsd:sequence/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="GetObjectAccessControlPolicy">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Bucket" type="xsd:string"/>
            <xsd:element name="Key" type="xsd:string"/>
            <xsd:element name="AWSAccessKeyId" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Timestamp" type="xsd:dateTime" minOccurs="0"/>
            <xsd:element name="Signature" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Credential" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="GetBucketAccessControlPolicy">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Bucket" type="xsd:string"/>
            <xsd:element name="AWSAccessKeyId" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Timestamp" type="xsd:dateTime" minOccurs="0"/>
            <xsd:element name="Signature" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Credential" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:complexType abstract="true" name="Grantee"/>

<xsd:complexType name="User" abstract="true">
    <xsd:complexContent>
        <xsd:extension base="tns:Grantee"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="AmazonCustomerByEmail">
    <xsd:complexContent>
        <xsd:extension base="tns:User">
            <xsd:sequence>
                <xsd:element name="EmailAddress" type="xsd:string"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="CanonicalUser">
    <xsd:complexContent>
        <xsd:extension base="tns:User">
            <xsd:sequence>
                <xsd:element name="ID" type="xsd:string"/>
                <xsd:element name="DisplayName" type="xsd:string" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="Group">
    <xsd:complexContent>
        <xsd:extension base="tns:Grantee">
            <xsd:sequence>
                <xsd:element name="URI" type="xsd:string"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="Permission">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="READ"/>
        <xsd:enumeration value="WRITE"/>
        <xsd:enumeration value="READ_ACP"/>
        <xsd:enumeration value="WRITE_ACP"/>
```

```
            <xsd:enumeration value="FULL_CONTROL"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="StorageClass">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="STANDARD"/>
            <xsd:enumeration value="REDUCED_REDUNDANCY"/>
            <xsd:enumeration value="UNKNOWN"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="Grant">
        <xsd:sequence>
            <xsd:element name="Grantee" type="tns:Grantee"/>
            <xsd:element name="Permission" type="tns:Permission"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="AccessControlList">
        <xsd:sequence>
            <xsd:element name="Grant" type="tns:Grant" minOccurs="0" maxOccurs="100"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:element name="CreateBucketConfiguration">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="LocationConstraint" type="tns:LocationConstraint"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="LocationConstraint">
        <xsd:simpleContent>
            <xsd:extension base="xsd:string"/>
        </xsd:simpleContent>
    </xsd:complexType>

    <xsd:element name="AccessControlPolicy">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="Owner" type="tns:CanonicalUser" minOccurs="0"/>
                    <xsd:element name="AccessControlList" type="tns:AccessControlList"/>
                </xsd:sequence>
            </xsd:complexType>
    </xsd:element>

    <xsd:element name="SetObjectAccessControlPolicy">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Bucket" type="xsd:string"/>
            <xsd:element name="Key" type="xsd:string"/>
            <xsd:element name="AccessControlList" type="tns:AccessControlList"/>
            <xsd:element name="AWSAccessKeyId" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Timestamp" type="xsd:dateTime" minOccurs="0"/>
            <xsd:element name="Signature" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Credential" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="SetObjectAccessControlPolicyResponse">
    <xsd:complexType>
        <xsd:sequence/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="SetBucketAccessControlPolicy">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Bucket" type="xsd:string"/>
            <xsd:element name="AccessControlList" type="tns:AccessControlList" minOccurs="0"/>
            <xsd:element name="AWSAccessKeyId" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Timestamp" type="xsd:dateTime" minOccurs="0"/>
            <xsd:element name="Signature" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Credential" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="SetBucketAccessControlPolicyResponse">
    <xsd:complexType>
        <xsd:sequence/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="GetObject">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Bucket" type="xsd:string"/>
            <xsd:element name="Key" type="xsd:string"/>
            <xsd:element name="GetMetadata" type="xsd:boolean"/>
            <xsd:element name="GetData" type="xsd:boolean"/>
            <xsd:element name="InlineData" type="xsd:boolean"/>
            <xsd:element name="AWSAccessKeyId" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Timestamp" type="xsd:dateTime" minOccurs="0"/>
            <xsd:element name="Signature" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Credential" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="GetObjectResponse">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="GetObjectResponse" type="tns:GetObjectResult"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="GetObjectResult">
    <xsd:complexContent>
        <xsd:extension base="tns:Result">
            <xsd:sequence>
                <xsd:element name="Metadata" type="tns:MetadataEntry" minOccurs="0"
                  maxOccurs="unbounded"/>
                <xsd:element name="Data" type="xsd:base64Binary" nillable="true"/>
                <xsd:element name="LastModified" type="xsd:dateTime"/>
                <xsd:element name="ETag" type="xsd:string"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="GetObjectExtended">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Bucket" type="xsd:string"/>
            <xsd:element name="Key" type="xsd:string"/>
            <xsd:element name="GetMetadata" type="xsd:boolean"/>
            <xsd:element name="GetData" type="xsd:boolean"/>
            <xsd:element name="InlineData" type="xsd:boolean"/>
            <xsd:element name="ByteRangeStart" type="xsd:long" minOccurs="0"/>
            <xsd:element name="ByteRangeEnd" type="xsd:long" minOccurs="0"/>
            <xsd:element name="IfModifiedSince" type="xsd:dateTime" minOccurs="0"/>
            <xsd:element name="IfUnmodifiedSince" type="xsd:dateTime" minOccurs="0"/>
            <xsd:element name="IfMatch" type="xsd:string" minOccurs="0" maxOccurs="100"/>
            <xsd:element name="IfNoneMatch" type="xsd:string" minOccurs="0"
              maxOccurs="100"/>
            <xsd:element name="ReturnCompleteObjectOnConditionFailure" type="xsd:boolean"
              minOccurs="0"/>
            <xsd:element name="AWSAccessKeyId" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Timestamp" type="xsd:dateTime" minOccurs="0"/>
            <xsd:element name="Signature" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Credential" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="GetObjectExtendedResponse">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="GetObjectResponse" type="tns:GetObjectResult"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="PutObject">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Bucket" type="xsd:string"/>
            <xsd:element name="Key" type="xsd:string"/>
            <xsd:element name="Metadata" type="tns:MetadataEntry" minOccurs="0"
              maxOccurs="100"/>
            <xsd:element name="ContentLength" type="xsd:long"/>
            <xsd:element name="AccessControlList" type="tns:AccessControlList" minOccurs="0"/>
            <xsd:element name="StorageClass" type="tns:StorageClass" minOccurs="0"/>
            <xsd:element name="AWSAccessKeyId" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Timestamp" type="xsd:dateTime" minOccurs="0"/>
            <xsd:element name="Signature" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Credential" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="PutObjectResponse">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="PutObjectResponse" type="tns:PutObjectResult"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="PutObjectResult">
    <xsd:sequence>
        <xsd:element name="ETag" type="xsd:string"/>
        <xsd:element name="LastModified" type="xsd:dateTime"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:element name="PutObjectInline">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Bucket" type="xsd:string"/>
            <xsd:element name="Key" type="xsd:string"/>
            <xsd:element minOccurs="0" maxOccurs="100" name="Metadata"
              type="tns:MetadataEntry"/>
            <xsd:element name="Data" type="xsd:base64Binary"/>
            <xsd:element name="ContentLength" type="xsd:long"/>
            <xsd:element name="AccessControlList" type="tns:AccessControlList" minOccurs="0"/>
            <xsd:element name="StorageClass" type="tns:StorageClass" minOccurs="0"/>
            <xsd:element name="AWSAccessKeyId" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Timestamp" type="xsd:dateTime" minOccurs="0"/>
            <xsd:element name="Signature" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Credential" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="PutObjectInlineResponse">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="PutObjectInlineResponse" type="tns:PutObjectResult"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="DeleteObject">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Bucket" type="xsd:string"/>
            <xsd:element name="Key" type="xsd:string"/>
            <xsd:element name="AWSAccessKeyId" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Timestamp" type="xsd:dateTime" minOccurs="0"/>
            <xsd:element name="Signature" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Credential" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="DeleteObjectResponse">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="DeleteObjectResponse" type="tns:Status"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="ListBucket">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Bucket" type="xsd:string"/>
            <xsd:element name="Prefix" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Marker" type="xsd:string" minOccurs="0"/>
            <xsd:element name="MaxKeys" type="xsd:int" minOccurs="0"/>
            <xsd:element name="Delimiter" type="xsd:string" minOccurs="0"/>
            <xsd:element name="AWSAccessKeyId" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Timestamp" type="xsd:dateTime" minOccurs="0"/>
            <xsd:element name="Signature" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Credential" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="ListEntry">
    <xsd:sequence>
        <xsd:element name="Key" type="xsd:string"/>
        <xsd:element name="LastModified" type="xsd:dateTime"/>
        <xsd:element name="ETag" type="xsd:string"/>
        <xsd:element name="Size" type="xsd:long"/>
        <xsd:element name="Owner" type="tns:CanonicalUser" minOccurs="0"/>
        <xsd:element name="StorageClass" type="tns:StorageClass"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="VersionEntry">
    <xsd:sequence>
        <xsd:element name="Key" type="xsd:string"/>
        <xsd:element name="VersionId" type="xsd:string"/>
        <xsd:element name="IsLatest" type="xsd:boolean"/>
        <xsd:element name="LastModified" type="xsd:dateTime"/>
        <xsd:element name="ETag" type="xsd:string"/>
        <xsd:element name="Size" type="xsd:long"/>
        <xsd:element name="Owner" type="tns:CanonicalUser" minOccurs="0"/>
        <xsd:element name="StorageClass" type="tns:StorageClass"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="DeleteMarkerEntry">
    <xsd:sequence>
        <xsd:element name="Key" type="xsd:string"/>
        <xsd:element name="VersionId" type="xsd:string"/>
        <xsd:element name="IsLatest" type="xsd:boolean"/>
        <xsd:element name="LastModified" type="xsd:dateTime"/>
        <xsd:element name="Owner" type="tns:CanonicalUser" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="PrefixEntry">
    <xsd:sequence>
        <xsd:element name="Prefix" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>
```

```
<xsd:element name="ListBucketResult">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Metadata" type="tns:MetadataEntry" minOccurs="0"
             maxOccurs="unbounded"/>
            <xsd:element name="Name" type="xsd:string"/>
            <xsd:element name="Prefix" type="xsd:string"/>
            <xsd:element name="Marker" type="xsd:string"/>
            <xsd:element name="NextMarker" type="xsd:string" minOccurs="0"/>
            <xsd:element name="MaxKeys" type="xsd:int"/>
            <xsd:element name="Delimiter" type="xsd:string" minOccurs="0"/>
            <xsd:element name="IsTruncated" type="xsd:boolean"/>
            <xsd:element name="Contents" type="tns:ListEntry" minOccurs="0"
             maxOccurs="unbounded"/>
            <xsd:element name="CommonPrefixes" type="tns:PrefixEntry" minOccurs="0"
             maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="ListVersionsResult">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Metadata" type="tns:MetadataEntry" minOccurs="0"
             maxOccurs="unbounded"/>
            <xsd:element name="Name" type="xsd:string"/>
            <xsd:element name="Prefix" type="xsd:string"/>
            <xsd:element name="KeyMarker" type="xsd:string"/>
            <xsd:element name="VersionIdMarker" type="xsd:string"/>
            <xsd:element name="NextKeyMarker" type="xsd:string" minOccurs="0"/>
            <xsd:element name="NextVersionIdMarker" type="xsd:string" minOccurs="0"/>
            <xsd:element name="MaxKeys" type="xsd:int"/>
            <xsd:element name="Delimiter" type="xsd:string" minOccurs="0"/>
            <xsd:element name="IsTruncated" type="xsd:boolean"/>
            <xsd:choice minOccurs="0" maxOccurs="unbounded">
                <xsd:element name="Version" type="tns:VersionEntry"/>
                <xsd:element name="DeleteMarker" type="tns:DeleteMarkerEntry"/>
            </xsd:choice>
            <xsd:element name="CommonPrefixes" type="tns:PrefixEntry" minOccurs="0"
             maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="ListAllMyBuckets">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="AWSAccessKeyId" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Timestamp" type="xsd:dateTime" minOccurs="0"/>
            <xsd:element name="Signature" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="ListAllMyBucketsEntry">
    <xsd:sequence>
        <xsd:element name="Name" type="xsd:string"/>
        <xsd:element name="CreationDate" type="xsd:dateTime"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:element name="ListAllMyBucketsResult">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Owner" type="tns:CanonicalUser"/>
            <xsd:element name="Buckets" type="tns:ListAllMyBucketsList"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="ListAllMyBucketsList">
    <xsd:sequence>
        <xsd:element name="Bucket" type="tns:ListAllMyBucketsEntry" minOccurs="0"
         maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:element name="PostResponse">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Location" type="xsd:anyURI"/>
            <xsd:element name="Bucket" type="xsd:string"/>
            <xsd:element name="Key" type="xsd:string"/>
            <xsd:element name="ETag" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:simpleType name="MetadataDirective">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="COPY"/>
        <xsd:enumeration value="REPLACE"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:element name="CopyObject">
    <xsd:complexType>
        <xsd:sequence>
```

```
            <xsd:element name="SourceBucket" type="xsd:string"/>
            <xsd:element name="SourceKey" type="xsd:string"/>
            <xsd:element name="DestinationBucket" type="xsd:string"/>
            <xsd:element name="DestinationKey" type="xsd:string"/>
            <xsd:element name="MetadataDirective" type="tns:MetadataDirective"
             minOccurs="0"/>
            <xsd:element name="Metadata" type="tns:MetadataEntry" minOccurs="0"
             maxOccurs="100"/>
            <xsd:element name="AccessControlList" type="tns:AccessControlList" minOccurs="0"/>
            <xsd:element name="CopySourceIfModifiedSince" type="xsd:dateTime"
             minOccurs="0"/>
            <xsd:element name="CopySourceIfUnmodifiedSince" type="xsd:dateTime"
             minOccurs="0"/>
            <xsd:element name="CopySourceIfMatch" type="xsd:string" minOccurs="0"
             maxOccurs="100"/>
            <xsd:element name="CopySourceIfNoneMatch" type="xsd:string" minOccurs="0"
             maxOccurs="100"/>
            <xsd:element name="StorageClass" type="tns:StorageClass" minOccurs="0"/>
            <xsd:element name="AWSAccessKeyId" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Timestamp" type="xsd:dateTime" minOccurs="0"/>
            <xsd:element name="Signature" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Credential" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="CopyObjectResult">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="LastModified" type="xsd:dateTime"/>
            <xsd:element name="ETag" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="RequestPaymentConfiguration">
    <xsd:sequence>
        <xsd:element name="Payer" type="tns:Payer" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="Payer">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="BucketOwner"/>
        <xsd:enumeration value="Requester"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:element name="VersioningConfiguration">
```

```
        <xsd:complexType>
        <xsd:sequence>
                <xsd:element name="Status" type="tns:VersioningStatus" minOccurs="0" />
                <xsd:element name="MfaDelete" type="tns:MfaDeleteStatus" minOccurs="0" />
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:simpleType name="MfaDeleteStatus">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Enabled"/>
            <xsd:enumeration value="Disabled"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="VersioningStatus">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Enabled"/>
            <xsd:enumeration value="Suspended"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="NotificationConfiguration">
        <xsd:sequence>
            <xsd:element name="TopicConfiguration" minOccurs="0" maxOccurs="unbounded"
              type="tns:TopicConfiguration"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="TopicConfiguration">
        <xsd:sequence>
            <xsd:element name="Topic" minOccurs="1" maxOccurs="1" type="xsd:string"/>
            <xsd:element name="Event" minOccurs="1" maxOccurs="unbounded" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>

</xsd:schema>
```

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 either in an XML request body or as request headers.

access key

The Base64 encoding of the username for a user account.

access protocol

See namespace access protocol.

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).

Amazon Web Services

The set of infrastructure and application services that make up the Amazon cloud computing platform. These services include a user authentication method that is also implemented in HCP.

annotation

A discrete unit of custom metadata.

anonymous access

A method of access to a namespace wherein the user or application gains access without presenting any credentials. *See also* authenticated access.

appendable object

An object to which data can be added after it has been successfully stored. Appending data to an object does not modify the original fixed-content data, nor does it create a new version of the object. Once the new data is added to the object, that data also cannot be modified.

Appendable objects are supported only with the CIFS and NFS protocols.

authenticated access

A method of access to the HCP system wherein the user or application presents credentials to gain access. *See also* anonymous access.

authentication

See user authentication.

AWS

See Amazon Web Services.

B

bucket

The HS3 term for a namespace.

C

canned ACL

A predefined set of grants of permissions.

CIFS

Common Internet File System. One of the namespace access protocols supported by HCP. CIFS lets Windows clients access files on a remote computer as if the files were part of the local file system.

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 access permission mask

A set of permissions that determine which of these operations are allowed in a namespace: read (including read ACL), write (including write ACL and change owner), delete, purge, privileged operations, and search. Data access permission masks are defined at the system, tenant, and namespace level. The effective permissions for a namespace are those that are allowed at all three levels.

deleted version

A version of an object that serves as an indicator that the object has been deleted. A deleted version of an object has a version ID but does not have any data or metadata.

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.

E

effective permissions

For a namespace, the permissions that are included in all three of the system-level, tenant-level, *and* namespace-level permission masks.

ETag

An identifier for the content of an object.

F

fixed-content data

A digital asset ingested into HCP and preserved in its original form as the core part of an object. Once stored, fixed-content data cannot be modified.

H

HCP

See Hitachi Content Platform (HCP).

HCP Data Migrator (HCP-DM)

An HCP utility that can transfer data from one location to another, delete data from a location, and change object metadata in a namespace. Each location can be a local file system, an HCP namespace, a default namespace, or an HCAP 2.*x* archive.

HCP-DM

See HCP Data Migrator (HCP-DM).

HCP metadata query API

See metadata query API.

HCP node

See node.

HCP search facility

One of the search facilities available for use with the HCP Search Console. This facility is integrated with HCP and works internally to perform searches and return results to the Search Console.

HDDS

See Hitachi Data Discovery Suite (HDDS).

HDDS search facility

One of the search facilities available for use with the HCP Search Console. This facility interacts with Hitachi Data Discovery Suite.

Hitachi Content Platform (HCP)

A distributed object-based 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.

Hitachi Data Discovery Suite (HDDS)

A Hitachi product that enables federated searches across multiple HCP systems and other supported systems.

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.

HS3 API

One of the namespace access protocols supported by HCP. HS3 is a RESTful, HTTP-based API that is compatible with Amazon S3. Using HS3, users and applications can create and manage buckets and bucket contents.

HTTP

HyperText Transfer Protocol. One of the namespace access protocols supported by HCP. In the context of namespace access, the HTTP protocol is also called the REST API.

HTTPS

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

I

index

An index of the objects in namespaces that is used to support search operations. The metadata query engine, HDDS search facility, and HCP search facilities each maintain a separate index..

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.

metadata query engine

One of the search facilities available for use with HCP. The metadata query engine works internally to perform searches and return results either through the metadata query API or to the HCP Metadata Query Engine Console (also known as the HCP Search Console).

Metadata Query Engine Console

The web application that provides interactive access to the HCP search functionality provided by the metadata query engine.

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.

namespace access protocol

A protocol that can be used to transfer data to and from namespaces in an HCP system. HCP supports the HTTP, HS3, WebDAV, CIFS, NFS, and SMTP protocols.

NFS

Network File System. One of the namespace access protocols supported by HCP. NFS lets clients access files on a remote computer as if the files were part of the local file system.

node

A server running HCP software and networked with other such servers to form an HCP system.

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 indexing and versioning.

object-based query

In the metadata query API, 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 folder 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

In the metadata query API, a query that searches not only for objects currently in the repository but also for information about objects that have been deleted, 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, and location (that is, the namespace and/or folder that contains the object).

P

permission

One of these:

- In a data access permission mask, the condition of allowing a specific type of operation to be performed in a namespace.
- In a user account, the granted ability to perform a specific type of operation in a given namespace.
- In an ACL associated with a bucket or object, the granted ability to perform a specific type of operation on the bucket or object.

permission mask

See data access permission mask.

privileged delete

A delete operation that works on an object regardless of whether the object is under retention, except if the object is on hold. This operation is available only to users and applications with explicit permission to perform it.

privileged purge

A purge operation that works on an object regardless of whether the object is under retention, except if the object is on hold. This operation is available only to users and applications with explicit permission to perform it.

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

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, as XML) 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.

REST API

One of the namespace access protocols supported by HCP. The REST API is also called the HTTP protocol.

retention hold

See hold.

retention period

The period of time during which an object cannot be deleted (except by means of a privileged delete).

retention setting

The property that determines the retention period for an object.

S

Search Console

The web application that provides interactive access to HCP search functionality. When the Search Console uses the HCP metadata query engine for search functionality, it is called the Metadata Query Engine Console.

secret key

The MD5 hash of the password for a user account.

service point

A tenant that serves as the point of access to HCP in an HS3 API request.

signature

A value calculated using specific elements of an HS3 request, including the secret key of the requester and a date and time.

SMTP

Simple Mail Transfer Protocol. The namespace access protocol HCP uses to receive and store email data directly from email servers.

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, 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

Unix

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

user account

A set of credentials that gives a user access to a given tenant and its namespaces.

user authentication

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

V

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.

W

WebDAV

Web-based Distributed Authoring and Versioning. One of the namespace access protocols supported by HCP. WebDAV is an extension of HTTP.

WORM

Write once, read many. A data storage property that protects the stored data from being modified or overwritten.

X

XML

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

Index

Symbols

.metapairs annotations
- about 18–20
- unexpected content 21

A

access control lists
- *See* ACLs

access keys
- AWS authentication 43
- HCP authentication 203

AccessControlList element in ACL XML 28
AccessControlPolicy element in ACL XML 28
AccessDenied error code 54
acl query parameter
- adding ACLs to buckets 73–74
- adding ACLs to objects 142
- retrieving bucket ACLs 79, 79
- retrieving object ACLs 148

ACL request body
- *See also* ACLs; adding ACLs to buckets
- about 27
- removing ACLs 32
- sample 31–32, 147
- XML format 28–31

ACL response body
- *See also* ACLs; retrieving bucket ACLs
- about 79
- formatting 80, 149
- sample 82–83, 151–152

ACLs
- *See also* ACL request body; adding ACLs to buckets; adding ACLs to objects
- about 2, 22–23
- adding to buckets 73–79
- adding to objects 141–148
- adding when creating buckets 60
- adding when storing objects 120
- canned 25
- copying objects 152
- enforcement 23
- grantees 24–25
- invalid in copy object requests 153
- invalid in create bucket requests 60
- invalid in store object requests 120
- nonexistent user accounts 73, 142
- permissions 23–24
- removing 32
- request headers, about 25–27
- request headers, example 77, 146
- retrieving for buckets 79–83
- retrieving for objects 148–152
- specifying canned 26
- specifying in individual grant headers 26–27
- specifying in request bodies 27–32
- specifying, about 25

Active Directory
- user accounts 9
- user authentication 37

AD
- *See* Active Directory

adding ACLs to buckets
- *See also* ACLs
- about 73
- examples 77–79
- HTTP status codes 76–77
- request headers 74–75
- request line 74
- response headers 75

adding ACLs to objects
- *See also* ACLs
- about 141–142
- examples 146–148
- HTTP status codes 144–146
- request headers 142–144
- request line 142
- response headers 144

all users
  ACL grantee 25
  anonymous access, AWS authentication 44
  anonymous access, HCP authentication 203
allocated space for buckets 34
annotations
  .metapairs with unexpected content 21
  .metapairs, about 18–20
  about 2, 18
anonymous access
  Authorization request header, AWS authentication 44
  Authorization request header, HCP authentication 203
anonymous users 10
  See also anonymous access
assumptions, sample Java application 209
authenticated users
  about 9
  ACL grantee 24–25
Authorization request header
  anonymous access, AWS authentication 44
  anonymous access, HCP authentication 203
  AWS authentication 42–43
  HCP authentication 203
AWS authentication 42–44

B

BadDigest error code 54
browse permission 10
Bucket element in bucket list XML 67
bucket list response body
  See also listing buckets
  formatting 66
  sample 70
bucket listing response body
  See also listing bucket contents
  about 91
  elements 102–106
  formatting 99
  sample 108–109, 109–110, 111–112, 113–115
  XML format for current items 100
  XML format for object versions 101
BucketAlreadyExists error code 54
BucketAlreadyOwnedByYou error code 54
BucketNotEmpty error code 54
buckets
  See also namespaces
  about 3
  ACLs 22–23
  adding ACls to 73–79
  allocated space 34
  changing owners 21–22
  checking existence 70–73
  checking versioning status 87–91
  creating 60–65
  data access permission masks 35
  deleting 115–117
  enabling/disabling versioning 83–87
  listing 65–70
  listing contents 91–115
  names 14
  namespace access protocols enabled for new 5
  owners 21–22, 60
  renaming 14
  retrieving ACLs for 79–83
  search enabled 34–35

C

Cache-Control request header 51
Cache-Control response header 53
canned ACLs
  about 25
  specifying 26
case sensitivity
  acl query parameter 74, 79, 142, 148
  bucket names 14
  delimiter query parameter 96
  key-marker query parameter 97
  marker query parameter 97
  max-keys query parameter 98
  object names 14
  percent-encoded values 17
  prefix query parameter 98
  request headers 49
  URLs 41
  versionId query parameter 133, 155, 169
  version-id-marker query parameter 98
  versioning query parameter 85, 88
  versions query parameter 98
change owner permission 11
changing
  bucket names 14
  bucket owners 21–22, 27
  object owners 22, 27
  passwords 45–46
checking bucket existence
  about 70
  example 72–73
  HTTP status codes 71–72
  request headers 71
  request line 70
  response headers 71

checking object existence
  about 131
  conditionally 132
  examples 138–141
  HTTP status codes 137–138
  information returned 131
  request headers 133–135
  request line 132–133
  response headers 135–137
checking versioning status
  See also versioning
  about 87
  example 90–91
  HTTP status codes 89–90
  request headers 88
  request line 88
  response headers 89
CIFS protocol
  about 5
  object names 16
common
  prefixes 92–93
  request headers 49–51
  response headers 52–53
CommonPrefixes element in bucket listing XML 102
concurrent writes to same object 185
conditionally
  checking object existence 132
  copying objects 153–154
  retrieving objects 168
connection failure handling 188
Connection request header 51
Connection response header 53
Content-Disposition request header 51
Content-Disposition response header 53
Content-Encoding request header 51
Content-Encoding response header 53
Content-Language request header 51
Content-Language response header 53
Content-Length request header
  adding ACLs to buckets 74
  adding ACLs to objects 142
  creating buckets 61
  creating folders 127, 128
  enabling/disabling versioning 85
  storing objects 122
Content-Length response header 52
Content-MD5 request header 121, 122
Content-Range response header 173
Contents element in bucket listing XML 102
Content-Type request header
  adding ACLs to buckets 74
  creating folders 127, 128
  enabling/disabling versioning 85
  storing objects 122
Content-Type response header
  about 52
  checking object existence 136
  checking versioning status 89
  copying objects 159
  listing bucket contents 99
  listing buckets 66
  retrieving bucket ACLs 80
  retrieving object ACLs 149
  retrieving objects 173
copy object response body
  about 153
  formatting 159
  sample 164, 165, 166
  XML format 160
copying objects
  about 152–153
  conditionally 153–154
  examples 162–166
  HTTP status codes 161–162
  invalid ACLs 153
  replacing ACLs 152
  replacing custom metadata 152, 153
  request headers 154–159
  request line 154
  response body 160
  response headers 159–160
  to themselves 153
CopyObjectResult element in object copy XML 160
creating buckets
  about 60
  examples 64–65
  HTTP status codes 62–63
  invalid ACLs 60
  request headers 61–62
  request line 60
  response headers 62
creating folders
  See also folders
  about 127
  example 130
  HTTP status codes 129–130
  request headers 128
  request line 128
  response headers 129
creating object versions 120
  See also storing objects
CreationDate element in bucket list XML 67

credentials
- about 9
- invalid 44
- providing with HS3 requests 37
- specifying for AWS authentication 42–43
- specifying for HCP authentication 203

custom metadata
- about 2, 18
- adding when storing objects 120
- maximum size 20
- operations allowed for objects under retention 21
- property names 19, 20
- replacing for existing objects 153, 165–166
- replacing for object copies 152
- retrieving 20, 131, 139–140
- storing 18–19

D

data access permission masks 35

data access permissions
- about 10–11
- granted by ACLs 23–24

Data Migrator 8–9

Date request header 50

Date response header 52

datetime values, format for Date and x-amz-date request headers 50

default retention setting 18

DELETE bucket 190
- *See also* deleting buckets

DELETE method
- deleting buckets 115
- deleting objects 178

DELETE object 194
- *See also* deleting objects

delete permission 11

deleted versions
- about 34
- in bucket listings 91

DeleteMarker element in bucket listing XML 102

deleting buckets
- about 115
- example 117
- HTTP status codes 116–117
- request headers 115
- request line 115
- response headers 116

deleting objects
- about 178
- examples 180–182
- HTTP status codes 179–180
- request headers 179
- request line 178
- response headers 179
- under repair 186
- versioning disabled 34
- versioning enabled 34

Delimiter element in bucket listing XML 102

delimiter query parameter
- about 92–93
- example 107–109
- valid values 96

disabling SSL certificate verification 42

disabling versioning
- *See* enabling/disabling versioning

DisplayName element
- ACL XML 29
- bucket list XML 68
- bucket listing XML 102

downloading s3curl 12

E

EmailAddress element in ACL XML 29

empty objects 186

Enabled versioning status 84

enabling/disabling versioning
- *See also* versioning
- about 33, 83–84
- example 87
- HTTP status codes 86
- request headers 85
- request line 85
- response headers 85–86

encryption of objects 121

enforcing ACLs 23

ensuring object integrity 121

error codes 53–57

error response body 53

ETag element
- bucket listing XML 103
- object copy XML 160

ETag response header
- *See also* ETags
- checking object existence 136
- copying objects 159
- creating folders 129
- retrieving objects 173
- storing objects 120, 123

ETags 120

examples
- about 12
- ACL request body 31–32
- ACL response body 82–83, 151–152
- adding an ACL to a bucket by specifying individual grants 77–78

adding an ACL to a bucket by using an ACL request body 78–79
adding an ACL to an object by using a canned ACL 146
adding an ACL to an object by using an ACL request body 147–148
bucket list response body 70
bucket listing response body 108–109, 109–110, 111–112, 113–115
checking the existence of a bucket 72–73
checking the existence of a folder 141
checking the existence of an object 138–139
checking the versioning status of a bucket 90–91
checking whether an object has been modified 140
conditionally copying an object from one bucket to another 163–164
conditionally retrieving an object 176
copy object response body 164, 165, 166
creating a bucket 64
creating a bucket with an ACL 64–65
creating a folder 130
deleting a bucket 117
deleting a folder 181–182
deleting an object 181
enabling versioning for a bucket 87
listing items a few at a time 109–112
listing the buckets you own 69–70
listing the items in a folder 107–109
listing the versions of an individual object 113–115
recovering an old version of an object 164–165
replacing custom metadata for an existing object 165–166
retrieving a bucket ACL 81–83
retrieving an object ACL 150–152
retrieving an old version of an object 176–177
retrieving custom metadata for an object version 139–140
retrieving part of an object 177–178
sample Java application 209–216
storing an object 125–126
storing an object with custom metadata and an ACL 126–127
versioning request body 87
versioning response body 91

Expect request header 122
Expires response header 53

F

failed connections, handling 188
failed PUT requests 185–186
folders
  See also objects
  about 15–16
  in bucket listings 91
  checking existence 131–141
  creating 127–130
  deleting 178–182
  names 15, 127
  structuring 184–185
  version IDs 33
formatting
  ACL response body 80, 149
  bucket list response body 66
  bucket listing response body 99
  versioning response body 88
formatting copy object response body 159
forward slashes
  creating folders 127
  in object names 15–16

G

GET bucket 190
  See also listing bucket contents
GET bucket acl 191
  See also retrieving bucket ACLs
GET bucket versioning 191
  See also checking versioning status
GET bucket versions 191
  See also listing bucket contents
GET method
  checking versioning status 87
  listing bucket contents 91
  listing buckets 65
  retrieving bucket ACLs 79
  retrieving object ACLs 148
  retrieving objects 166
GET object 195
  See also retrieving objects
GET object acl 196
  See also retrieving object ACLs
GET service 190
  See also listing buckets
Grant element in ACL XML 29
grantable permissions 24
Grantee element in ACL XML 30
grantees
  about 24–25
  in ACL request body 28–31
  in individual request headers 26–27

H

handling failed connections 188
HCP
  about 1–2
  access by hostname 38–39, 184
  access by IP address 39–40, 184
HCP authentication 203
HCP Data Migrator 8–9
HCP Search Console 7–8
HCP search facility 8
HCP user accounts 9
HCP-DM 8–9
HDDS search facility 8
HEAD bucket 192
  See also checking bucket existence
HEAD method
  checking bucket existence 70
  checking object existence 131
HEAD object 196
  See also checking object existence
Hitachi Content Platform
  See HCP
hold
  about 18
  object versions 33
Host request header 39–40, 50
hostnames
  HCP access by 38–39, 184
  mappings 40–41
hosts files 40–41
HS3 API
  about 4–5
  quick reference 190–200
  requests 48–51
  responses 51–57
  XML schema 218–231
hs3 interface identifier 203
HTTP protocol 5
HTTP status codes
  about 51–52
  adding ACLs to buckets 76–77
  adding ACLs to objects 144–146
  checking bucket existence 70, 71–72
  checking object existence 137–138
  checking versioning status 89–90
  copying objects 161–162
  creating buckets 62–63
  creating folders 129–130
  deleting buckets 116–117
  deleting objects 179–180
  enabling/disabling versioning 86
  listing bucket contents 106–107
  listing buckets 68–69
  retrieving bucket ACLs 81
  retrieving object ACLs 150
  retrieving objects 174–175
  storing objects 124–125

I

ID element
  ACL XML 30
  bucket list XML 68
  bucket listing XML 103
If-Match request header
  checking object existence 132, 133
  retrieving objects 168, 170
If-Modified-Since request header
  checking object existence 132
  retrieving objects 168, 171
If-None-Match request header
  checking object existence 132
  retrieving objects 168, 172
If-Unmodified-Since request header
  checking object existence 132
  retrieving objects 168, 172
indexing 8
individual grant headers 26–27
InternalError error code 55
invalid
  ACLs in copy object requests 153
  ACLs in create bucket requests 60
  ACLs in store object requests 120
  credentials 44
  query parameters 49
InvalidAccessKey error code 44
InvalidAccessKeyId error code 54
InvalidArgument error code 55
InvalidBucketName error code 55
InvalidBucketState error code 55
InvalidRange error code 55
InvalidRequest error code 55
IP addresses, HCP access by 39–40, 184
IsLatest element in bucket listing XML 103
IsTruncated element in bucket listing XML 103

J

Java application, sample 209–216

K

Key element in bucket listing XML 103
KeyMarker element in bucket listing XML 104
key-marker query parameter
  about 95–96
  example 109–112
  valid values 97

KeyTooLong error code 55

L

LastModified element
  bucket listing XML 104
  object copy XML 160
Last-Modified response header
  checking object existence 136
  retrieving objects 173
libraries required for sample Java application 210–211
limiting
  listings of current items 92–94
  version listings 94–96
ListAllMyBucketsResult element in bucket list XML 68
ListBucketResult element in bucket listing XML 104
listing bucket contents
  about 91–92
  delimiter query parameter 92–93
  examples 107–115
  HTTP status codes 106–107
  key-marker query parameter 95–96
  marker query parameter 93–94
  max-keys query parameter 94
  prefix query parameter 94
  query parameters 96–98
  request headers 99
  request line 96–98
  response body 100–106
  response headers 99
  version-id-marker query parameter 95–96
  versions query parameter 91
listing buckets
  about 65
  example 69–70
  HTTP status codes 68–69
  request headers 66
  request line 65
  response body 67–68
  response headers 66
ListVersionsResult element in bucket listing XML 104
Location response header
  creating buckets 62
  enabling/disabling versioning 86

M

Mac OS X hosts file 40
MalformedXML error code 56
Marker element in bucket listing XML 104
marker query parameter
  about 93–94
  example 107–109
  valid values 97
maximum
  buckets owned by single user 22
  custom metadata size 20
  grants per ACL 23
  items in bucket listings 92
MaxKeys element in bucket listing XML 104
max-keys query parameter
  about 94
  examples 109–112
  valid values 98
metadata 2
metadata query API 6–7
metadata query engine 8
Metadata Query Engine Console 8
MetadataTooLarge error code 56
MethodNotAllowed error code 56
multithreading 186–187
multivalued custom metadata properties
  retrieving 20
  storing 19

N

Name element
  bucket list XML 68
  bucket listing XML 104
names
  bucket 14
  changing bucket 14
  custom metadata properties 19, 20
  folders 15, 127
  objects 14–17, 120
namespace access protocols
  about 5
  new buckets 5
Namespace Browser
  about 6
  changing passwords 45–46
namespaces 3
  See also buckets
network bandwidth, saving 121
NextKeyMarker element in bucket listing XML 96, 105
NextVersionIdMarker element in bucket listing XML 96, 105
NFS protocol
  about 5
  object names 16
nodes
  about 3

selection 184
targeting requests to 39–40
non-ASCII, nonprintable characters in object names 16
NoSuchBucket error code 56
NoSuchKey error code 56
NotImplemented error code 56

O

object versions
See also objects; versioning
about 33
creating 32–33, 120
deleted 34
listing 91
pruning 34
recovering 164–165
object-based queries 6–7
objects
See also object versions
about 2–3
ACLs 23
adding ACLs to 141–148
changing owners 22
checking existence 131–141
copying 152–166
deleting 178–181
deleting under repair 186
deleting with versioning disabled 34
deleting with versioning enabled 34
empty 186
encryption 121
integrity when storing 121
names 14–17, 120
organizing 15
owners 22, 120, 153
replacing custom metadata 165–166
retrieving 166–178
retrieving ACLs for 148–152
retrieving partial 166–168
sample 12
storing 120–127
version IDs 33
versioning 32–34
versions 3
writing concurrently to same 185
OperationAborted error code 56
operation-based queries 7
operations, HS3 4–5
organizing objects 15
OutOfSpace error code 56
overriding response headers 169
Owner element
ACL XML 30
bucket list XML 68
bucket listing XML 105
owners
buckets 21–22, 60
changing 27
listing buckets 65
object copies 153
objects 22, 120
permissions 24

P

partial objects, retrieving 166–168
passwords
about 46
changing 45–46
percent-encoding special characters in object names 16–17
Permission element in ACL XML 31
permissions
ACLs 23–24
data access 10–11
grantable 24
new buckets 60
persistent connections 187
Pragma request header 51
Pragma response header 53
PreconditionFailed error code 56
Prefix element in bucket listing XML 105
prefix query parameter
about 94
examples 107–109, 113–115
valid values 98
presigned URLs
about 42
user authentication 43
privileged permission 11
pruning 34
purge permission 11
PUT bucket 192
See also creating buckets
PUT bucket acl 193
See also adding ACLs to buckets
PUT bucket versioning 194
See also enabling/disabling versioning
PUT method
adding ACLs to buckets 73
creating buckets 60
creating folders 127
enabling/disabling versioning 83
storing objects 120

PUT object 197
    See also storing objects
PUT object acl 198
    See also adding ACLs to objects
PUT object copy 199–200
    See also copying objects

Q

query parameters
    acl, adding ACLs to buckets 73–74
    acl, adding ACLs to objects 142
    acl, retrieving bucket ACLs 79, 79
    acl, retrieving object ACLs 148
    delimiter 92–93, 96
    invalid 49
    key-marker 95–96, 97
    marker 93–94, 97
    max-keys 94, 98
    prefix 94, 98
    in request lines 49
    versionId, checking object existence 131, 133
    versionId, copying objects 152, 155
    versionId, retrieving objects 166, 169–170
    version-id-marker 95–96, 98
    versioning, checking versioning status 87–88
    versioning, enabling/disabling versioning 83, 85
    versions 91
quick reference 190–200
quotation marks with URLs 42

R

Range request header 166–168, 172
read ACL permission 11
read permission 10–11
recovering object versions 164–165
removing ACLs 32
renaming buckets 14
replacing custom metadata
    existing objects 153, 165–166
    object copies 152
request bodies
    ACL 27–32
    versioning 84
request headers
    adding ACLs to buckets 74–75
    adding ACLs to objects 142–144
    Authorization, AWS authentication 42–43
    Authorization, HCP authentication 203
    Cache-Control 51
    checking bucket existence 71
    checking object existence 133–135
    checking versioning status 88
    Connection 51
    Content-Disposition 51
    Content-Encoding 51
    Content-Language 51
    Content-Length 122
    Content-Length, adding ACLs to buckets 74
    Content-Length, adding ACLs to objects 142
    Content-Length, creating buckets 61
    Content-Length, creating folders 128
    Content-Length, enabling/disabling versioning 85
    Content-MD5 121, 122
    Content-Type 122
    Content-Type, adding ACLs to buckets 74
    Content-Type, creating folders 128
    Content-Type, enabling/disabling versioning 85
    copying objects 154–159
    creating buckets 61–62
    creating folders 128
    Date 50
    deleting buckets 115
    deleting objects 179
    enabling/disabling versioning 85
    Expect 122
    Host 39–40, 50
    If-Match, checking object existence 132, 133
    If-Match, retrieving objects 168, 170
    If-Modified-Since, checking object existence 132
    If-Modified-Since, retrieving objects 168, 171
    If-None-Match, checking object existence 132
    If-None-Match, retrieving objects 168, 172
    If-Unmodified-Since, checking object existence 132
    If-Unmodified-Since, retrieving objects 168, 172
    listing bucket contents 99
    listing buckets 66
    Pragma 51
    Range 166–168, 172
    response-cache-control 169, 172
    response-content-disposition 169, 172
    response-content-encoding 169, 172
    response-content-language 169, 172
    response-content-type 169, 172
    response-expires 169, 172
    retrieving bucket ACLs 80
    retrieving object ACLs 149

retrieving objects 170–172
storing objects 122–123
x-amz-acl 26
x-amz-copy-source 155
x-amz-copy-source-if-match 153–154, 155
x-amz-copy-source-if-modified-since 153–154, 156
x-amz-copy-source-if-none-match 153–154, 157
x-amz-copy-source-if-unmodified-since 153–154, 157
x-amz-date 51
x-amz-grant- 26–27
x-amz-meta- 18–19
x-amz-metadata-directive 153, 158
x-amz-server-side-encryption 123
x-amz-server-side-encryption, copying objects 158
x-amz-server-side-encryption, storing objects 121
x-hcp-pretty-print, checking versioning status 88
x-hcp-pretty-print, copying objects 159
x-hcp-pretty-print, listing bucket contents 99
x-hcp-pretty-print, listing buckets 66
x-hcp-pretty-print, retrieving bucket ACLs 80
x-hcp-pretty-print, retrieving object ACLs 149

request lines
about 48–49
adding ACLs to buckets 74
adding ACLs to objects 142
checking bucket existence 70
checking object existence 132–133
checking versioning status 88
copying objects 154
creating buckets 60
creating folders 128
deleting buckets 115
deleting objects 178
enabling/disabling versioning 85
listing bucket contents 96–98
listing buckets 65
retrieving bucket ACLs 79
retrieving object ACLs 148
retrieving objects 169–170
storing objects 121

required libraries, sample Java application 210–211

response bodies
ACL 79
bucket list 67–68
bucket listing 91, 100–106
copy object 160
error 53
versioning 87

response headers
adding ACLs to buckets 75
adding ACLs to objects 144
Cache-Control 53
checking bucket existence 71
checking object existence 135–137
checking versioning status 89
Connection 53
Content-Disposition 53
Content-Encoding 53
Content-Language 53
Content-Length 52
Content-Range 173
Content-Type 52
Content-Type, checking object existence 136
Content-Type, checking versioning status 89
Content-Type, copying objects 159
Content-Type, listing bucket contents 99
Content-Type, listing buckets 66
Content-Type, retrieving bucket ACLs 80
Content-Type, retrieving object ACLs 149
Content-Type, retrieving objects 173
copying objects 159–160
creating buckets 62
creating folders 129
Date 52
deleting buckets 116
deleting objects 179
enabling/disabling versioning 85–86
ETag 120
ETag, checking object existence 136
ETag, copying objects 159
ETag, creating folders 129
Etag, retrieving objects 173
ETag, storing objects 123
Expires 53
Last-Modified, checking object existence 136
Last-Modified, retrieving objects 173
listing bucket contents 99
listing buckets 66
Location, creating buckets 62
Location, enabling/disabling versioning 86
overriding 169
Pragma 53
retrieving bucket ACLs 80
retrieving object ACLs 149
retrieving objects 173–174
Server 52
storing objects 123–124
Transfer-Encoding 52

x-amz-copy-source-version-id 159
x-amz-delete-marker, checking object existence 136
x-amz-delete-marker, deleting objects 179
x-amz-meta-, checking object existence 136
x-amz-meta-, retrieving objects 173
x-amz-missing-meta, checking object existence 137
x-amz-missing-meta, retrieving objects 174
x-amz-server-side-encryption, copying objects 159
x-amz-server-side-encryption, storing objects 121, 124
x-amz-version-id, checking object existence 137
x-amz-version-id, copying objects 160
x-amz-version-id, creating folders 129
x-amz-version-id, deleting objects 179
x-amz-version-id, retrieving objects 174
x-amz-version-id, storing objects 124
response status lines 51
response-cache-control request header 169, 172
response-content-disposition request header 169, 172
response-content-encoding request header 169, 172
response-content-language request header 169, 172
response-content-type request header 169, 172
response-expires request header 169, 172
REST API 5
retention
about 17
default setting 18
hold 18
object versions 33
retrieving bucket ACLs
See also ACLs
about 79
example 81–83
HTTP status codes 81
request headers 80
request line 79
response headers 80
retrieving custom metadata
See also custom metadata
about 20, 131
example 139–140
retrieving object ACLs
See also ACLs
about 148
example 150–152
HTTP status codes 150
request headers 149
request line 148
response headers 149
retrieving objects
about 166
conditionally 168
examples 176–178
HTTP status codes 174–175
overriding response headers 169
partial 166–168
request headers 170–172
request line 169–170
response headers 173–174

S
s3curl
disabling SSL certificate verification 42
downloading 12
sample Java application
about 209–210
assumptions 209
code 211–216
required libraries 210–211
sample objects 12
saving network bandwidth 121
schema, XML 218–231
search
enabled for buckets 34–35
object naming considerations 17
permission 11
Search Console 7–8
search facilities 8
secret keys
AWS authentication 43
HCP authentication 203
Server response header 52
service points 38
ServiceUnavailable error code 57
SignatureDoesNotMatch error code 44, 57
signatures
about 43
calculating 44
Size element in bucket listing XML 105
size, maximum
custom metadata 20
SMTP protocol 5
space allocated to buckets 34
special characters in object names, percent-encoding 16–17
specifying ACLs
See also ACLs
about 25
ACL request body 27–32

canned 26
individual grant headers 26–27
SPNEGO 37
SSL security
about 5–6, 39
disabling SSL certificate verification 42
status codes
See HTTP status codes
Status element in versioning XML 84
status lines, response 51
StorageClass element in bucket listing XML 106
storing custom metadata 18–19
See also custom metadata
storing objects
encryption 121
ETags 120
examples 125–127
failed requests 185–186
HTTP status codes 124–125
invalid ACLs 120
request headers 122–123
request line 121
response headers 123–124
saving network bandwidth 121
structuring folders 184–185
Suspended versioning status 84
system metadata 2

T
tenants 3
Transfer-Encoding response header 52

U
unexpected content in .metapairs annotations 21
Unix hosts file 40
URI element in ACL XML 31
URLs for access to HCP (AWS authentication)
case sensitivity 41
IP addresses in URLs 39–40
targeting a bucket 38
targeting a tenant 38
targeting an object 39
URLs for access to HCP (HCP authentication)
targeting a bucket 202
targeting a tenant 202
targeting an object 202–203
user accounts
about 9
nonexistent in ACLs 73, 142
user authentication
about 9, 42
Active Directory 37
AWS 42–44
HCP 203
invalid credentials 44
methods 37
UTF-8-encoding object names 17

V
Version element in bucket listing XML 106
version IDs
about 33
copied objects 152
VersionId element in bucket listing XML 106
versionId query parameter
checking object existence 131, 133
copying objects 152, 155
retrieving objects 166, 169–170
VersionIdMarker element in bucket listing XML 106
version-id-marker query parameter
about 95–96
example 111–112
valid values 98
versioning
See also enabling/disabling versioning
about 32–34
checking status 87–91
enabling/disabling 33, 83–87
pruning 34
status 84
versioning query parameter
checking versioning status 87–88
enabling/disabling versioning 83, 85
versioning request body
See also enabling/disabling versioning
format 84
sample 87
versioning response body
See also checking versioning status
about 87
formatting 88
sample 91
VersioningConfiguration element in versioning XML 84
versions query parameter
about 91
examples 109–115

W
WebDAV protocol 5
Windows hosts file 40
WORM 2
write ACL permission 11

write permission 11
writing concurrently to same object 185

X

x-amz-acl request header
- about 26
- removing ACLs 32

x-amz-copy-source request header 155
x-amz-copy-source-if-match request header 153–154, 155
x-amz-copy-source-if-modified-since request header 153–154, 156
x-amz-copy-source-if-none-match request header 153–154, 157
x-amz-copy-source-if-unmodified-since request header 153–154, 157
x-amz-copy-source-version-id response header 159
x-amz-date request header 51
x-amz-delete-marker response header
- checking object existence 136
- deleting objects 179

x-amz-grant- request headers 26–27
x-amz-meta- request header 18–19
x-amz-meta- response header
- checking object existence 136
- retrieving objects 173

x-amz-metadata-directive request header 153, 158
x-amz-missing-meta response header
- checking object existence 137
- retrieving objects 174

x-amz-server-side-encryption request header
- copying objects 158
- storing objects 121, 123

x-amz-server-side-encryption response header
- copying objects 159
- storing objects 121, 124

x-amz-version-id response header
- checking object existence 137
- copying objects 160
- creating folders 129
- deleting objects 179
- retrieving objects 174
- storing objects 124

x-hcp-pretty-print request header
- checking versioning status 88
- copying objects 159
- listing bucket contents 99
- listing buckets 66
- retrieving bucket ACLs 80
- retrieving object ACLs 149

XML formats
- ACL request body 28–31
- bucket list response body 67–68
- bucket listing response body, current items 100
- bucket listing response body, object versions 101
- copy object response body 160
- error response body 53
- versioning request body 84

XML schema, HS3 API 218–231

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-92ARC037-00