HITACHI
Inspire the Next

Hitachi Content Platform

Using the HCP OpenStack Swift API

Hitachi Data Systems

MK-92ARC041-00

Contents

Preface

This book is the usage guide for Hitachi Content Platform (**HCP**) HSwift API. It provides the information you need to set up and use the HSwift API. There may be instances where you want to reference other material.

Intended audience

This book is intended for people want to learn how to use the HSwift API to store and manage containers and objects. 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 through a third-party, compatible tool (such as curl). 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. The book also assumes that you are familiar with the OpenStack Swift API and Keystone.

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

Product version

This book applies to release 7.1 of HCP.

Related documents

The following documents contain additional information about Hitachi Content Platform:

- *Administering HCP* - This book explains how to use an HCP system to monitor and manage a digital object repository. It discusses the capabilities of the system, as well as its hardware and software components. The book presents both the concepts and instructions you need to configure the system, including creating the tenants that administer access to the repository. It also covers the processes that maintain the integrity and security of the repository contents.

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

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

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

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

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

- *Using the HCP HS3 API* - This book contains the information you need to use the HCP HS3 API. This S3™-compatible, RESTful, HTTP-based API enables you to work with buckets and objects in HCP. The book introduces the HCP concepts you need to understand in order to use HS3 effectively and contains instructions and examples for each of the bucket and object operations you can perform with HS3.

- *Using the 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 or move data between local file systems, namespaces in HCP, and earlier HCAP archives. It also supports bulk delete operations and bulk operations to change object metadata. Additionally, it supports associating custom metadata and ACLs with individual objects. The book describes both the interactive window-based interface and the set of command-line tools included in HCP-DM.

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

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

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

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

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

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

Getting help

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

- United States: (800) 446-0744

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

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

Comments

Please send us your comments on this document:

HCPDocumentationFeedback@hds.com

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

Thank you!

1

Introduction to 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 that data. Objects exist in containers, which are logical partitions of the repository.

Hitachi Content Platform

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

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.

HCP can store multiple versions of an object, thus providing a history of how the data has changed over time. Each version is a separate object, with its own system metadata and, optionally, its own custom metadata and ACL. Except for the PUT command, which can replace the current version of an object with a newer one, HSwift API operations can only impact the latest version of an object.

Containers and accounts

An HCP repository is partitioned into namespaces which are called containers in the context of the HSwift API. A **container** is a logical grouping of objects such that the objects in one container are not visible in any other container.

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

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

Containers are managed by administrative entities called **tenants** in HCP. Tenant, in the context of HSwift, are referred to as **account**, and typically corresponds to an organization, such as a company or a division or department within a company.

This book uses the terms account and container when discussing the HSwift API and it uses the terms tenant and namespace when discussing HCP interfaces.

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 32. For information on when to use an IP address instead of a domain name, see "Using an IP address in a URL" on page 39.

Replication

Replication is the process of keeping selected tenants and containers in two or more HCP systems in sync with each other. Basically, this entails copying object creations, deletions, and metadata changes between systems. HCP also replicates tenant and container configurations, and user accounts.

A **replication topology** is a configuration of HCP systems that are related to each other through replication. Typically, the systems in a replication topology are in separate geographic locations and are connected by a high-speed wide area network.

You can read from container on all systems to which those namespaces are replicated. The replication configuration set at the system level determines on which systems you can write to containers.

Replication has several purposes, including:

- If one system in a replication topology becomes unavailable (for example, due to network issues), another system in the topology can provide continued data availability.
- If one system in a replication topology suffers irreparable damage, another system in the topology can serve as a source for disaster recovery.
- If multiple HCP systems are widely separated geographically, each system may be able to provide faster data access for some applications than the other systems can, depending on where the applications are running.
- If an object cannot be read from one system in a replication topology (for example, because a node is unavailable), HCP can try to read it from another system in the topology. Whether HCP tries to do this depends on the container configuration.
- If a system that participates in a replication topology is unavailable, HTTP requests to that system can be automatically serviced by another system in the topology. Whether HCP tries to do this depends on the container configuration.

Note: Not all HCP systems support replication. In a system that supports replication, whether any given tenant or container is replicated depends on the replication configuration.

About the HCP OpenStack Swift compatible API

The HCP HSwift API is a RESTful, HTTP-based API that is compatible with OpenStack. Using this API, you can:

- Create a container (PUT container)
- List the containers you own (GET account)
- List account metadata (HEAD account)
- Check a container's metadata (HEAD container)
- List the objects in a container (GET container)
- Delete a container as long as the container don’t have any objects in it (DELETE container)
- Store an object in a container (PUT object)
- Create a directory in a container (PUT object)
- Retrieve custom metadata for an object (HEAD object)
- Copy an object (COPY object)
- Retrieve an object (GET object)
- Delete an object (DELETE object)

To use the HSwift API to perform the operations listed above, you can write applications that use any standard HTTP client library. HSwift is also compatible with many third-party tools that OpenStack Swift implements. For information on configuring third-party tools for use with HSwift, see "Chapter D: Using third party tools" on page 143.

Other container access methods

HCP allows access to container (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 HSwift API, HCP supports access to namespace content through these protocols: HTTP (also called the HCP REST API), WebDAV, CIFS, HS3, 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 HSwift API to create a namespace (container), both the HSwift 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 access protocols for any namespace. File system protocols such as CIFS and NFS may be enabled only on a namespace that is not optimized for cloud protocols only (see the Managing a Tenant and Its Namespace manual for details). Cloud protocols such as HSwift, S3, and HCP REST may be enabled/disabled at any time regardless of optimization or protocol used to create the namespace.

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 HSwift, are immediately accessible through any other protocol that’s enabled for the namespace. Default namespaces cannot use the HSwift API.

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

Namespace browser

The HCP Namespace Browser lets you manage content in and view information about HCP 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, including:
  - The namespaces that you own or can access
  - Retention classes available for a given namespace
  - Permissions for namespace access
  - Namespace statistics such as the number of objects in a given namespace or the total capacity of the namespace

The Namespace Browser is not available for the default namespace. However, you can use a web browser to view the contents of that namespace.

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 by a user or application, or deleted through disposition, purged, or pruned. For namespaces that support versioning, operation-based queries can return both current and old versions of objects.

- Criteria for operation-based queries can include object status (for example, created or deleted), change time, index setting, and location.

The metadata query API returns object metadata only, not object data. The metadata is returned either in XML format, with each object represented by a separate element, or in JSON format, 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 that were stored before a certain date or that are larger than a specified size. You can then delete the objects listed in the search results or prevent those objects from being deleted. Similar to the metadata query API, the Search Console returns only object metadata, not object data.

By offering a structured environment for performing searches, the Search Console facilitates e-discovery, namespace analysis, and other activities that require the user to examine 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 either of these two 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 **Hitachi Data Discovery Suite (DDS) search facility** — This facility interacts with HDDS, which performs searches and returns results to the HCP Search Console. To use the HDDS search facility, you need to first install and configure HDDS, which is a separate product from HCP. HDDS is a separate product from HCP. The HDDS search facility works only with version 2.*x* of HDDS.

Note: Currently, HDDS does not support the use of IPv6 networks for communication with HCP.

The Search Console can use only one search facility at any given time. The search facility is selected at the HCP system level. If no facility is selected, the HCP system does not support use of 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, moving, and deleting data. With HCP-DM, you can:

- Copy or move objects, files, and directories between the local file system, HCP namespaces, default namespaces, and earlier HCAP archives
- Delete individual objects, files, and directories and perform bulk delete operations
- Watch directories for objects to automatically copy or move
- Set conditions for copy, move, and delete operations
- View the content of current and old versions of objects and the content of files
- 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
- 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*.

Accounts

In the context of this book, the term **HSwift account** is synonymous with the terms **HCP tenant** and **Keystone HCP tenant**. Each term pertains to its individual interface. In order to use an HSwift account with the HSwift API, the HSwift account must be associated with an HCP tenant with the same name as the HSwift account.

Once an HSwift account is created, you need to authenticate when you store and manage containers and objects. If you want to use Keystone authentication, you need to include a Authentication header and a Keystone authentication token with your command. This token verifies that you are authorized to work with containers and objects on the HSwift account. For more information about integrating your HSwift with HCP and Keystone, see Appendix B: "Keystone integration" on page 127.

If you choose to authenticate with an alternative authentication method, you can use local authentication to access your HSwift account. This method requires that you generate a temporary authentication token with your **HCP user account** (*not* your HSwift account) credentials. For more information on alternative authentication methods, see Appendix C: "Alternate authentication methods" on page 137.

Note: Your HSwift account and HCP user account are *not* the same. The HSwift account is the name of your HCP tenant in the context of the HSwift API. Your HCP user account is a set of credentials that gives an HCP user access to other interfaces. For more information on HCP user accounts, see *Administering HCP*.

Data access permissions

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

Data access permissions are container specific. That is, they are granted separately for individual containers.

Each data access permission allows you to perform certain operations. However, not all operations allowed by data access permissions apply to every HCP interface.

Although many of the operations allowed by data access permissions are not supported by the HSwift 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 container are:

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

For more information on:

- Container ACLs, see "Access control lists" on page 23
- Versioning, see "Versioning" on page 30

- The HCP Search Console, see "HCP search console" on page 8

Note: Some of the features and data access permissions listed here are not available for HSwift.

Examples in this book

This book contains instructions and examples for using HSwift to perform the operations listed in "About the HCP OpenStack Swift compatible API" on page 5 and Chapter 5: "Working with objects" on page 79. The examples use a command-line tool called **cURL**. **cURL** is freely available open-source software. You can download it from http://curl.haxx.se/download.html

After downloading **cURL**, you need to configure it to work with HCP. For information on this, see "Chapter D: Using third party tools" on page 143.

2

Container and object properties

Containers 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 are visible through the HSwift API. Others, such as versioning, aren't visible through HSwift but still affect that API.

This chapter describes container and object properties that have an impact on the HSwift API.

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

Container names

When you create a container, you give it a name. The name you specify for a container must be from one through 63 characters long and can contain only alphanumeric characters and hyphens (-) but cannot start or end with a hyphen.

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

The name you give a container must be unique for the tenant (account) for which you create the container. Different tenants can have containers with the same name.

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

Tenant administrators can change the name of a container any time after you create the container. When the name of a container changes, the URL for the container changes 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 although they are not recommended for use.
- Are case sensitive.

`.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 information on copying objects, see "Copying an object" on page 94.

Forward slashes in object names

When using the HSwift 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 directories, and forward slashes (/) in object names are simply part of those names.
- In a structured space, forward slashes in object names serve as directory separators, and objects can be grouped into directories.

To support the structured view, when you use HSwift to store an object that has forward slashes in its name, HCP also creates directories and subdirectories 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 directory named `quarterly_rpts`.

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

Names cannot be shared between directories and objects in the same container. If a container has a directory called `quarterly_rpts`, it cannot have an object with the name `quarterly_rpts`. If a container has an object by that name already, you cannot create a directory with that name.

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

You can also use HSwift to create directories and subdirectories by themselves. For example, you could create a directory named `mktg` with a subdirectory 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` directory. Directory names follow the same rules as object names.

When you use HCP interfaces other than HSwift to view or manage objects stored through HSwift, HCP always treats forward slashes in object names as directory 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` directory.

Grouping objects into directories not only lets you easily manage those objects, it can also enhance HCP performance. For information on how the directory structures you create can affect performance, see "Directory structures" on page 114.

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 directory. 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 directory 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 directory names that are longer than 255 bytes. An object stored through HSwift 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 HSwift 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	Description
Space	%20
Tab	%09
New line	%0A
Carriage return	%0D
+	%2B
%	%25
#	%23

(Continued)

Character	Description
?	%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 containers, 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 container 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 HSwift API to change the retention setting for an object.

Default retention setting

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

You cannot use the HSwift API to change the default retention setting for a container. However, tenant administrators can change this setting for the containers you create.

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 containers. 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 HSwift

With the HSwift API, you use X-Object-Meta- request headers to specify custom metadata. You can use these headers when you store or copy an object. With the X-Object-Meta- request headers, 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 2015, you would specify these headers:

```
X-Object-Meta-department: Sales&Mktg
X-Object-Meta-year: 2015
```

HCP stores the custom metadata you specify with HSwift 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[2015]]></meta-year>
</metapairs>
```

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

For each property/value pair, the name of the corresponding element in the .metapairs XML annotation is the concatenation of 'meta-' and the metadata key, 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 for 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 custom metadata property/value pair, the value is enclosed in a CDATA section. This makes the property become the value of the corresponding element.

If you specify an X-Object-Meta- header with no value, HCP doesn't store an element for the property named in the header. If all the X-Object-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-Object-Meta- property multiple times with different values. In the same .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-Object-Meta-author: P.D Grey
X-Object-Meta-author: Morgan White
X-Object-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-Object-Meta- headers result in three separate XML elements: X-Object-Meta-date_ written, X-Object-Meta-Date_Written, and X-Object-Meta-DATE_ WRITTEN.

Retrieving custom metadata with HSwift

When you check the existence of or retrieve an object that has a .metapairs annotation containing well-formed XML, the response headers include X-

Object-Meta-name 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 multiple X-Object-Meta- headers with individual property/value pairs. Here's an example that shows an XML element with comma-separated values and the X-Object-Meta- header that results from that element:

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

X-Object-Meta-author: P.D Grey
X-Object-Meta-author: Morgan White
X-Object-Meta-author: Paris Black
```

Custom metadata usage considerations

These following considerations apply to using custom metadata with the HSwift 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-Object-Meta- headers returned when you retrieve or check the existence of an object match the X-Object-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-Object-Meta- header won’t match the X-Object-Meta- header specified when the object was stored or copied. For example, if the specified header is X-Object-Meta-city/town, the returned header is X-Object-Meta-city_town.

Custom metadata size
When you use HSwift to store or copy an object, you can specify at most two kilobytes 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 an object under retention depends on a container setting. When you create a container, it's set to allow only the addition of custom metadata for objects under retention. You cannot use the HSwift API to change this setting. However, tenant administrators can change this setting for the containers you create.

.metapairs annotations with unexpected content
You could use HCP interfaces other than HSwift to store annotations named .metapairs, although it is not recommended. As a result, annotations named .metapairs are not guaranteed to be compatible with HSwift.

Here are some ways in which HCP responds to HSwift requests for object 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-Object-Missing-Meta- header with a value of 1 (one) and does not return any X-Object-Meta- headers.

If an element name doesn't start with *meta-*, HCP doesn't return an X-Object-Meta- header for the element.

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

If a *meta-* element has an attribute, HCP ignores the attribute and returns the applicable X-Object-Meta- header.

If the XML contains nested elements and the lowest-level element is a *meta-* element, HCP returns an X-Object-Meta- header for the element. It does not return X-Object-Meta- headers for any other elements in that nested structure.

Access control lists

HCP supports access control lists (ACLs) only for containers. An ACL grants specified Keystone authenticated users permissions to perform specific operations within a container. ACLs can also make containers public.

A container ACL grants permissions to perform operations on a container and on all objects in the container. For example, an ACL for a container could grant public read permission for that container. In this case any authenticated users can retrieve any objects in that container. Unauthenticated users may also be granted read access in this case, but only if the HCP tenant administrator has allowed unauthenticated users access to the namespace via namespace protocol settings.

Through HSwift ACLs, you can either grant container permissions to specific Keystone users or make containers public. User ACLs that have been set on a container may be viewed by an authorized user by using the **GET** or **HEAD** method on the container. User ACLs are not visible through the Tenant Management Console. If you make a container public, its permissions appear in the **Protocol** tab of the **Namespace** page, under the **Minimum Data Access Permissions** panel on the Tenant Management Console.

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

Permissions granted by HSwift ACLs apply to the container regardless of the protocol or API used to access the container.

Note: HSwift does not support the configuration of object ACLs although they can be set through other interfaces.

ACL permissions

Adding an ACL to a container gives the Keystone user certain data access permissions for that container.

The table below lists the permissions you can give in an ACL through the HSwift API 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 11.

ACL permission	Data access permissions
Read	Browse and read
Write	Write and delete

When you grant ACL permissions, you either assign these permissions to individual Keystone users or make the container public. Public containers are accessible by all authenticated users unless the Tenant Administrator has allowed unauthenticated users access to the namespace through namespace protocols settings.

ACLs can only be set by a Keystone HCP tenant admin or an HCP user with tenant management privileges.

Setting ACL permissions

You can grant ACLs to individual Keystone users or you can use them to make the permissions they grant public to anyone accessing the container.

In the following example, the X-Container-Write request header is used to give Keystone users P.D.Grey and Megan White the read permission:

"X-Container-Write: P.D.Grey, Megan White"

Note: The character limit to the header string is 8192 characters. You cannot add more Keystone users to the list once you have used all the characters available.

In order to make a container public you enter the asterisk (*) as the request header's value.

In the following example, the X-Container-Read request header is used to make the Read ACL public for the container:

"X-Container-Read: *"

Removing an ACL

You can remove an ACL from a container by using the X-Remove-Container-name header. This header can also be used to remove any other form of container metadata. When writing the header, substitute *name* with the ACL permission you want to remove (read or write). This is the format of the header:

"**X-Remove-Container-***name*"

Replication collisions

If users write to multiple systems in a replication topology, collisions can occur when different changes are made to the same objects on different systems. The way HCP handles collisions that occur due to replication depends on the type of collision. However, the general rule is that more recent changes have priority over conflicting less recent changes.

For an more information about replication, see the *Replicating Tenants and Namespaces* manual.

Object content collisions

An object content collision occurs when:

1. Two objects with the same name are created in a replicated container on two systems in a replication topology, and the object has different content on the two systems.

2. The object on one of the systems is replicated to the other system.

If versioning is enabled for the container (through the Tenant Management Console), no collision occurs. Instead, the less recently created of the two objects becomes an old version of the more recently created object.

When an object content collision occurs, the more recently created object keeps its name and location. The other object is either moved to the `.lost+found` directory in the same container or renamed, depending on the container configuration.

When HCP moves an object to the `.lost+found` directory, the full object path becomes `.lost+found/replication/`*system-generated-directory*`/`*old-object-path*.

When renaming an object due to a content collision, HCP changes the object name to *`object-name`*`.collision` or *`object-name`*.*version-id*`.collision`, where *`version-id`* is the version ID of the object. HCP uses the second format only if versioning has at one point been enabled for the namespace that contains the object but is not currently enabled.

If the new name is already in use, HCP changes the object name to *`object-name`*`.1.collision` or *`object-name`*.*version-id*`.1.collision`, as applicable. If that name is already in use, HCP successively increments the middle integer by one until a unique name is formed.

Objects that have been relocated or renamed due to content collisions are flagged as replication collisions in their system metadata. You can use the metadata query API to search for objects that are flagged as replication collisions.

If an object that's flagged as a replication collision changes (for example, the object ACL changes), its collision flag is removed. If you create a copy of a flagged object with a new name, the collision flag is not set on the copy.

Depending on the container configuration, objects flagged as replication collisions may be automatically deleted after a set number of days. The days are counted from the time the collision flag is set. If the collision flag is removed from an object, the object is no longer eligible for automatic deletion.

You cannot use the HSwift API to change the way HCP handles objects that are flagged as replication collisions. However, tenant administrators can change this configuration for the containers you create.

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

Custom metadata collisions

A custom metadata collision occurs when these events occur in the order shown:

1. One of these changes occurs:

   – Custom metadata is added to a given object on each of two systems in a replication topology, but the added custom metadata is different on the two systems.

     The addition of custom metadata to an object on only one of the systems does not result in a custom metadata collision. Instead, the new custom metadata is replicated from that system to the other system without conflict.

   – The custom metadata for a given object is replaced on each of two systems in a replication topology, but the replacement custom metadata is different on the two systems.

   – The custom metadata for a given object is replaced on one system in a replication topology, and the same custom metadata is removed on another system in the topology.

2. The change made on one of the systems is replicated to the other system.

Custom metadata is treated as a single unit. If a collision occurs when a custom metadata change for a given object is replicated from one system (system A) in a replication topology to another system (system B) in the topology:

- If the last change on system A is more recent than the last change on system B, HCP applies the change from system A to the custom metadata on system B.
- If the last change on system B is more recent than the last change on system A, HCP does not change the custom metadata on system B.

For example, suppose a given object starts out with the same custom metadata on system A and system B. The table below shows a sequence of events in which the custom metadata for the object is changed and the change is then replicated.

Sequence	Event
1	On system B, a client replaces the custom metadata for the object with new custom metadata.
2	On system A, a client replaces the custom metadata for the object with different custom metadata from the custom metadata used on system B.
3	The change on system A is replicated to system B. The resulting custom metadata for the object on system B is the new custom metadata from system A.

Access control list collisions

Object ACLs cannot be changed through the HSwift API but collisions can still occur. An ACL collision occurs when these events occur in the order shown:

1. Different changes are made to the ACL for a given object on each of two systems in a replication topology (on interfaces that support object ACLs).

2. The changed ACL on one of the systems is replicated to the other system.

An ACL is treated as a single unit. If a collision occurs when a changed ACL for a given object is replicated from one system (system A) in a replication topology to another system (system B) in the topology:

- If the last change to the ACL on system A is more recent than the last change to the ACL on system B, HCP changes the ACL on system B to match the changed ACL on system A.
- If the last change to the ACL on system B is more recent than the last change to the ACL on system A, HCP does not change the ACL on system B.

For example, suppose the ACL for a given object starts out with these grants on both system A and system B:

All users: read
User lgreen: write
User mwhite: write, delete

The table below shows a sequence of events in which the ACL for the object is changed and the change is then replicated.

Name	Description
1	On system B, a client changes the permissions in the ACL to: All users: read User lgreen: write, delete User mwhite: write, delete, read ACL
2	On system A, a client changes the permissions in the ACL to: All users: read User mwhite: write User pdgrey: write
3	The changed ACL on system A is replicated to system B. The resulting ACL for the object on system B contains these permissions: All users: read User mwhite: write User pdgrey: write

Versioning

Versioning is a feature that enables the creation and management of multiple versions of an object. Although this can be activated through the HCP API, HSwift does not support working with versioned objects.

If you store an object that has the same name as an existing object through HSwift, the old object is replaced by the newer one if versioning is enabled. If versioning is disabled you receive a 409 error code and the operation fails. Whether versioning is initially enabled when you create a container is determined by a tenant-level configuration setting.

Disabling versioning does not cause old versions of object to be deleted. Old versions of objects still exist in the HCP system although they cannot be accessed through the HSwift API. If you want to list older versions of objects it can be done through the HCP API.

Each version of an object is an object in its own right. It has system metadata and can have custom metadata and an ACL.

Allocated space

When you create a container, the container is allocated storage space based on the default quota set by the tenant administrator. Quota is the amount of space available for storing objects in the container.

The container quota may be changed by the tenant administrator through the Tenant Management Console, a Keystone Tenant admin through the HSwift API, or by an HCP user with tenant namespace management privileges.

URLs for access to HCP

With the HSwift API, the **endpoint** is a tenant. Every request you make to access HCP using HSwift 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 container within the tenant, or an object within a container 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 endpoints. Each tenant has its own default configuration settings for new containers. Containers, therefore, can have different characteristics depending on the tenants within which you create them.

When accessing a tenant that requires you to be an authenticated user, you need to provide Keystone authenticated credentials or HCP user credentials. for more information of Keystone credentials see, "Using a Keystone Authentication Token" on page 135. For more information of HCP user credentials, see "Using Local Authentication" on page 140.

Using SSL encryption

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

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

Account-level operations such as creating, updating or listing the objects of a container in an account can be done with both HTTP and HTTPS.

URL considerations

The following considerations apply to URLs in HSwift API requests.

Case sensitivity in URLs

In the URLs you specify in HSwift requests:

- HTTP and HTTPS are not case sensitive.
- Hostnames are not case sensitive.
- Container 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 HSwift 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 cURL, you disable SSL certificate verification by including the **`-k`** or **`--insecure`** command in the request command line.

Resource path line

The resource path line for an HSwift request specifies the operation target, whether it be an account, container, or object.

In the resource path for an HSwift request that uses Keystone authentication, the account name is interchangeable with the Tenant Id Prefix (default is AUTH_) followed by the Keystone Tenant ID. For example:

```
api.hcp.example.com/swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e/finance
```

In the resource path, the tenant ID prefix and Keystone tenant ID can be replaced with the HSwift account name. For example:

api.hcp.example.com/swift/v1/europe/finance

If the target of the request is a container, the resource path includes a forward slash followed by the container name.

If the target of the operation is an object, the request line includes a forward slash followed by the object name following the container name.

For example, here are the Keystone tenant authentication token and account name resource paths for a request to create an object named `Q4_2012.ppt`:

api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/finance/Q4_2012.ppt

api.hcp.example.com/swift/v1/europe/finance/Q4_2012.ppt

If the request uses any query parameters, those parameters are appended to the resource path.

For example, here is a request URL to alphabetically list all containers in the europe account between the letters E and S:

http://api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e?marker=e&end_marker=s

http://api.hcp.example.com/swift/v1/europe?marker=e&end_marker=s

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

Targeting an account

In an HSwift request, if the target is an account, you use a URL in this format:

http[**s**]**://api.***hcp-domain-name***/swift/v1/***AUTH_ID*

In the format above, the account name is replaced with the Keystone Prefix tenant ID (*AUTH_*) and the Keystone tenant ID. This is interchangeable with the HSwift account name.

Here’s an example in which the HSwift account is europe and the account name is used in place of the Keystone Prefix tenant ID and the Keystone tenant ID:

```
https://api.hcp.example.com/swift/v1/europe
```

Targeting a container

In an HSwift request that uses Keystone authentication, if the target is a container, you use a URL in this format:

http[**s**]**://api.***hcp-domain-name***/swift/v1/***AUTH_ID***/***container-name*

In the format above, the account name is replaced with the Keystone Prefix tenant ID (*AUTH_*) and the Keystone tenant ID. This is interchangeable with the HSwift account name.

Here’s an example in which the HSwift account name is europe and the container name is finance. The account name is used in place of the Keystone Prefix tenant ID and the Keystone tenant ID:

```
https://api.hcp.example.com/swift/v1/europe/finance
```

Targeting an object

In an HSwift request that uses Keystone authentication, if the target is an object, you use a URL in the following format:

http[**s**]**://api.***hcp-domain-name***/swift/v1/***AUTH_ID***/***container-name***/***object-name*

In the format above, the account name is replaced with the Keystone Prefix tenant ID (*AUTH_*) and the Keystone tenant ID. This is interchangeable with the HSwift account name.

Here’s an example in which the HSwift account name is europe, the container name is finance, and the object name is `Q4_2012.ppt`. The account name is used in place of the Keystone Prefix tenant ID and the Keystone tenant ID:

```
https://api.hcp.example.com/swift/v1/europe/finance/Q4_2012.ppt
```

Query parameters

Some HSwift requests accept query parameters. Query parameters are appended to a resource URL following a question mark (?). Multiple parameters are joined by ampersands (&).

The following considerations apply to query parameters:

- If you specify an invalid value for any query parameter, HCP returns a status code of 400 (Bad Request)
- If you specify a query parameter that's not valid for the request, HCP ignores it.

Query parameters are case sensitive.

Limiting a listing of current items

You can use the **delimiter, format, limit, marker, end_marker, path**, and **prefix** query parameters, alone or in combination with each other, to limit the list of the current objects in an account or container and control the format in which the response body is returned.

delimiter
The delimiter parameter is a single alphanumeric character used to request all objects in a container up through the first occurrence of the character specified by the parameter.

The delimiter parameter may be used to request directory listings where the forward (/) slash character is the path delimiter. Using the forward slash delimiter in combination with the prefix parameter, you can list the contents of any directory in the container. Omit the path parameter to list the root directory.

When you use the delimiter query parameter, the listing includes both objects and directories. If you omit the delimiter query parameter, **GET** requests only list objects.

For example, this is the content of an container:

Items:

```
AcctgBestPractices.doc
acctg/
hum_res/
mktg/
```

```
mktg/campaign_GoGetEm_expenses.xls
mktg/campaign_LiveIt_expenses.xls
quarterly_rpts/
quarterly_rpts/budget_proposals/Q2_2012.ppt
quarterly_rpts/budget_proposals/Q3_2012.ppt
quarterly_rpts/budget_proposals/quotas/Q4_2012.ppt
sales/
sales_quotas_2013.pdf
```

The container listing returned in response to a **GET** request with the **delimiter=/** query parameter contains these items:

Returned values:

```
AcctgBestPractices.doc
acctg/
hum_res/
mktg/
quarterly_rpts/
sales/
sales_quotas_2013.pdf
```

The lists of named items included in a listing are subject to any other character specified in the request.

Note: This query parameter can only be used to request a container listing, not an account listing.

format
You use the format query parameter to determine the way the response body is formatted. The default method is plain text, but this can be changed to either XML or JSON.

The Accept request header serves the same function as the format query parameter. If they are both used, the Accept header is ignored.

Here is an example of how to use the query parameter:

```
format=xml
```

limit
You use the **limit** query parameter to limit the number of items in the returned container listing to fewer than 10,000.

For example, the container listing returned in response to a **GET** request with the **limit=5** query parameter contains these items:

```
AcctgBestPractices.doc
mktg/campaign_GoGetEm_expenses.xls
mktg/campaign_LiveIt_expenses.xls
quarterly_rpts/budget_proposals/Q2_2012.ppt
quarterly_rpts/budget_proposals/Q3_2012.ppt
```

marker

A container listing is returned in alphabetical order. You use the **marker** query parameter to only list the items that sort alphabetically after the value of the marker parameter.

This is useful for paging through long lists of objects in a container. Using the name of the last object in the current page as the marker parameter retrieves the next page of objects.

For example, the container listing returned in response to a **GET** request with the **marker=mktg/campaign_LiveIt_expenses.xls** query parameter contains these items because they all appear after `mktg/campaign_LiveIt_expenses.xls`:

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

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

end_marker

You use the **end_marker** query parameter to only list the items that sort alphabetically before the value of the **end_marker** parameter.

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

```
AcctgBestPractices.doc
mktg/campaign_GoGetEm_expenses.xls
mktg/campaign_LiveIt_expenses.xls
```

path

You use the **path** query parameter to list all objects within the directory of a specified path. If there are subdirectories in the selected path the contents of the subdirectories are not listed. For instance, the following objects are returned in response to **GET** container request:

```
AcctgBestPractices.doc
acctg/
hum_res/
mktg/
mktg/campaign_GoGetEm_expenses.xls
mktg/campaign_LiveIt_expenses.xls
quarterly_rpts/
quarterly_rpts/budget_proposals/Q2_2012.ppt
quarterly_rpts/budget_proposals/Q3_2012.ppt
quarterly_rpts/budget_proposals/quotas/Q4_2012.ppt
sales/
sales_quotas_2013.pdf
```

path=quarterly_rpts returns:

```
quarterly_rpts/
quarterly_rpts/budget_proposals/
```

prefix

You use the **prefix** query parameter to request a container listing that contains only items with names that begin with the specified character string (the prefix).

The **prefix** query parameter is used in conjunction with the **delimiter** parameter to list the contents of directories.

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

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

Using an IP address in a URL

Normally, you let HCP choose the node on which to process an HSwift 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.

Here's an example of a curl command that uses a URL with an IP address. In this instance, the tenant name is replaced by its Keystone Prefix tenant ID and the Keystone tenant ID:

```
curl -X PUT -T http://192.168.210.16/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/finance /Q4_2012.ppt -H "X-Auth-Token:
dc5efec8f546455eac974e7bbfd0dd87" -H "Content-Length: 63"
```

The formats for IP address URLs are:

```
https://ip-address/swift/v1/tenant-name/namespace-name/object-name

https://ip-address/swift/v1/AUTH_tenant-id/namespace-name/object-name
```

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 114.

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

3

Requests and responses

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

This section goes over the common request and response headers that HSwift supports. Due to storage differences, not all requests and responses from the Swift API can be implemented when storing to HCP. This chapter highlights the ones that HCP does accept.

Requests

HSwift is an HTTP-based API. As is standard for HTTP, the structure for an HSwift 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 HSwift request specifies the operation to be performed, whether the target of the operation is an account, container, or object, and the version of HTTP on which the HSwift API is based.

In an HSwift request that uses Keystone authentication, the account name is interchangeable with the Keystone tenant authentication token.

For example, here are the Keystone tenant authentication token and account name request lines for a request to create a container named finance:

```
PUT /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e/finance HTTP/1.1
```

```
PUT /swift/v1/europe/finance HTTP/1.1
```

If the target of the request is a container, the request line includes a forward slash followed by the container name.

If the target of the operation is an object, the request line includes a forward slash followed by the object name following the container name.

For example, here are the Keystone tenant authentication token and account name request lines for a request to create an object named `Q4_2012`:

```
PUT /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e/finance/Q4_2012.ppt
HTTP/1.1
```

```
PUT /swift/v1/europe/finance/Q4_2012.ppt HTTP/1.1
```

If the request uses headers such as the X-Container-Write header it is shown under the request.

For example, here are the Keystone tenant authentication token and account name request lines and X-Write-Container headers for a request to create a container named finance and add an ACL to the container:

```
PUT /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e/finance/Q4_2012.ppt/
HTTP/1.1
"X-Container-Write: *"

PUT /swift/v1/europe/finance/Q4_2012.ppt/ HTTP/1.1
"X-Container-Write: *"
```

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 Keystone tenant authentication token and account name request lines for a request to list all containers in between the letters E and S:

```
GET /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e?marker=e&end_marker=s/
HTTP/1.1

GET /swift/v1/europe?marker=e&end_marker=s/ HTTP/1.1
```

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

Common request headers

Some request headers are common to all HSwift requests. The table below describes those headers. Request headers that are specific to certain requests are described in the section for those requests in Chapter 4: "Working with containers" on page 49 and "Chapter 5: Working with objects" on page 79.

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

Name	Description
X-Auth-Token	See Chapter 1: "Issuing Keystone authentication tokens to users" on page 1. See "Using Local Authentication" on page 140. Used to supply the Keystone authentication token or local authentication token.

Responses

The response to an HSwift 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 HSwift request specifies the version of HTTP on which the HSwift API is based and an HTTP status code that indicates the outcome of the request. HTTP status codes in the 2xx range indicate that the requested operation was successful. HTTP status codes in the 3xx, 4xx, and 5xx 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 Success
```

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 46. For the status codes that are specific to certain requests, see the individual request descriptions in "Chapter 4: Working with containers" on page 49 and "Chapter 5: Working with objects" on page 79.

Common response headers

Some response headers are common to all HSwift 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 4: Working with containers" on page 49 and "Chapter 5: Working with objects" on page 79.

Name	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 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 Mat 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 V7.1.0.16)
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.
X-Trans-Id	HCP returns a universally unique identifier (UUID). This UUID does not map to any entries in the HCP database.

Error response body

When an HSwift request results in an error, HCP returns information about the error in an error request body. An error request body contains HTML in this format:

```
<html>
     <h1>HTTP-status-code</h1>
     <p>error-message</p>
</html>
```

Error codes

Every error response body contains an HTTP status code and a description that provides more information about the error. The error codes themselves can be found with the response header. 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 HSwift requests. For information on the HTTP status codes returned in response to specific requests, see the individual request descriptions "Chapter 4: Working with containers" on page 49 and "Chapter 5: Working with objects" on page 79.

Error code	Description	HTTP Status Code
304	One of: • The request included an If-None-Match header, and the ETag for the specified object matched a value specified by the header. • The request included an If-Modified-Since header, and the specified object was not modified after the date and time specified by the header.	Not modified
401	Token Expired. Token Invalid. No Token Provided.	Unauthorized
403	Authenticated user does not have permission to perform requested operation.	Forbidden
404	The resource identified in the URL could not be found on the server.	Not Found
405	The HTTP method is not supported	Method Not Allowed

(Continued)

Error code	Description	HTTP Status Code
412	One of: • The request included an If-Match header, and the ETag for the specified object doesn't not match any of the values specified by the header. • The request included an If-Unmodified-Since header, and the specified object was modified after the date and time specified by the header.	Precondition Failed

4

Working with containers

For each container 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 HSwift requests and HCP responses to those requests, see Chapter 3: "Requests and responses" on page 41

For information about the examples in this chapter, see "Examples in this book" on page 13.

Creating a container

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

When you create a container, you specify a name for it. For information on naming containers, see "Container names" on page 15. If you use the **PUT** method to add an another container with a name that is identical to a container that already exists, you overwrite the metadata of the current container.

You can specify an ACL for a container in the same request as you use to create the container. To do this, you need to use ACL headers. For more information on ACLs, see "Access control lists" on page 23.

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

When you create a container, you get browse, read, and write data access permissions for the container.

For information on data access permissions see "Data access permissions" on page 11.

PUT container request line

Depending on whether the HSwift request uses a Keystone tenant ID or the account name, the request line for a request to create a container has either of these formats:

- With the Keystone tenant ID:

  ```
  PUT /swift/v1/tenant-ID/container-name HTTP/1.1
  ```

- With the account name:

  ```
  PUT /swift/v1/account-name/container-name HTTP/1.1
  ```

PUT container request headers

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

Name	Type	Description
X-Auth-Token	String	See "Common request headers" on page 43
X-Container-Meta-Quota-Bytes	String	Sets a limit on the size, in bytes, of objects that can be stored in the container. Once you set this quota it cannot be removed, only changed.
X-Container-Read	String	Adds Read, Browse or both data access permissions to the container. This header can be used to grant permissions to individual Keystone users or make the container public. For valid values for this header, see "Access control lists" on page 23
X-Container-Write	String	Adds Write, Delete or both data access permissions to the container. This header can be used to permissions to individual Keystone users or make the container public. For valid values for this header, see "Setting ACL permissions" on page 25"Access control lists" on page 23

PUT container response headers

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

Name	Type	Description
Content-length	String (Required)	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 is **0** (zero).
Content-Type	String (Required if the Content-Length is greater than 0)	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	Datetime (Required)	See, "Common response headers" on page 44
X-Trans-Id	Uuid (Required)	See, "Common response headers" on page 44

PUT container HTTP response codes

The table below describes HTTP status codes that can be returned in response to a request to create a container. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 46.

Code	Meaning	Description
201	Created	The container is created.
202	Accepted	The container already exists.

PUT container examples

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

Creating a container example

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

Request with cURL command line

```
curl -X PUT http://api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/finance -i -H "X-Auth-Token:
dc5efec8f546455eac974e7bbfd0dd87"
```

Request headers

```
PUT /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e/finance HTTP/1.1
X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87
```

Response headers

```
HTTP/1.1 201 Created
Date: Fri, 31 Oct 2014 12:45:01 GMT
Server: HCP V7.1.0.353
Content-Length: 0
X-Trans-Id: tx78cf0d46829f49529ed1a-0053fccad0
```

Creating a container with an ACL example (data permission only)

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

Request with cURL command line

```
curl -X PUT http://api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/human-resources -i -H "X-Auth-Token:
dc5efec8f546455eac974e7bbfd0dd87" -H "X-Container-Read: *" -H "X-Container-Write:
mwhite, pdgrey"
```

Request headers

```
PUT /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e/finance HTTP/1.1
X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87
```

Response headers

```
HTTP/1.1 202 Created
Date: Fri, 31 Oct 2014 12:48:13 GMT
Server: HCP V7.1.0.353
Content-Length: 0
X-Trans-Id: tx78cf0d46829f49529ed1a-0053fccad0
```

Listing account metadata

You use the HTTP **HEAD** method to list account metadata. You need an account to request this operation.

The target of a request to list the account metadata is a tenant (that is, the service point). When you list account metadata, you are returned information about the total number of containers on that account, the number of objects in the account, and the total amount of bytes used.

The total number of containers is calculated in the context of that tenant and even incorporates containers that don't have the HSwift API enabled. The object count is calculated in the same manner.

HEAD account request line

Depending on whether the HSwift request uses a Keystone tenant ID or the account name, the request line for a request to list account containers has either of these formats:

- With the Keystone tenant ID:

  ```
  HEAD /swift/v1/tenant-ID HTTP/1.1
  ```

- With the account name:

  ```
  HEAD /swift/v1/account-name HTTP/1.1
  ```

HEAD account request headers

The table below describes the headers you can use in a request to list account container data.

Name	Type	Description
X-Auth-Token	String	See "Common request headers" on page 43

HEAD account response headers

The table below describes the response headers returned in response to a successful request to list account container data.

Name	Type	Description
Content-length	String (Required)	See, "Common response headers" on page 44
Content-Type	String (Required if Content-Length is less than 0)	See, "Common response headers" on page 44
Date	Datetime (Required)	See, "Common response headers" on page 44
X-Account-Bytes-Used	Int (Required)	The number of containers on the account. This number is limited to the total number of namespaces that an HCP tenant may have which is 10,000.
X-Account-Object-Count	Int (Required)	This number is the total amount of objects that the account has across all of its containers.
X-Trans-Id	Uuid (Required)	See, "Common response headers" on page 44

HEAD account HTTP response codes

The table below describes HTTP status codes that can be returned in response to a request to list account container data. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 46.

Code	Meaning	Description
204	Success	If the request succeeds, the operation returns this code.
401	Unauthorized	If you do not have the permissions to view the account's container data.

Listing an account's metadata example

Here's a sample **HEAD** request that lists the metadata of an account named europe.

Request with curl command line

```
curl -i -X HEAD http://api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e -H "X-Auth-Token:
dc5efec8f546455eac974e7bbfd0dd87"
```

Request headers

```
HEAD /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e HTTP/1.1
X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87
```

Response headers

```
HTTP/1.1 204 No Content
Date: Fri, 31 Oct 2014 12:53:19 GMT
X-Trans-Id: 47b950a0-2e0d-4c25-86a3-fbd709b55a08
X-Account-Object-Count: 7
X-Account-Container-Count: 2
X-Account-Bytes-Used: 349
Content-Length: 0
```

Listing containers

You use the HTTP **GET** method to list the containers in an account. You need an account to request this operation.

The target of a request to list the container is a tenant (that is, the service point). The list of containers in the response contains only containers created in the context of that tenant and lists all containers with HSwift API enabled. The containers are listed in alphanumeric order.

The list of containers is returned in a plain text response body, although it can be configured to be returned in XML or JSON response bodies. For information on the format of this response body, see "GET account response body" on page 58

Limiting the container listing

You use query parameters to limit the items included in a container listing. For more information on query parameters, see "Query parameters" on page 36.

GET account request line

Depending on whether the HSwift request uses a Keystone tenant ID or the account name, the request line for a request to list the containers you own has this format:

- With the Keystone tenant ID:

  ```
  GET /swift/v1/tenant-ID HTTP/1.1
  ```

- With the account name:

  ```
  GET /swift/v1/account-name HTTP/1.1
  ```

GET account request headers

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

Name	Type	Description
Accept	String	Set this header to application/json, application/xml, or text/xml. The response body will be serialized in the specified format.
X-Auth-Token	String	See, "Common request headers" on page 43

GET account response headers

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

Name	Type	Description
Content-Length	String (Required)	See, "Common response headers" on page 44
Content-Type	String (Required)	See, "Common response headers" on page 44
Date	Datetime (Required)	See, "Common response headers" on page 44
X-Account-Bytes-Used	Int (Required)	This number is the total amount of bytes stored in object storage by this account. The number includes to total amount of bytes across all objects in all containers on the account.
X-Account-Container-Count	Int (Required)	The number of containers on the account. This number is limited to the total number of namespaces that an HCP tenant may have which is 10,000.
X-Account-Object-Count	Int (Required)	This number is the total amount of objects that the account has across all of its containers.
X-Trans-Id	Uuid (Required)	See, "Common response headers" on page 44

GET account response body

By default, HCP returns the list of containers you own in a plain text format. For the purpose of this example, the response body have been converted to XML using the Accept header. This is the response body, in XML format:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<account name="europe">
  <container>
    <name>finance</name>
    <bytes>8192</bytes>
    <count>1</count>
  </container>
  <container>
    <name>human-resources</name>
    <bytes>0</bytes>
```

```
        <count>0</count>
      </container>
    </account>
```

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

Element	Description
Account name	Root element
Bytes	Child of the **Container** element. The **Bytes** element specifies the size, in bytes, of the content of the item. The response body includes **Size** elements for the listed directories. However, because directories have no content, the value of this element for a directory is always **0** (zero).
Count	Child of the **Container** element. The **Count** element specifies the number of directories inside the **Container** element.
Container	Child of the **Account name** element and container for the **Name**, **Bytes**, and **Count** elements. The response body contains one **Container** element for each container you own.
Name	Child of the **Container** element. The **Name** element specifies the name of a **Container**.

GET account HTTP response codes

The table below describes HTTP status codes that can be returned in response to a request to list the containers you own. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 46.

Code	Meaning	Description
200	Success	The response body has listed the containers on the account.
204	Success	This code applies only to the Plain Text Response. The response body shows no containers. Either the account has no containers or you are paging through a long list of names by using the marker, limit, or end_marker query parameters, and you have reached the end of the list.

Listing containers example

Here's a sample **GET** request that returns a list of the containers in the context of the europe tenant.

Request with curl command line

```
curl -X GET http://api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e -H "Accept: application/xml" -H "X-Auth-Token:
dc5efec8f546455eac974e7bbfd0dd87"
```

Request headers

```
GET /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e HTTP/1.1
X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87
Accept: application/XML
```

Response headers with containers

```
HTTP/1.1 200 OK
Date: Fri, 17 Jan 2014 16:03:23 GMT
Server: HCP V7.1.0.353
X-Trans-Id: 28b610b7-0fbc-4d83-821b-f7d2a3f20e27
Content-Type: application/xml;charset=UTF-8
X-Account-Object-Count: 1
X-Account-Container-Count: 2
X-Account-Meta-Subject: Literature
X-Account-Bytes-Used: 14
Content-Length: 0
```

Response body with containers

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<account name="europe">
   <container>
      <name>finance</name>
      <bytes>8192</bytes>
      <count>1</count>
   </container>
   <container>
      <name>human-resources</name>
      <bytes>0</bytes>
      <count>0</count>
   </container>
</account>
```

Checking a container's metadata

You use the HTTP **HEAD** method to check a container's metadata. To successfully check a container's metadata, you need read permission for the container.

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

HEAD container request line

Depending on whether the HSwift request uses a Keystone tenant ID or the account name, the request line for a request to check a container's metadata has either of these formats:

- With the Keystone tenant ID:

  ```
  HEAD /swift/v1/tenant-ID/container-name HTTP/1.1
  ```

- With the account name:

  ```
  HEAD /swift/v1/account-name/container-name HTTP/1.1
  ```

HEAD container request headers

The table below describes the headers you can use in a request to check a container's metadata.

Name	Type	Description
X-Auth-Token	String	See, "Common request headers" on page 43

HEAD container response headers

The table below describes the response headers returned in response to a successful request to check a container's metadata.

Name	Type	Description
Accept-Ranges	String (Required)	Always **bytes**. This header shows the type of ranges that an object accepts from a Range request header.
Content-Length	String (Required)	See, "Common response headers" on page 44
Content-Type	String (Required)	See, "Common response headers" on page 44
Date	String (Required)	See, "Common response headers" on page 44
X-Container-Bytes-Used	Int (Required)	The total amount of bytes of data stored in the container. This number is capped by the *X-Container-Meta-Quota-Bytes* header.
X-Container-Meta-Quota-Bytes	String	If a quota has been set on the amount of bytes that can be stored by this container, this header returns the size, in bytes, of objects that can be stored in the container.
X-Container-Object-Count	Int (Required)	The total number of objects stored in the container. This number includes all objects in all directories that are in the container. Directories are not included in this number.

(Continued)

Name	Type	Description
X-Container-Read	String	Lists the Read and Browse permissions existing on the container. This header can either list individual Keystone users with the Read permissions or mark the container as public. For valid values for this header, see "Setting ACL permissions" on page 25
X-Container-Write	String	Lists the Write ACL permissions existing on the container. This header can either list individual Keystone users with Write permissions or mark the container as public. For valid values for this header, see "Setting ACL permissions" on page 25.
X-Trans-Id	Uuid (Required)	See, "Common response headers" on page 44

HEAD container HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to check a container's metadata. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 46.

Code	Meaning	Description
204	No Content	No Response Body.

Checking a container's metadata example

Here's a sample **HEAD** request that checks the metadata of a container named finance.

Request with curl command line

```
curl -i -X HEAD http://api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/finance -H "X-Auth-Token:
dc5efec8f546455eac974e7bbfd0dd87"
```

Request headers

```
HEAD /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e/finance HTTP/1.1
X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87
```

Response headers

```
HTTP/1.1 204 No Content
Date: Fri, 31 Oct 2014 13:07:45 GMT
Server: HCP V7.1.0.353
X-Trans-Id: 3c74e66d-9fb7-407d-9762-391d3e77014f
X-Container-Object-Count: 1
X-Container-Bytes-Used: 0
X-Container-Meta-Quota-Bytes: 10737418240
X-Container-Read:
X-Container-Write: *
Accept-Ranges: bytes
Content-Length: 0
```

Adding an ACL to a container (data access permission only)

You use the HTTP **POST** method with either the read or write ACL header to add an ACL to an existing container. Adding an ACL to a container replaces the existing read or write ACL in its entirety. You cannot modify an existing ACL in place.

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

POST container request line

Depending on whether the HSwift request uses a Keystone tenant ID or the account name, the request line for a request to add an ACL (write or read) to a container has either of these formats:

- With the Keystone tenant ID:

```
POST /swift/v1/tenant-ID/container-name HTTP/1.1
"X-Container-Write: acl-permitted-account-name"
```

or

```
POST /swift/v1/tenant-ID/container-name HTTP/1.1
"X-Container-Read: acl-permitted-account-name"
```

- With the account name:

```
POST /swift/v1/account-name/container-name HTTP/1.1
"X-Container-Write: acl-permitted-account-name"
```

or

```
POST /swift/v1/account-name/container-name HTTP/1.1
"X-Container-Read: acl-permitted-account-name"
```

POST container request headers

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

Name	Type	Description
X-Auth-Token	String (Required)	See, "Common request headers" on page 43
X-Container-Meta-Quota-Bytes	String	Sets a limit on the size, in bytes, of objects that can be stored in the container.
X-Container-Read	String	Adds a Read, Browse or both access permissions to the container. This header can be used to add permissions to individual Keystone users or make the container public. For valid values for this header, see "Setting ACL permissions" on page 25

(Continued)

Name	Type	Description
X-Container-Write	String	Adds Write, Delete or both access permissions to the container. This header can be used to add permissions to individual Keystone users or make the container public. For valid values for this header, see "Setting ACL permissions" on page 25.
X-Remove-Container-name	String	This header removes other headers supported by the POST command. Instead of replacing the data with new information, this header acts as a delete for other headers. Here is an example of how to use the header. If you want to get rid of all your X-Container-Write ACL permissions on a container you format the X-Remove-Container-name header like this: *X-Remove-Container-Write* When you execute this example operation all write ACLs are removed from the container.

POST container response headers

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

Name	Type	Description
Content-Length	String (Required)	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 is **0** (zero).

(Continued)

Name	Type	Description
Content-Type	String (Required if Content-Length is greater than 0)	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	Datetime (Required)	See, "Common response headers" on page 44
X-Trans-Id	Uuid (Required)	See, "Common response headers" on page 44

POST container HTTP status codes

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

Code	Meaning	Description
204	No Content	The POST was successful

Adding an ACL to a container examples

The following sections show sample **POST** requests for adding ACLs to containers.

Adding an ACL to a container by specifying individual users example

Here's a sample **POST** request that adds an ACL to the finance container by using the X-Container-Write and X-Container-Read headers. The ACL grants write permission to the Keystone authenticated user pdgrey, and read permissions to Keystone authenticated users pdgrey and mwhite.

Request with curl command line

```
curl -i -X POST http://api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/finance -H "X-Auth-Token:
dc5efec8f546455eac974e7bbfd0dd87" -H "X-Container-Write: pdgrey" -H "X-Container-
Read: pdgrey, mwhite"
```

Request headers

```
POST /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e/finance HTTP/1.1
X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87
X-Container-Write: pdgrey
X-Container-Read: pdgrey, mwhite
```

Response headers

```
HTTP/1.1 204 No Content
Date: Fri, 31 Oct 2014 13:07:45 GMT
Server: HCP V7.1.0.353
X-Trans-Id: 3c74e66d-9fb7-407d-9762-391d3e77014f
Content-Length: 0
Content-Type: application/octet-stream
```

Adding a public ACL to a container example

Here's a sample **POST** request that adds an ACL that give public read, browse, write, and delete data permissions to the finance container by using the X-Container-Write and X-Container-Read headers.

Request with curl command line

```
curl -i -X POST http://api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/finance -H "X-Auth-Token:
dc5efec8f546455eac974e7bbfd0dd87" -H "X-Container-Write: *" -H "X-Container-Read:
*"
```

Request headers

```
POST /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e/finance HTTP/1.1
X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87
X-Container-Write: *
X-Container-Read: *
```

Response headers

```
HTTP/1.1 204 No Content
Date: Fri, 31 Oct 2014 13:07:45 GMT
Server: HCP V7.1.0.353
X-Trans-Id: 3c74e66d-9fb7-407d-9762-391d3e77014f
Content-Length: 0
Content-Type: application/octet-stream
```

Listing container contents

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

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

```
quarterly_rpts/
quarterly_rpts/Q4_2012
```

By default, a container listing includes only the current (or only) versions of objects. You cannot request to view older versions of objects using HSwift.

You can use query parameters to list only a subset of the items in a container. For information on using these parameters, see "Limiting the container listing" on page 57.

A container listing is returned in a plain text response body although the format can be changed to either an XML or JSON response body. In the response body, items in the container listing occur in alphanumeric order.

By default, a container listing can include at most 10,000 items. However, you can use the **limit** 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 object listing

You use query parameters to limit the items included in a object listing. For more information on query parameters, see, "Query parameters" on page 36

GET container request line

Depending on whether the HSwift request uses a Keystone tenant ID or the account name, the request line for a request to list the contents of a container has either of these formats:

- With the Keystone tenant ID:

  ```
  GET /swift/v1/tenant-ID/container-name[?query-parameters]
  HTTP/1.1
  ```

- With the account name:

  ```
  GET /swift/v1/account-name/container-name[?query-parameters]
  HTTP/1.1
  ```

In these formats, query-parameters can be none, one, or a combination of the query parameters described in "Query parameters" on page 36.

GET container request headers

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

Name	Type	Description
X-Auth-Token	String	See, "Common request headers" on page 43

(Continued)

Name	Type	Description
Accept	String	Set this header to application/json, application/xml, or text/xml. The response body will be serialized in the specified format. This can also be accomplished with the format query parameter, see "Limiting the container listing" on page 57. If you use both at once, the response body defers to the format provided by the query parameter.

GET container response headers

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

Name	Type	Description
Accept-Ranges	String (Required)	Always **bytes**. This header shows the type of ranges that an object accepts from a Range request header.
Content-Length	String (Required)	See, "Common response headers" on page 44
Content-Type	String (Required)	See, "Common response headers" on page 44
Date	Datetime (Required)	See, "Common response headers" on page 44
X-Container-Bytes-Used	Int (Required)	The total amount of bytes of data stored in the container. This number is capped by the *X-Container-Meta-Quota-Bytes* header.
X-Container-Meta-Quota-Bytes	String	If a quota has been set on the amount of bytes that can be stored by this container, this header returns the size, in bytes, of objects that can be stored in the container.

(Continued)

Name	Type	Description
X-Container-Object-Count	Int (Required)	The total number of objects stored in the container. This number includes all objects in all directories that are in the container. Directories are not included in this number.
X-Trans-Id	Uuid (Required)	See, "Common response headers" on page 44

GET container response body

The response body returned in response to a request to list the contents of a container in an XML response body, in this format:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<container name="sales-practice">
  <object>
    <name>mktg/campaign_GoGetEm_expenses.xls</name>
    <hash>D357101A82D4F262E25860284C13EAC58F9A27A6AD70491BC6AAB4FB51
13ABBC</hash>
    <bytes>23040</bytes>
    <content_type>application/octet-stream</content_type>
    <last_modified>2014-10-31T12:39:16.000055</last_modified>
    <subdir xsi:nil="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"/>
  </object>
</container>
```

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

Element	Description
Bytes	Child of the **Object** element. The **Bytes** element specifies the size, in bytes, of the content of the item. The response body includes **Bytes** elements for the listed folders. However, because folders have no content, the value of this element for a folder is always **0** (zero).
Container	Root element
Content-Type	Child of the **Object** element. The **Content-Type** element specifies 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.
Hash	Child of the **Object** element. The **Hash** element specified the calculated the Base64-encoded MD5 hash of the item.
Last_Modified	Child of the **Object** element. The **Last_Modified** 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: 2014-10-20T16:34:27.000135 Modifying an object means modifying its metadata. You cannot modify the content of an object.

(Continued)

Element	Description
Object	Child of the **Container** element and container for the **Bytes**, **Content-Type**, **Hash**, **Last Modified** and **subdir xsi:nil** elements. The response body contains one **Container** element for each container you own.
subdir xsi:nil="true | false"	Child of the **Container** element. The response body specifies whether the **Object** element is a sub-directory (true) or object (false). The response body lists elements for sub-directories as well as objects. However, because sub-directories have no content, the value of the **Bytes** element is always **0** (zero) and the value of the **Content_Type** is always **application/octet-stream**.

GET container HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to list container contents. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 46.

Code	Meaning	Description
200	OK	The response body lists the objects.
204	No Content	The response body shows no objects. Either the container has no objects or you are paging through a long list of names by using the marker, limit, or end_marker query parameters, and you have reached the end of the list.
404	Not Found	If the container does not exist.

Listing the items in a directory example

Here's a sample **GET** request for a container listing that lists the objects that are in the `mktg` directory and, separately, the subdirectories that are in the `mktg` directory. The requests 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 curl command line

```
curl -i -X GET http://api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/finance?prefix=mktg&marker=l&delimiter=/" -H
"X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87" -H "Accept: application/xml"
```

Request headers

```
GET /swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/finance?prefix=mktg&marker=mktg&delimiter
=/" HTTP/1.1
X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87
```

Response headers

```
HTTP/1.1 200 OK
Date: Fri, 31 Oct 2014 16:41:58 GMT
Server: HCP V7.1.0.353
X-Trans-Id: 2a6d516f-b78b-418b-aea3-12fc86316ad4
X-Container-Object-Count: 2
X-Container-Bytes-Used: 46080
X-Container-Meta-Quota-Bytes: 53687091200
X-Container-Read:
X-Container-Write:
Content-Type: application/xml;charset=UTF-8
```

Deleting a container

You use the HTTP **DELETE** method to delete a container. To delete a container, you need to have a Keystone or local authentication token for the HSwift account in which the container exists.

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

DELETE container request line

Depending on whether the HSwift request uses a Keystone tenant ID or the account name, the request line for a request to delete a container has either of these formats:

- With the Keystone tenant ID:

  ```
  DELETE /swift/v1/tenant-ID/container-name HTTP/1.1
  ```

- With the account name:

  ```
  DELETE /swift/v1/account-name/container-name HTTP/1.1
  ```

DELETE container request headers

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

Name	Type	Description
X-Auth-Token	String	See, "Common request headers" on page 43

DELETE container response headers

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

Name	Type	Description
Content-Length	String (Required)	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 is **0** (zero).

(Continued)

Name	Type	Description
Content-Type	String (Required if Content-Length is greater than 0)	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	Datetime (Required)	See, "Common response headers" on page 44
X-Trans-Id	Uuid (Required)	See, "Common response headers" on page 44

DELETE container HTTP status codes

The table below describes HTTP status codes tat can be returned in response to a request to delete a container. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 46.

Code	Meaning	Description
204	No Content	The container was deleted.
404	Not Found	The container you are trying to delete does not exist in this account.
409	Conflict	The container you are trying to delete is not empty.

Deleting a container example

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

Request with curl command line

```
curl -i -X DELETE http://api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/sales-mkgt -H "X-Auth-Token:
dc5efec8f546455eac974e7bbfd0dd87"
```

Request headers

```
DELETE /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e/sales-mkgt HTTP/1.1
X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87
```

Response headers

```
HTTP/1.1 204 No Content
Date: Fri, 31 Oct 2014 14:06:11 GMT
Server: HCP V7.1.0.353
Content-Length: 0
X-Trans-Id: tx66248155dbb74b6a85f6d-0053fccdcd
Content-Type: application/octet-stream
```

5

Working with objects

With the HSwift API, you can perform operation 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 request 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 the HSwift requests and HCP responses to those requests, Chapter 3: "Requests and responses" on page 41

For information about the examples in the chapter, see "Examples in this book" on page 13.

Storing an object

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

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 16.

If versioning is enabled and you try to store an object with the same name as an existing object, HSwift replaces the old object with a newer version. 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 versioning, see "Versioning" on page 30.

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-Object-Meta- headers. For information on custom metadata, see "Custom metadata" on page 20.

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, copying an object, and retrieving an object.

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 Base 64-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 422 (Uprocessable Entity) status code and does not store the object.

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.

PUT object request line

Depending on whether the HSwift request uses a Keystone tenant ID or the account name, the request line for a request to store an object has either of these formats:

- With the Keystone tenant ID:

  PUT /swift/v1/*tenant-ID***/***container-name***/***object-name* **HTTP/1.1**

- With the account name:

  PUT /swift/v1/*account-name***/***container-name***/***object-name* **HTTP/1.1**

PUT object request headers

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

Name	Type	Description
X-Auth-Token	String	See, "Common request headers" on page 43
Content-Encoding	String	Specifies whether the request body is compressed. If so, enter **gzip**.
Content-Length	Integer	Specifies the size, in bytes, of the data being stored.
Etag	String	This value is the MD5 checksum of the object content. It is recommended to compute the MD5 checksum of the response body and compare this value with this ETag header. If they differ, the content may have been corrupted.

(Continued)

Name	Type	Description
Expect	String	Use the Expect header with the value of 100-Continue in combination with the If-None-Match header with the value set to an asterisk (*) to query whether the server already has an object of the same name before any data is sent.
If-None-Match	String	Specifies one or more values for comparison with the ETag of the specified source object. 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 more characters and must be enclosed in double quotation marks ("). Alternatively, you can specify a single asterisk (*) as the value of the If-None-Match header. No ETags match an asterisk in an If-None-Match header.
Transfer-Encoding	String	If this header is used, always set to **chunked**. This header should be used if HCP cannot determine the size of the response body before formulating the response.

(Continued)

Name	Type	Description
X-Copy-From	String	Specifies the source container and object in this format: */container-name/source-object-name* You must UTF-8-encode and then URL-encode the names of the container and object before you include them in the header. Using X-Copy-From locates the source object to copy, and when it is used in conjecture with a **PUT** command X-Copy-From has the same effect as using the **COPY** operation to copy an object. For more information on the **COPY** method, see "Copying an object" on page 94.
X-Object-Meta-name	String	For objects only, specifies a custom metadata property/value pair. The response headers include one X-Object-Meta- header for each applicable pair. For more information on this header, see "Custom metadata" on page 20

PUT object response headers

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

Name	Type	Description
Content-Length	String (Required)	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 is **0** (zero).

(Continued)

Name	Type	Description
Content-Type	String (Required)	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.
ETag	String (Required)	Specifies the ETag for the object.
Date	Datetime (Required)	See, "Common response headers" on page 44.
X-Trans-Id	Uuid (Required)	See, "Common response headers" on page 44.

PUT object 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 46.

Code	Meaning	Description
201	Created	The object has been added to the container.
408	Request Timeout	The request has timed out and not completed. It is no longer trying to resend the request.
409	Conflict	The object exists already and versioning is not enabled.
411	Length Required	Indicates a missing Transfer-Encoding or Content-Length request header.
412	Precondition Failed	If using "If-None-Match: *" header and an object already exists at the specified path.
422	Unprocessable Entity	MD5 checksum of the data that is written to the object store does not match the optional ETag value.

Storing an object example

Here's a sample **PUT** request that stores an object named `Q4_2012.ppt` in the finance container.

Request with curl command line

```
curl -k -i -X PUT http://api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/finance/Q4_2012.ppt -H "X-Auth-Token:
dc5efec8f546455eac974e7bbfd0dd87" -H "Content-Length: 63" -H "Etag:
61d952ad1d4bdb14aa221d046a6289de"
```

Request headers

```
PUT /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e/finance/Q4_2012.ppt
HTTP/1.1
X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87
```

Response headers

```
HTTP/1.1 201 Created
Date: Mon, 20 Oct 2014 20:48:36 GMT
Server: HCP V7.1.0.353
ETag: "1774af03f76ef80fa0c08b9dbd8d19f9"
X-Trans-Id: txfe5244ce53a142288ef97-00541b3b32
Content-Type: application/octet-stream
Content-Length: 0
```

Creating a directory

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

To tell HCP to create a directory instead of an object in response to a **PUT** request, you need to include a forward slash (/) after the directory's name.

When you create a directory, you specify a name for it. The rules for directory names are the same as the rule for object names. For more information on object names, see "Object names" on page 16.

PUT object create directory request line

Depending on whether the HSwift request uses a Keystone tenant ID or the account name, the request line for a request to create a directory has either of these formats:

- With the Keystone tenant ID:

  ```
  PUT /swift/v1/tenant-ID/container-name/directory-name/ HTTP/1.1
  ```

- With the account name:

  ```
  PUT /swift/v1/account-name/container-name/directory-name/
  HTTP/1.1
  ```

PUT object create directory request headers

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

Name	Type	Description
X-Auth-Token	String	See, "Common request headers" on page 43
Content-Encoding	String	Specifies whether the request body is compressed. If so, enter **gzip**.
Content-Length	Integer	Specifies the size, in bytes, of the data being stored.
Etag	String	This value is the MD5 checksum of the object content. It is recommended to compute the MD5 checksum of the response body and compare this value with this ETag header. If they differ, the content may have been corrupted.
Expect	String	Use the Expect header with the value of 100-Continue in combination with the If-None-Match header with the value set to an asterisk (*) to query whether the server already has an object of the same name before any data is sent.

(Continued)

Name	Type	Description
If-None-Match	String	Specifies one or more values for comparison with the ETag of the specified source object. 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 more characters and must be enclosed in double quotation marks ("). Alternatively, you can specify a single asterisk (*) as the value of the If-None-Match header. No ETags match an asterisk in an If-None-Match header.
Transfer-Encoding	String	If this header is used, always set to **chunked**. This header should be used if HCP cannot determine the size of the response body before formulating the response.
X-Copy-From	String	Specifies the source container and object in this format: */container-name/source-directory-name* You must UTF-8-encode and then URL-encode the names of the container and object before you include them in the header. Using X-Copy-From locates the source object to copy, and when it is used in conjecture with a **PUT** command X-Copy-From has the same effect as using the **COPY** operation to copy an object.

PUT object create directory response headers

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

Name	Type	Description
Content-Length	String (Required)	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 is **0** (zero).
Content-Type	String (Required)	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.
ETag	String (Required)	Specifies the ETag for the object.
Date	Datetime (Required)	See, "Common response headers" on page 44.
X-Trans-Id	Uuid (Required)	See, "Common response headers" on page 44.

PUT object create directory 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 46.

Code	Meaning	Description
201	Created	The object has been added to the container.
408	Request Timeout	The request has timed out and not completed. It is no longer trying to resent the request.
409	Conflict	The object already exists and versioning is not enabled.
411	Length Required	Indicates a missing Transfer-Encoding or Content-Length request header.
412	Precondition Failed	If using "If-None-Match: *" header and an object already exists at the specified path.
422	Unprocessable Entity	MD5 checksum of the data that is written to the object store does not match the optional ETag value.

Creating a directory example

Here's a sample **PUT** request that creates a directory named `budget_proposals` in the `r&d` directory in the finance container. If the `r&d` directory doesn't already exist, this request also creates that directory.

Request with curl command line

```
curl -k -i -X PUT http://api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/finance/r&d/budget_proposals/ -H "X-Auth-
Token: dc5efec8f546455eac974e7bbfd0dd87"
```

Request headers

```
PUT /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e/finance/r&d/budget_
proposals/ HTTP/1.1
X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87
```

Response headers

```
HTTP/1.1 201 Created
Date: Fri, 24 Oct 2014 14:29:06 GMT
Server: HCP V7.1.0.353
ETag: d41d8cd98f00b204e9800998ecf8427e
X-Trans-Id: txfe5244ce53a142288ef97-00541b3b32
Content-Type: application/octet-stream
Content-Length: 0
```

Checking the metadata of an object or directory

You use the HTTP **HEAD** method to check the metadata of an object or directory in a container. To check the metadata of an object or directory, you need read permission for the container containing the object or directory.

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

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

- The ETag for the specified item (ETag header). For information on ETags, see "Storing an object" on page 80
- The date the specified item was last modified (Last-Modified header).
- The Internet media type of the specidied item (Content-Type header).
- The size of the specified item, in bytes (and Content-Length header).
- For an object, any applicable custom metadata stored in the .metapairs annotation for that item (X-Object-Meta- headers). For information on X-Object-Meta- headers, see "Custom metadata" on page 20.

HEAD object request line

Depending on whether the HSwift request uses a Keystone tenant ID or the account name, the request line for a request to check the metadata of an object has either of these formats:

- With the Keystone tenant ID:

  HEAD /swift/v1/*tenant-ID***/***container-name***/***folder-name***/***object-name*
  HTTP/1.1

- With the account name:

```
HEAD /swift/v1/account-name/container-name/folder-name/object-
name HTTP/1.1
```

Depending on whether the HSwift request uses a Keystone tenant ID or the account name, the request line for a request to check the metadata of a directory has either of these formats:

- With the Keystone tenant ID:

  ```
  HEAD /swift/v1/tenant-ID-token/container-name/directory-name/
  HTTP/1.1
  ```

- With the account name:

  ```
  HEAD /swift/v1/account-name/container-name/directory-name/
  HTTP/1.1
  ```

HEAD object request headers

The table below describes the headers you can use in a request to check the metadata of an object or directory.

Name	Type	Description
X-Auth-Token	String	See, "Common request headers" on page 43

HEAD object response headers

The table below describes the headers you can use in a request to check the metadata of an object or directory.

Name	Type	Description
Content-Length	String (Required)	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 is **0** (zero).

(Continued)

Name	Type	Description
Content-Type	String (Required)	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	Datetime (Required)	See, "Common response headers" on page 44
ETag	String (Required)	This value is the MD5 checksum value of the object content.
Last-Modified	String (Required)	The date and time that the object was created or the last time that the metadata was modified.
X-Object-Meta-name	String (Required)	For object, specifies a custom metadata property/value pair. The response headers include one X-Object-Meta-name header for each applicable pair. For more information on this header, see "Custom metadata" on page 20.
X-Trans-Id	Uuid (Required)	See, "Common response headers" on page 44

HEAD object HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to check the metadata of an object or directory. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 46

Code	Meaning	Description
200	Success	The object data has been retrieved.
204	Request Timeout	The request has timed out and the operation was not completed. Hswift is no longer retrying the operation.

HEAD object examples

The following sections show sample **HEAD** requests for checking the metadata of objects, and directories.

Retrieving custom metadata for an object

Here's a sample **HEAD** request that checks the metadata of an object named `sales_quotas_2013.pdf`, which has custom metadata.

Request with curl command line

```
curl -k -i -X HEAD http://api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/finance/sales_quotas_2013.pdf -H "X-Auth-
Token: dc5efec8f546455eac974e7bbfd0dd87"
```

Request headers

```
HEAD /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e/finance/sales_quotas_
2013.pdf HTTP/1.1
X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87
X-Object-Meta-author: P.D Grey
```

Response headers

```
HTTP/1.1 201 Created
Date: Fri, 24 Oct 2014 14:29:06 GMT
Server: HCP V7.1.0.353
X-Trans-Id: 156263da-2e0c-49d6-b237-dc7745bd3975
Accept-Ranges: bytes
X-Static-Large-Object: false
ETag: d41d8cd98f00b204e9800998ecf8427e
Last-Modified: Fri, 31 Oct 2014 20:40:16 GMT
X-Object-Meta-author: P.D Grey
Content-Type: application/octet-stream
Content-Length: 0
```

Checking the metadata of a directory example

Here's a sample **HEAD** request that checks the metadata of a directory named `r&d` in the finance directory.

Request with curl command line

```
curl -k -i -X HEAD http://api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/finance/r&d/ -H "X-Auth-Token:
dc5efec8f546455eac974e7bbfd0dd87"
```

Request headers

```
HEAD /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e/finance/r&d/ HTTP/1.1
X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87
```

Response headers

```
HTTP/1.1 200 OK
Date: Fri, 31 Oct 2014 20:55:13 GMT
Server: HCP V7.1.0.353
X-Trans-Id: 05c677f6-2044-412d-a9c6-850298b4971a
Accept-Ranges: bytes
X-Static-Large-Object: false
ETag: d41d8cd98f00b204e9800998ecf8427e
Last-Modified: Fri, 31 Oct 2014 20:10:44 GMT
Content-Type: application/octet-stream
Content-Length: 0
```

Copying an object

You use the HTTP **COPY** method with the Destination header to copy an object from one location to another. The source and target locations can be two different containers within the same tenant, or they can be the same container. 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 container containing the source object or for the source object itself and write permission for the target container.

When copying an object, you can specify a name for the target object that's different from the name of the source object.

Regardless of whether or not versioning is enabled, only the latest version of a source object is taken for the copy.

By default, HCP copies custom metadata for the source object to the target object. However, in the copy request, you can specify additional custom metadata to be used for the target object. To apply this custom metadata to the target object, you need to include the X-Object-Meta-name header in the copy request.

Copying an object to itself
If the target object you specify in a copy request identifies an existing object , HCP replaces the existing object with the new copy. If versioning is disabled the operation fails and a 409 (Conflict) HTTP status code appears.

COPY object request line

Depending on whether the HSwift request uses a Keystone tenant ID or the account name, the request line for a request to copy an object has either of these formats:

- With the Keystone tenant ID:

  ```
  COPY /swift/v1/tenant-ID/container-name/directory-name/object-
  name HTTP/1.1
  "Destination: /container-name/object-name"
  ```

- With the account name:

  ```
  COPY /swift/v1/account-name/container-name/directory-name/object-
  name HTTP/1.1
  "Destination: /container-name/object-name"
  ```

COPY object request headers

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

Name	Type	Description
X-Auth-Token	String	See, "Common request headers" on page 43
Destination	String (Required)	Specifies the destination of the copied container and object, in this format: */container-name/source-object-name* The initial forward slash (/) is required. You must UTF-8-encode and then URL-encode the names of the destination container and object before you include them in this header.

(Continued)

Name	Type	Description
X-Object-Meta-name	String	For objects only, specifies a custom metadata property/value pair. The response headers include one X-Object-Meta- header for each applicable pair. For more information on this header, see "Custom metadata" on page 20

COPY object response headers

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

Name	Type	Description
Content-Length	String (Required)	See, "Common response headers" on page 44.
Content-Type	String (Required)	See, "Common response headers" on page 44.
Date	Datetime (Required)	See, "Common response headers" on page 44.
ETag	String (Required)	This value is the MD5 checksum of the uploaded object content. The value is not quoted. If you supplied an ETag request header and the operation was successful, the values are the same. If you did not supply an ETag request header, check the ETag response header value against the object content you have just uploaded.
Last-Modified	String (Required)	The date and time that the copied object was created or last had its metadata modified.
X-Copied-From	String	Shows the container and object name of the original source object from which the new object was copied. This is the format: *container/object*

(Continued)

Name	Type	Description
X-Copied-From-Last-Modified	String	Shows the last modified date and time for the original source object from which the new object was copied.
X-Object-Meta-name	String	For object, specifies a custom metadata property/value pair. The response headers include one X-Object-Meta-name header for each applicable pair. For more information on this header, see "Custom metadata" on page 20.
X-Trans-Id	Uuid (Required)	See, "Common response headers" on page 44.

COPY 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 46.

Code	Meaning	Description
201	Success	No response body.
408	Request Timeout	The request has timed out and the operation was not completed. Hswift is no longer retrying the operation.
409	Conflict	The object exists already and versioning is not enabled.

Copying an object from one container to another and adding custom metadata example

Here's a sample **COPY** request that copies an object named `campaigns/GoGetEm.xls` from the sales-mktg container to the finance container, where the target object that results from the copy operation is named `mktg/campaign_GoGetEm_expenses.xls` With this copy, the metadata property/value pair business/campaign is added to the object.

Request with curl command line

```
curl -k -i -X COPY http://api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/sales-mktg/campaigns/GoGetEm.xls -H "X-Auth-
Token: dc5efec8f546455eac974e7bbfd0dd87" -H "Destination: finance/mktg/campaign_
GoGetEm_expenses.xls" -H "X-Object-Meta-business: campaign"
```

Request headers

```
COPY /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e/sales-
mktg/campaign/GoGetEm.xls HTTP/1.1
X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87
x-Object-business: campaign
Destination: /finance/mktg/campaign_GoGetEm_expenses.xls
```

Response headers

```
HTTP/1.1 201 Created
Date: Mon, 03 Nov 2014 14:06:29 GMT
Server: HCP V7.1.0.353
X-Trans-Id: 1c66338b-d2e8-49ab-aa46-15f41677a919
ETag: d41d8cd98f00b204e9800998ecf8427e
Content-Type: application/octet-stream
Content-Length: 0
X-Copied_From-Last-Modified:Mon, 03 Nov 2014 14:06:29 GMT
X-Copied-From: /finance/mktg/campaign_GoGetEm_expenses.xls
Last-Modified: Mon, 03 Nov 2014 14:06:29 GMT
```

Retrieving an object

You use the HTTP **GET** method to retrieve an object from a container. Retrieving an object means retrieving the object data. To retrieve an object, you need read permission for the container containing the object or for the object itself.

A request to retrieve an object retrieves the current version of the object. This happens regardless of whether or not versioning is enabled.

If a retrieved object has custom metadata, the headers returned in response to the request include the X-Object-Meta-name header which lists the name of each metadata item.

Retrieving part of an object

You can use the Range header in a **GET** request to retrieve only part of an object. 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,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 the 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 200 (Success) 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 412 (Precondition Failed) status code and does not return any data.
- If you specify an offset of zero, HCP returns a 412 (Precondition Failed) 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 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** request conditional:

- The If-Match and If-None-Match headers compare the ETag for the requested object to one or more values that you specify. Typically, each value is the ETag for an object of interest.
- The If-Modified-Since and If-Unmodified-Since headers compare the date and time the requested object was last modified to a date and time that you specify.

If the requested object 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.

GET object request line

Depending on whether the HSwift request uses a Keystone tenant ID or the account name, the request line for a request to retrieve an object has either of these formats:

- With the Keystone tenant ID:

  ```
  GET /swift/v1/tenant-ID/container-name/directory-name/object-name
  HTTP/1.1
  ```

- With the account name:

  ```
  Get /swift/v1/account-name/container-name/directory-name/object-
  name HTTP/1.1
  ```

GET object request headers

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

Name	Type	Description
X-Auth-Token	String	See, "Common request headers" on page 43

(Continued)

Name	Type	Description
If-Match	Dict	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 (Preconditions 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.
If-None-Match	String	Specifies one or more values for comparison with the ETag of the specified object. 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.

(Continued)

Name	Type	Description
If-Modified-Since	Dict	Specifies a date and time, in Greenwich Mean Time (GMT), for comparison with the date and time the specified object 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.

(Continued)

Name	Type	Description
		• *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.
If-Unmodified-Since	Dict	Specifies a date and time, in Greenwich Mean Time (GMT), for comparison with the date and time the specified object 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.
Range	Dict	Retrieves part of an object. For valid values for this header, see "Retrieving an object" on page 98

GET object response headers

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

Name	Type	Description
Accept-Ranges	String (Required)	Always **bytes**. This header shows the type of ranges that an object accepts from a Range request header.
Content-Encoding	String	Specifies whether the request body is compressed. The only possible value for this response header is **gzip**.
Content-Length	String (Required)	See, "Common response headers" on page 44.

(Continued)

Name	Type	Description
Content-Type	String (Required)	See, "Common response headers" on page 44.
Date	Datetime (Required)	See, "Common response headers" on page 44.
ETag	String (Required)	This value is the MD5 checksum values of the object content.
Last-Modified	String (Required)	The date and time that the object was created or the last time that the metadata was modified.
X-Object-Meta-name	String	For object, specifies a custom metadata property/value pair. The response headers include one X-Object-Meta-name header for each applicable pair. For more information on this header, see "Custom metadata" on page 20.
X-Trans-Id	Uuid (Required)	See, "Common response headers" on page 44.

GET object 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 46.

Code	Meaning	Description
200	Success	The response body contains the object data.
304	Not Modified	If using If-Non-Match header and Etag matches, or If-Modified-Since. The response body may contain information regarding the precondition failure.
403	Forbidden	Authenticated user does not have permission to perform the requested operation
404	Not Found	The object you requested could not be retrieved.

(Continued)

Code	Meaning	Description
412	Precondition Failed	If using If-Match header and Etag does not match, or if-Unmodified-Since. The response body may contain information regarding the precondition failure.

GET object examples

The following sections show sample **GET** requests for retrieving objects.

Conditionally retrieving an object example

Here's a sample **GET** request that retrieves an object named `mktg/campaign_GoGetEm_expenses.xls` in the finance container only if the ETag for the object doesn't match a specified value. In this example, the specified value is the `ETag of mktg/campaign_GoGetEm_expenses.xls`.

Request with curl command line

```
curl -v -X GET http://api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/finance/mktg/campaign_GoGetEm_expenses.xls -
H "X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87" -H "If-None-Match:
d41d8cd98f00b204e9800998ecf8427e"
```

Request headers

```
GET /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e/finance/mktg/campaign_
GoGetEm_expenses.xls HTTP/1.1
X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87
If-None-Match: d41d8cd98f00b204e9800998ecf8427e
```

Response headers

```
HTTP/1.1 304 Not Modified
Date: Fri, 24 Oct 2014 13:03:44 GMT
X-Trans-Id: 854ad794-2a9e-4d02-9012-35e43d3d69ec
Accept-Ranges: bytes
X-Static-Large-Object: false
ETag: "d41d8cd98f00b204e9800998ecf8427e"
Server: HCP V7.1.0.353
Content-Type: text/html;charset=UTF-8
Content-Length: 0
Transfer-Encoding: chunked
```

Retrieving part of an object example

Here's a sample **GET** request that retrieves the first hundred thousand bytes of the object named `quarterly_rpts/Q4_2012.ppt` in the finance container.

Request with curl command line

```
curl -v -X GET http://api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/finance/quarterly_rpts/Q4_2012.ppt -H "X-Auth-
Token: dc5efec8f546455eac974e7bbfd0dd87" -H "Range: bytes=0-99999"
```

Request headers

```
GET /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e/finance/quarterly_rpts/Q4_
2012.ppt HTTP/1.1
X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87
Range: bytes=0-99999
```

Response headers

```
HTTP/1.1 403 Forbidden
Date: Fri, 24 Oct 2014 14:03:39 GMT
Content-Length: 0
Server: Jetty(7.6.0.v20120127)
HTTP/1.1 200 OK

Date: Mon, 03 Nov 2014 14:54:48 GMT
Server: HCP V7.1.0.353
X-Trans-Id: 9cb4f3af-2e8f-498c-9a9a-633b414c99a0
Accept-Ranges: bytes
X-Static-Large-Object: false
ETag: d41d8cd98f00b204e9800998ecf8427e
Last-Modified: Wed, 29 Oct 2014 13:40:34 GMT
Content-Length: 100000
Cache-Control: no-cache,no-store
Pragma: no-cache
Content-Type: application/octet-stream
Content-Disposition: attachment; filename="directory";
```

Deleting an object or directory

You use the HTTP **DELETE** method to delete an object or directory in a container. To check the delete an object or container, you need write permission for the container containing the object or directory.

DELETE object request line

Depending on whether the HSwift request uses a Keystone tenant ID or the account name, the request line for a request to delete an object has either of these formats:

- With the Keystone tenant ID included in the hostname:

  ```
  DELETE /swift/v1/tenant-ID/container-name/directory-name/object-
  name HTTP/1.1
  ```

- With the account name following the hostname:

  ```
  DELETE /swift/v1/account-name/container-name/directory-
  name/object-name HTTP/1.1
  ```

Depending on whether the HSwift request uses a Keystone tenant ID or the account name, the request line for a request to delete a directory has either of these formats:

- With the Keystone tenant ID included in the hostname:

  ```
  DELETE /swift/v1/tenant-ID/container-name/directory-name/
  HTTP/1.1
  ```

- With the account name following the hostname:

  ```
  DELETE /swift/v1/account-name/container-name/directory-name/
  HTTP/1.1
  ```

DELETE object request headers

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

Name	Type	Description
X-Auth-Token	String	See, "Common request headers" on page 43

DELETE object response headers

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

Name	Type	Description
Content-Length	String (Required)	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 is **0** (zero).
Content-Type	String (Required)	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	Datetime (Required)	See, "Common response headers" on page 44.
X-Trans-Id	Uuid (Required)	See, "Common response headers" on page 44.

DELETE object HTTP status codes

The table below describes HTTP status codes that can be returned in response to a request to delete an object or directory. For more information on HTTP status codes and the error codes that can accompany them, see "Error codes" on page 46.

Code	Meaning	Description
204	No Content	No Response Body.
404	Not Found	HTML error message in response body.

DELETE object examples

The following sections show sample **DELETE** requests for deleting objects and directories.

Deleting an object example

Here's a sample **DELETE** request that deletes an object named `hum_res/budget_proposals/BudgProp-2013` from the finance container.

Request with curl command line

```
curl -v -X DELETE http://api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/finance/hum_res/budget_proposals/BudgProp-
2013 -H "X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87"
```

Request headers

```
DELETE /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e/finance/hum_
res/budget_proposals/BudgProp-2013 HTTP/1.1
X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87
```

Response headers

```
HTTP/1.1 204 No Content
Date: Mon, 03 Nov 2014 15:04:57 GMT
Server: HCP V7.1.0.353
Content-Type: text/html;charset=UTF-8
Content-Length: 538
```

Deleting a directory example

Here's a sample **DELETE** request that deletes a directory named `hum_res/budget_proposals/` from the finance container.

Request with curl command line

```
curl -v -X DELETE http://api.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/finance/hum_res/budget_proposals/ -H "X-Auth-
Token: dc5efec8f546455eac974e7bbfd0dd87"
```

Request headers

```
DELETE /swift/v1/AUTH_6b6884ebb6f441cfbb7e740f6a927c9e/finance/hum_
res/budget_proposals/ HTTP/1.1
X-Auth-Token: dc5efec8f546455eac974e7bbfd0dd87
```

Response headers

```
HTTP/1.1 204 No Content
Date: Mon, 03 Nov 2014 15:04:57 GMT
Server: HCP V7.1.0.353
```

6

Usage considerations

This chapter contains considerations that apply to using the HSwift 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 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.

Directory structures

Because of the way HCP stores objects, the directories you create and the way you store objects in them can have an impact on performance. Here are some guidelines for creating effective directory structures:

- Plan the directory structure for a container before storing objects in the container. Make sure all users of the container are aware of these plans.

- Avoid structures that result in a single directory 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 directory structure such as year/month/day/hour/minute/second. If you ingest only a few directory per second, a less fine-grained structure would be better.
- Follow these guidelines on the size of the directory structure:
  - Try to balance the width and depth of the directory structure.
  - Do not create directory structures that are more than 20 levels deep. Instead, create flatter directory structures.
  - Avoid placing a large number of objects (greater than 100,000) in a single directory. Instead, create multiple directories 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 (outside of the HSwift API) for the target container:

- If versioning is enabled, HCP creates one version 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 container 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 container 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 directories. If that's not possible, multiple threads working against a single directories 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 HSwift 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 object, 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 HSwift request if either of these happens:

- The client cannot establish an HSwift 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 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 HSwift Content-Length response header. If the numbers match, the returned data is complete.

A

Quick reference

This appendix contains a quick reference to the HSwift 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 request that use the method. The required headers are listed first followed by optional headers enclosed in square brackets.

The methods are grouped by the level of access (HSwift account (that is, tenant), container, or object). Within each grouping, the methods are presented in alphabetical order.

For general information on HSwift requests, see Chapter 3: "Requests and responses" on page 41

Description and permission	**Query parameters**	**Request headers**
Account-level operation		
GET account		

(Continued)

Description and permission	Query parameters	Request headers

Description: Returns a list of containers owned by the requester.
 	
Permission: None; Requester must be an authenticated user.
 	
More information: "Listing containers" on page 56
	limit = integer
	
	marker=string
	
	end_marker=string
	
	format=string
	
	prefix=string
		"X-Auth-Token: *Keystone-Authentication-Token*"
		
		"Accept: application/xml
		or
		application/json
		or
		text/xml"

HEAD account		
Description: Returns the number of containers, objects, and bytes on an account. **Permission:** None; Requester must be an authenticated user. **More information:** "Listing account metadata" on page 54	N/A	"X-Auth-Token: *Keystone-Authentication-Token*"
Container-level operations		
DELETE container		

(Continued)

Description and permission	Query parameters	Request headers
Description: Deletes a container. **Permission:** Write. **More information:** "Deleting a container" on page 75	N/A	"X-Auth-Token: *Keystone-Authentication-Token*"
GET container		

Description: Lists object and directories in a container, along with the container's acl.
 	
Permission: Browse.
 	
More information: "Listing container contents" on page 69
	limit = integer
	
	marker=string
	
	end_marker=string
	
	prefix=string
	
	format=string
	
	delimiter=character
	
	path=string
		"X-Auth-Token: *Keystone-Authentication-Token*"
		
		"Accept: application/xml
		or
		application/json
		or
		text/xml"

HEAD container		

(Continued)

Description and permission	**Query parameters**	**Request headers**
Description: Checks the existence of a container. **Permission:** Read. **More information:** "Checking a container's metadata" on page 61	N/A	"X-Auth-Token: *Keystone-Authentication-Token*"
POST container		
Description: Adds an ACL to a container and sets the container quota. **Permission:** Write. **More information:** "Adding an ACL to a container (data access permission only)" on page 64	N/A	"X-Auth-Token: *Keystone-Authentication-Token*" "X-Container-Read: *Keystone-account-user* OR *" "X-Container-Write: *Keystone-account-user* OR *" "X-Container-Meta-Quota-Bytes: *Bytes-Quota*" "X-Remove-Container-*name*"
PUT container		

(Continued)

Description and permission	Query parameters	Request headers
Description: Creates a container and optionally adds an acl to it. **Permission:** Write. **More information:** "Creating a container" on page 50	N/A	"X-Auth-Token: *Keystone-Authentication-Token*" "X-Container-Read: *Keystone-account-user* OR *" "X-Container-Write: *Keystone-account-user* OR *" "X-Container-Meta-Quota-Bytes: *Bytes-Quota*"
Object-level operations		
COPY object		
Description: Creates a copy of an object and its metadata. **Permission:** Write. **More information:** "Copying an object" on page 94	N/A	"X-Auth-Token: *Keystone-Authentication-Token*" "Destination: */container-destination/object-destination*" "X-Object-Meta-*name*"
DELETE object		

(Continued)

Description and permission	Query parameters	Request headers
Description: Deletes an object or directory. **Permission:** Write. **More information:** "Deleting an object or directory" on page 108	N/A	"X-Auth-Token: *Keystone-Authentication-Token*"
GET object		
Description: Retrieves an object. **Permission:** Read. **More information:** "Retrieving an object" on page 98	N/A	"X-Auth-Token: *Keystone-Authentication-Token*" "Range: *first-byte-offset-last-byte-offset*" "If-Match: "*ETag-value*[,"*ETag value*"]" "If-None-Match: "*ETag value*[,"*ETag value*"]" "If-Modified-Since: *DDD, dd MMM yyyy HH:mm:ss* (+0000|GMT) OR *DDDD, dd-MMM-yy HH:mm:ss* (+0000|GMT) OR *DDD MMM d HH:mm:ss yyyy*" "If-Unmodified-Since: *DDD, dd MMM yyyy HH:mm:ss* (+0000|GMT) OR *DDDD, dd-MMM-yy HH:mm:ss* (+0000|GMT) OR *DDD MMM d HH:mm:ss yyyy*"
HEAD object		

(Continued)

Description and permission	Query parameters	Request headers
Description: Checks the existence of an object or directory. **Permission:** Read. **More information:** "Checking the metadata of an object or directory" on page 90	N/A	"X-Auth-Token: *Keystone-Authentication-Token*"
PUT object or directory		
Description: Stores an object or directory. **Permission:** Write. **More information:** "Storing an object" on page 80"Creating a directory" on page 85	N/A	"X-Auth-Token: *Keystone-Authentication-Token*" "Content-Length: *data-size-in-bytes*" "Transfer-Encoding: chunked" "X-Copy-From: *Container/Object*" "ETag: *MD5-checksum-value*" "Content-Encoding: gzip" "X-Object-Meta-*name*" "If-None-Match: "*ETag-value*[,"*ETag-value*"]" "Except: 100-Continue"

B

Keystone integration

Keystone is an OpenStack identity service that supports token-based authorization. Keystone generates authentication tokens with a predetermined expiration timer that are used to identify users attempting to store and manage containers and objects.

An HCP system can be configured to integrate with Keystone. With the system configured this way, you can use Keystone to authenticate and authorize users and their incoming storage management requests.

This manual assumes that you have a basic understanding of Keystone functionality and that Keystone is already installed on your computer. It also assumes that if you have the administrator role on HCP and have access to the HCP System Management Console.

This chapter covers the following:

- Selecting a Keystone Identity Service URL
- Creating a Service User's Username and Password
- Granting Keystone Service user the admin role
- Choosing the Keystone Tenant ID Prefix
- Enabling and configuring Keystone on the HCP System Management Console
- Enabling the HSwift service on Keystone
- Creating an HCP tenant
- Creating an Keystone HCP tenant and user

For information on alternative methods of authentication, see Appendix C: "Alternate authentication methods" on page 137.

Finding the Identity Service URL

The **Identity Service URL** is the Keystone endpoint with which HCP communicates. There are two different types of **Identity Service URL**s available on Keystone, the Public Identity Service URL and the Admin Identity Service URL. It's recommended to use the Admin Identity Service URL since the Public Service URL does not support user ACLs in this configuration.

Once you have chosen which identity service endpoint you want to use, follow these steps to retrieve the url:

1. Using the Python Keystone client, enter one of the following depending on whether you are looking for the public or admin URL:

   – If you want the public url, enter *keystone endpoint-get --service identity --endpoint-type publicURL*

   – If you want the admin url, enter *keystone endpoint-get --service identity --endpoint-type adminURL*

2. Record the URL for later use

Creating the Service User's Name and Password

The **Service User's Name** and **Password** are a set of credentials that HCP uses to authenticate itself with Keystone. The **Service User's Name** and **Password** should be set on the Keystone services tenant. It is recommended to make a new service user for HCP.

To create a Service User's name on Keystone:

1. On your Keystone Python client, enter *Keystone tenant-list*

2. Copy the **Services Tenant Id**

3. Enter **`keystone user-create --name`** *`New-Service-User's-username`* **`--pass`** *`New-Service-User's-password`* **`--tenant-id`** *`services-tenant-id`*

A Property/Value table appears confirming the creation of a new service user.

4. Record your new **Service User's Name** and **Password** for future use.

Granting the Keystone Service User the admin role

Once you create a Keystone services tenant user, you want to grant the user the Keystone admin role. The service tenant user must have the admin role in order to validate tokens and grant access to tenant and namespaces on HCP.

In order to grant the admin role to the service user:

1. In your Keystone Python client, enter **`keystone user-role-add --user`** *`service-tenant-username`* **`-- role admin --tenant services`**

Configuring Keystone on the System Management Console

To configure HCP to use Keystone, you need to supply it with an Identity Service URL, the Service User's Name and Password, and the Service User's Tenant. To do this you need the system administrator role and access to the HCP System Management Console.

In order to connect HCP to Keystone:

1. In the System Management Console, in the top-level menu, mouse over **Security** to display a drop down menu.

2. In the secondary menu, click on **OpenStack**.

3. On the **OpenStack Identity Service** page, select **Enable OpenStack Identity Service**.

   The **Configuration Settings** section appears.

4. Enter the following information:

   – **Identity Service URL** - For more information on the Identity Service URL, see "Finding the Identity Service URL" on the previous page.

- **Service User's Name** - For more information on the Service User's Name, see "Creating the Service User's Name and Password" on page 128.
- **Service User's API Key/Password** - For more information on the Service User's API Key/Password, see "Creating the Service User's Name and Password" on page 128.
- **Service User's Tenant** - The Service User's Tenant is the tenant on which you made your Keystone Tenant User. The tenant is called services.
- **Tenant ID Prefix** - The default Keystone Tenant ID Prefix is AUTH_. When HCP sees the Keystone Tenant ID Prefix in the HSwift account portion of a URL, HCP knows that the value that follows the prefix is a Keystone Tenant ID. For more information, see Chapter 1: "Resource path" on page 1.

5. Click on the **Test** button.
   - If the connection is unsuccessful, you receive a warning message stating that the operation cannot be completed. Reenter the information and continue.
   - If the connection is successful, you receive a successful connection message.
6. Once the connection is established, click on the **Update Settings** button.

Setting up a Keystone HSwift service

You need to have an object store service registered with Keystone in order to integrate HCP with Keystone. To register HSwift as an endpoint, you need to identify the Keystone service ID of the object-store service. Here is the command that lists the Keystone ID of the Swift service:

```
keystone service-get swift
```

To add a new HSwift service or create an HSwift and keep your current Swift service, follow these steps:

1. To register HCP as an endpoint with Keystone use the Keystone service create command where the service-id is the object-store service id identified in the previous step. The actual values for public, internal, and

admin URL may be found in the System Management Console on the **Openstack** page. The command is:

```
keystone endpoint-create \
 --region=region \
 --service-id=id_from_previous \
 --publicurl=https://api.hcp.example.com/swift/v1/AUTH_%(tenant_id)
s'\
 --internalurl= https://api.hcp.example.com/swift/v1/AUTH_%(tenant_
id)s'\
 --adminurl=https://api.hcp.example.com:8000/
```

If you are setting up HCP as a secondary object-store endpoint, you need to specify a unique `region` for the endpoint. Setting a different region allows you to have two swift endpoints configured for your Keystone Swift service.

Creating an HCP tenant

You only need to create an HCP tenant if one doesn't already exist. If you need to create an HCP tenant, see *Administering HCP* for more information.

In order for the HCP tenant to work with HSwift, the Management API needs to be enabled for the HCP tenant. MAPI is enabled through the Tenant Management Console. For more information, see *Managing a Tenant and its Namespace.*

Note: The tenant you create on HCP needs to have a name that is identical to its Keystone counterpart.

Note: If you rename the HCP tenant you must also rename its Keystone Tenant counterpart. Keystone authentication only works for HCP tenants that have a matching Keystone tenant.

Creating a Keystone HCP tenant and user

Once you have an HCP tenant, a Keystone HCP tenant with a name identical to the HCP tenant needs to be created. In order to create a Keystone HCP tenant:

1. In the Keystone client, enter **`keystone tenant-create --name`** *`hcp-tenant-name`*

Note: The tenant you create on Keystone needs to have a name that is identical to its HCP counterpart.

2. Add a user to the tenant by entering **`keystone user-create --name`** *`tenant_username`* **`--pass`** *`tenant-password`*

3. Once the user is created, grant the user the admin role by entering **`keystone user-role-add --user`** *`tenant_user`* **`--tenant`** *`hcp-tenant-name`* **`--role admin`**

Admin is the only Keystone role that HCP recognizes. If Keystone does not have the admin role then it needs to be defined.

Clearing the OpenStack Identity Service cache

Keystone validated tokens are cached so that all successive commands sent with the same token do not need to be re-validated. Caching also saves the Keystone Tenant Id and name mapping, making it so that HCP doesn't have to search for Keystone Tenant Ids with each request. Changes made to Keystone User roles or Keystone Tenant names are not reflected on HCP unless the cache is cleared or the token expires.

In order to clear your Identity Service cache so that tokens are re-validated:

1. On the System Management Console, in the top-level menu, mouse over **Security** to display a drop down menu.

2. In the secondary menu, click on **OpenStack**.

3. On the **OpenStack Identity Service** page, click on the **Clear Cache** button.

Keystone certificates

When connecting to Keystone through HTTPS, Keystone provides an SSL certificate which, if not signed by a trusted authority, must be manually accepted. Once you agree to trust the certificate it's cached for each future connection attempt to the Keystone server. Alternatively, you can manually upload the Keystone SSL certificate from your local machine.

When connecting to Keystone through HTTPS and configuring the Keystone identity service URL on HCP, you must enter the domain name (not the IP address) of the of the Keystone host. This domain name must match the **Subject Common Name** in the Keystone SSL certificate. Using the IP for an SSL connection to Keystone fails because the IP doesn't match the certificate Common Name. Additionally, the identity service endpoint URLs registered in the Keystone service must be registered with the domain name matching the Common Name in the SSL certificate.

Any Keystone SSL certificates can be deleted from the **OpenStack** page of the System Management Console.

Getting a Keystone Authentication Token

To get a Keystone Authentication Token, enter the following command in your Keystone client:

```
curl -X POST http://keystone.example.com:5000/v2.0/tokens \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d '{"auth": {"tenantName": "tenant-name","passwordCredentials":
{"username": "tenant-username", "password": "password"}}}'
```

In this particular example the credentials are requested in json format. The Keystone response looks like this:

```
{
   "access": {
      "serviceCatalog": [
      {
         "endpoints": [
         {
            "adminURL": "https://admin.hcp1.example.com:8000/",
            "id": "76ce30ce374a43d2812f6a78796fe6fa",
            "internalURL": "http://api.hcp1.example.com/swift/v1/AUTH_
            50c989a5206a46748d0985163f25b14b",
            "publicURL": "http://api.hcp1.example.com/swift/v1/AUTH_
            50c989a5206a46748d0985163f25b14b",
            "region": "New York"
         },
         {
            "adminURL": "HTTP://swift.example.com:8080",
            "id": "230f1ea7676d48079bea0a9edabcd88f",
            "internalURL": "HTTP://swift.example.com:8080/v1/AUTH_
            50c989a5206a46748d0985163f25b14b",
            "publicURL": "HTTP://swift.example.com:8080/v1/AUTH_
            50c989a5206a46748d0985163f25b14b",
            "region": "Los Angeles"
            }
         ],
         "name": "hswift",
         "type": "object-store"
      },
      {
         "endpoints": [
         {
            "adminURL": "https://keystone.example.com:35357/v2.0",
            "id": "48aa3755d8a549f6bda22b00fa9cde94",
            "internalURL": "https:// keystone.example.com:5000/v2.0",
            "publicURL": "https:// keystone.example.com:5000/v2.0",
            "region": "New York"
         }
      ],
         "name": "keystone",
         "type": "identity"
      }
   ],
   "token": {
      "expires": "2014-11-19T22:26:57Z",
      "id": "05c20875e3f2430ea10f45623c78cadd",
      "tenant": {
         "id": "50c989a5206a46748d0985163f25b14b",
         "name": "tenant-name"
      }
   },
   "user": {
      "id": "0d47cc2ba7744c4d97220983ae31f3b9",
      "name": "tenant-user",
      "roles": [
         {
            "name": "admin"
         }
      ],
      "username": " tenant-user"
      }
   }
    }
```

The json response contains named elements and named lists. The keystone token, that is passed to HCP in the X-Auth-Token header, can be found in the `id` element inside the token element inside the `access` element.

The authentication response from keystone also contains a `ServiceCatalog` list which lists the endpoints for all services integrated with Keystone.

Using a Keystone Authentication Token

A Keystone Authentication Token is sent with an HSwift command to verify that a user has the appropriate permissions required to perform the requested function. In order to send a Keystone Authentication Token you need to include the X-Auth-Token header with your command and add your token to the header.

Here is an example of how the X-Auth-Token is formatted:

```
"X-Auth-Token: Keystone-Authentication-Token"
```

Here is an example of how the X-Auth-Token is used in a cURL command:

```
curl -X PUT http://api.europe.hcp.example.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e/finance -i -H "X-Auth-Token:
dc5efec8f546455eac974e7bbfd0dd87"
```

C

Alternate authentication methods

HSwift supports multiple methods of user authentication. OpenStack Keystone is the recommended method but HCP can also be accessed through Local Authentication.

Creating a Temporary Authentication token

Temporary Authentication converts an HCP user account username and password into an token that can be used by RESTful APIs to authenticate with HCP. The token can be used with the HSwift X-Auth-Token header to validate a user without the use of Keystone.

The Temporary Authentication token is authenticated directly by HCP and is composed of an HCP user account username encoded in base64 and HCP user account password encoded in an md5 hash. The token does not expire unless the HCP user account username or password changes.

To create a Temporary Authentication token you enter the HSwift X-Auth-User header with your HCP user account username and the X-Auth-Key header with your HCP user account password in your next command.

HCP responds with a 204 HTTP status code and two headers, X-Auth-Token and X-Storage-Url. The X-Auth-Token is your Temporary Authentication token and X-Storage-Url is the URL of your HCP.

Here is an example of a curl command using Temporary Authentication and a valid response. In this example the HCP user account username is `lgreen` and the password is `start123`:

Curl command request

```
curl -v -X HEAD http://api.example.hcp.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e -H "X-Auth-User: lgreen" -H "X-Auth-Key:
start123"
```

Response if user is valid

```
HTTP/1.1 204 No Content
X-Auth-Token: HCP bGdyZWVu:a3b9c163f6c520407ff34cfdb83ca5c6
X-Storage-URL: https://example.com/swift/v1/tenant
```

Using Local Authentication

Local Authentication is an authentication method for users using RESTful APIs. When using Local Authentication, HCP forgoes the use of any third party identity service such as Keystone and instead verifies the user directly. To verify a user, Local Authentication requests tokens that are encoded with HCP user account credentials.

In order to use Local Authentication with HSwift, instead of passing a Keystone authentication token with the X-Auth-Token header, you replace the Keystone authentication token with the prefix HCP, followed by your Local Authentication token.

The Local Authentication token can be generated by using the Temporary Authentication service, see "Creating a Temporary Authentication token" on page 137, or by manually encoding your HCP user account username in base64 and your HCP user account passwork in an md5 hash. The username and password are separated by a colon (:) with no spaces in between.

Local Authentication does not authenticate HCP user accounts that are associated with different tenants than the one you're targeting. To access a tenant with Local Authentication, the management API must be enabled for that tenant.

Containers created using Local Authentication are not assigned to an HCP user account. Containers and objects stored through HSwift are not owned by an HCP user account. If you want to assign containers and objects to HCP user accounts, it must be done through a different interface, see "Other container access methods" on page 5.

Here is an example of how the X-Auth-Token header is formatted when using Local Authentication:

"**`X-Auth-Token: HCP`** *`base64-encoded-username`*`:`*`md5-encoded-password`*"

Here is an example of a curl request that uses a Local Authentication. The username for this HCP user account is `lgreen` and the password is `start123`:

Curl command request

```
curl -v -X HEAD http://api.example.hcp.com/swift/v1/AUTH_
6b6884ebb6f441cfbb7e740f6a927c9e -H "X-Auth-Token: HCP
bGdyZWVu:a3b9c163f6c520407ff34cfdb83ca5c6"
```

Response if token is valid

```
HTTP/1.1 200 OK
```

Response if token is invalid

```
HTTP/1.1 401 Unauthorized
```

D

Using third party tools

Hswift is compatible with other OpenStack tools and the third-party tools CloudBerry and Cyberduck. 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 HSwift.

General setup information for third-party tools

Typically, when configuring a third-party tool to work with HSwift, 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 containers and the object 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.

E

Sample JOSS application

This appendix contains a sample JOSS application that uses the HCP HSwift API and and JOSS SDK to perform a series of operations in HCP.

Assumptions

The application makes these assumptions:

- The HCP system has a tenant named europe.
- Keystone is installed and has a Keystone HCP tenant account named europe
- The Keystone tenant has an admin user with the username lgreen and the password p4ssw0rd
- The tenant has a user account with username lgreen and password p4ssw0rd.
- By default, versioning is disabled for new containers.
- The local file system has folders named `input` and `output` that are located in the current working folder for the application.
- The `input` directory contains two files, `Q4_2012.ppt` and `Q3_2012.ppt`.

What the application does

The sample application shown in this appendix uses the HSwift API to:

1. Create a container named finance in the context of the tenant named europe (the service point)

2. List the container for the europe tenant that are owned by the user lgreen
3. Add an ACL to the finance container
4. Store an object named quarterly_rpts/Q4_2012.ppt in the finance container, associating custom metadata with the object in the process
5. Store an object named quarterly_rpts/Q3_2012.ppt in the finance container
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 container named finance for the account named rsilver
8. Retrieve object quarterly_rpts/Q3_2012.ppt 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 object from the finance container
10. Delete the `quarterly_rpts` folder from the finance container (HCP created this folder automatically when the first object was stored)
11. Delete the finance container

Required library

To run the sample application presented in this appendix, you need to have installed the following library:

- JOSS SDK 0.9.8 downloadable from:

  http://joss.javaswift.org/

JOSS application

Here's the same JOSS application.

```
/**
* This sample Java application shows how to use the HCP
*HSwift API, which is compatible with JOSS SDK. The
*application uses the JOSS SDK.
*/
```

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

import org.javaswift.joss.client.factory.AccountConfig;
import org.javaswift.joss.client.factory.AccountFactory;
import org.javaswift.joss.client.factory.AuthenticationMethod;
import org.javaswift.joss.model.Account;
import org.javaswift.joss.model.Container;
import org.javaswift.joss.model.StoredObject;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;

public class HSwiftSampleApp {

    /**
    * @param args
    */

    public static void main(String[] args) {

    /*
    * Initialize access credentials for the HSwift client.
    */

    // base64 of HCP user name: "lgreen"
    String accessKey = "bGdyZWVu";
    // md5 of HCP user password: "p4ssw0rd"
    String secretKey = "2a9d119df47ff993b662a8ef36f9ea20";

    /*
    * Build the JOSS account connection to be used
    * for communication with HCP. The Keystone must have *an admin user
    lgreen with password p4ssw0rd and *tenant europe. The AuthUrl must
    point to the keystone *server.
    */

    AccountConfig accountConfig = new AccountConfig();
    accountConfig.setAuthUrl
    ("http://api.keystone.server.com:5000/v2.0/token");
    accountConfig.setUsername(accessKey);
    accountConfig.setPassword(secretKey);
    accountConfig.setAuthenticationMethod
    (AuthenticationMethod.KEYSTONE);
    Account europe = new AccountFactor(accountConfig).createAccount();

    /*
    * Now that the HSwift Client is created for HCP
    *usage, proceed with some operations.
    */
```

```
String containerName = "finance";
try {

    /*
    * Create a new container. With HCP, the container *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 container " + containerName + "\n");
    Container finance = europe.getContainer(containerName);
    finance.create();

    /*
    * List the containers you own at the service point.
    */

    System.out.println("Containers:");
    for (Container container : europe.list()) {
    System.out.println(" * " + container.getName());
    }

    System.out.println();

    /*
    * Add an ACL to the container to give read to a user with the
    specified user name.
    */

    finance.setContainerRights(null, "pdgray");

    /*
    * Upload a couple of objects to the container from *files on the
    local file system.
    */

    String firstFileName = "input/Q4_2012.ppt";
    String firstObjectName = "quarterly_rpts/Q4_2012.ppt";
    StoredObject Q4_2012 = finance.getObject(firstObjectName);

    System.out.println("Uploading first object to HCP from a
    file\n");
    Q4_2012.uploadObject(new FileInputStream(firstFileName));

    // Add metadata for first object
    Q4_2012.setAndSaveMetadata("Author", "P.D. Gray");
    Q4_2012.setAndSaveMetadata("Audit_Date", "2013-02-23");
```

```
// Write a second object without metadata.
System.out.println("Uploading second object to HCP from a
file\n");
String secondFileName = "input/Q3_2012.ppt";
String secondObjectName = "quarterly_rpts/Q3_2012.ppt";
StoredObject Q3_2012 = finance.getObject(secondObjectName);
Q3_2012.uploadObject(new FileInputStream(secondFileName));

/*
* List objects in the container with prefix *quarterly_rpts/Q.
The objects listing is limited *to 1,000 items per request.
*/

System.out.println("Objects:");
for (StoredObject object : finance.list("quarterly_rpts/Q", null,
1000)) {
     System.out.println(" * " + object.getName() + " " +
          "(size = " + object.getContentLength() + ")");
}

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");
InputStream responseStream = Q4_2012.downloadObjectAsInputStream
();

// Write the content to a file named Q4_2012.ppt in the output
folder.
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 container which gives full control to the
rsilver account.
*/

finance.setContainerRights("rsilver", "rsilver");
```

```
/*
* Perform a download of the second object.
*/

System.out.println("Checking the second object");
responseStream = Q3_2012.downloadObjectAsInputStream();

// Download it to a new file.
System.out.println("Downloading new revision\n");
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
          = responseStream.read(readBuffer))) {
     dataFile2.write(readBuffer, 0, conditionalReadSize);
}
dataFile2.close();

/*
* Delete the objects.
*/

System.out.println(
     "Deleting the objects created by this sample
application\n");
Q4_2012.delete();
Q3_2012.delete();

/*
* Delete the folder.
*/

System.out.println(
          "Deleting the folder created when the first object was
stored\n");
StoredObject folder = finance.getObject("quarterly_rpts");
folder.delete();

/*
* Delete the container.
*/

System.out.println("Deleting the finance container\n");
finance.delete();

} 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());
            }
        }
    }
```

Glossary

A

access control list (ACL)

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

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

ACL

See "access control list (ACL)" above.

annotation

A discrete unit of custom metadata.

authenticated access

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

C

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.

container

The HSwift term for a namespace.

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 repurpose object content.

D

domain

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

domain name system

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

DNS

See "domain name system" above.

E

ETag

An identified 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

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.

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 HCP 2.x archive.

HCP-DM

See "HCP Data Migrator (HCP-DM)" above.

HCP metadata query API

See "metadata query API" on page 158.

HCP tenant

An administrative entity created on HCP for the purpose of owning and managing namespaces. Tenants typically correspond to customers or business units. On Keystone, HCP tenants are represented as Keystone HCP tenants, see"Keystone HCP tenant" on page 157. On the HSwift API, HCP tenants are represented as HSwift accounts, see "HSwift account" on the next page.

HCP user account

A set of credentials that gives a user access to one or more of the System Management Console, the Tenant Management Console, the HCP management API, the HCP Search Console, namespace content through the namespace access protocols, the metadata query API, and HCP Data Migrator.

The HCP user account is used to generate Temporary Authentication tokens, see "Temporary Authentication" on page 162, and authenticate users through Local Authentication, see "Local Authentication" on page 158.

The HCP user account cannot be accessed, or be made the owner of a container through the HSwift API.

HDDS

See "hitachi data discovery suite (HDDS)" on the next page

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 data discovery suite (HDDS)

A Hitachi product that enables federated searches across multiple HCP systems and other supported systems.

host

A physical computer on which virtual machines are installed and run.

HSwift account

An HSwift account is the representation of an HCP tenant, see "HCP tenant" on the previous page, or Keystone HCP tenant, see "Keystone HCP tenant" on the facing page, on HSwift. The HSwift account holds containers, the way an HCP tenant holds namespaces. Unlike and HCP tenant, HSwift account have no owner unless assigned one outside of the HSwift API.

HSwift API

One of the namespace access protocols supported by HCP. HSwift is a RESTful, HTTP-based API that is compatible with OpenStack. Using HSwift, users and applications can create and manage containers and container 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" above and "SSL server certificate" on page 161.

J

JOSS

Java OpenStack Storage. A Java client for the OpenStack Storage content REST interface.

JSON

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

K

Keystone

Keystone is an OpenStack identity service that uses token-based user authentication. Keystone generates authentication tokens with a predetermined expiration timer that are used to identify users trying to store and manage containers and objects. The service can be configured to integrate with HCP.

Keystone HCP tenant

The Keystone HCP tenant is a tenant on Keystone that mirrors your HCP tenant, see "HCP tenant" on page 155. The Keystone HCP tenant must have an identical name to your HCP tenant (case sensitivity and spacing included) in order for the HCP tenant to be registered on Keystone.

On HCP, Keystone HCP tenants are represented as HCP tenants. On the HSwift API, Keystone HCP tenants are represented as the HSwift account "HSwift account" on the previous page.

Keystone services tenant

The Keystone services tenant is an automatically generated Keystone tenant that is used to authenticate and catalog the different services registered with Keystone. The users on the Keystone service tenant represent the service itself instead of an actual user. See, "Keystone services user" below.

Keystone services user

The Keystone services user is added to the Keystone services user tenant to register HCP with Keystone. The Keystone services user itself does not represent an actual user, but rather services that are registered and can be authenticated with Keystone. See, "Keystone services tenant" above.

L

Local Authentication

A form of authentication that uses an HCP distributed Temporary Authentication token to authorize a user commands. The token is validated by HCP instead of any third party service.

M

metadata

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

metadata query API

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

N

namespace

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

namespace access protocol

A protocol that can be used to transfer data to and from namespaces in an HCP system. HCP supports the HTTP, HSwift, HS3, WebDAV, CIFS, NFS, and SMTP protocols.

NFS

See "network file system" on page 1.

node

A server or virtual machine running HCP-VM software. Two nodes are networked together to form an HCP-VM 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 services, including shredding, indexing, versioning, and replication.

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 directory that contains the object).

Object-based queries searches 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 delete, 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 examples, created or deleted), change time, and location (that is, the namespace and/or directory that contains the object).

P

permission

One of these:

- In a data access permission, the condition of allowing a specific type of operation to be performed in a namespace.

– In a HCP user account, the granted ability to perform a specific type of operation in a given namespace.

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

Q

query

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

R

repository

The aggregate of the namespaces defined for an HCP system.

resource path line

The resource path line for an HSwift request specifies the operation target, whether it be an HSwift account, container, or object.

REST

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

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

REST API

One of the namespace access protocols supported by HCP. The REST API is also called the HTTP protocol.

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 an HCP user account.

service point

A tenant that serves as the point of access to HCP in an HSwift API request.

SSL

See "secure sockets layer" on page 1.

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.

Subject Common Name

The Common Name found in the subject of a SSL certificate. The Common Name must match the host name of the host providing the certificate.

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.

system management console

The system-specific web application that lets you monitor and manage HCP.

T

Temporary Authentication

A command that requests an authorization token from HCP that is created from Base64 encoding of an HCP user account username and MD5 encoding of an HCP user account password.

tenant management console

The tenant-specific web application that lets you monitor and manage tenants and namespaces.

U

unix

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

user authentication

The process of checking that the combination of a specified username and password is valid when a user tries to log into the System Management Console, the Tenant Management Console through the management API, or to access a namespace.

V

versioning

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

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

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-92ARC041-00