NetStorage HTTP API Specification

(For FileStore)

October 29, 2014

Akamai Technologies, Inc

Akamai Customer Care: **1-877-425-2832** or, for routine requests, e-mail: ccare@akamai.com
The Akamai Luna Control Center, for customers and resellers: http://control.akamai.com

US Headquarters
8 Cambridge Center
Cambridge, MA 02142

Tel: 617.444.3000
Fax: 617.444.3001

US Toll free 877.4AKAMAI (877.425.2624)

For a list of offices around the world, see:
http://www.akamai.com/html/about/locations.html

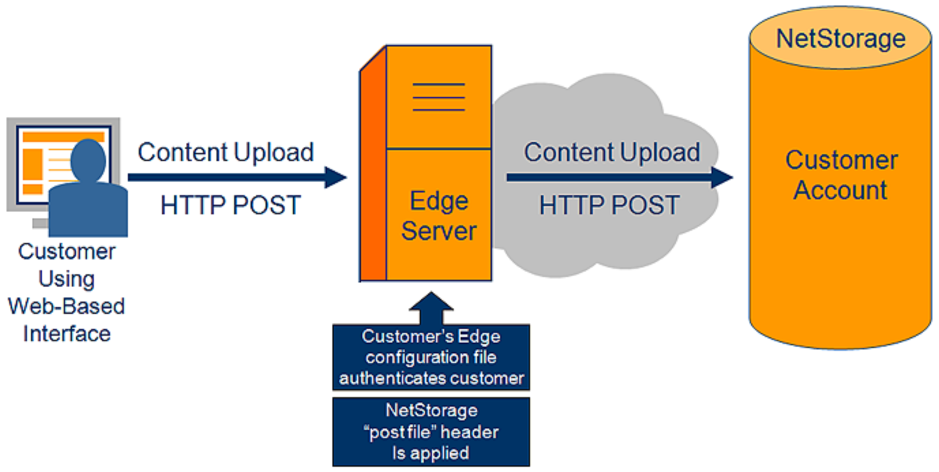
Akamai® NetStorage HTTP API Specification (For FileStore)

Contents

Preface

Welcome to the **NetStorage HTTP API Specification**. This API is a NetStorage feature that provides an HTTP POST method you can use to manage your content. Communication is conducted via the Edge network using a Web-based user interface of your own design. With this feature, you can perform operations such as uploading, deleting, and listing of content, thus providing you one more options for interfacing with your NetStorage account. Also, since communication is Web-based, you can apply security through the use of Secure Sockets Layer (SSL).

The following illustrates the flow of an upload request using this API:


Customer Using Web-Based Interface
Content Upload
HTTP POST
Edge Server
Content Upload
HTTP POST
NetStorage
Customer Account
Customer's Edge configuration file authenticates customer
NetStorage "post file" header Is applied

CAUTION: *With regard to the various operations available, be aware that only one operation per request is supported.*

Intended Audience

This documentation is intended for NetStorage customers who would prefer working directly with an API interface to perform various NetStorage-related operations that are targeting FileStore-format Storage Groups. Only those individuals that are experienced with the use of an API (and the generation of an applicable Web-based user interface required by this component) should attempt its use.

Additional Resources

While not required to use the components covered in this guide, the sections that follow discuss additional documentation and resources that can assist with its use.

Recommended Reading

- *NetStorage - Configuration Guide* **(for FileStore)** - Covers the use of the **NetStorage**:**Configuration** utility in the Luna Control Center, which is used to manage your NetStorage Storage Groups and associated Upload Accounts.

  https://control.akamai.com/dl/customers/NS/NS_Config_FS.pdf

Software Development Kits (SDKs)

The links that follow offer access to language-specific libraries that may assist in interacting with the NetStorage HTTP API.

IMPORTANT: *These components are not maintained/regulated by Akamai. While they can be incorporated to assist you in API use, Akamai does not support them (i.e., Akamai CCare will not offer support in their use), and Akamai can not be held liable if issues arise from their use.*

- **NetStorageKit (for .NET/c#)**:

  https://github.com/akamai-open/NetStorageKit-C-Sharp

- **NetStorageKit (for Java)**:

  https://github.com/akamai-open/NetStorageKit-Java

Using the API

In This Chapter

Pre-Requisites

The sections that follow document various pre-requisites that must be met before you can start using the **Akamai NetStorage HTTP API** (hereinafter, simply referred to as the "API").

Ensuring Port Availability

To allow access to the API through a Firewall, the following ports must be open:

- **Outgoing Access** - Ports 80/443
- **Incoming Access** - Dynamic range on the local system

Provisioning Access to the API

Access to the API must be established within a NetStorage Upload Account, prior to API use. This is accomplished via the **NetStorage:Configuration** utility available in the Luna Control Center. Additionally, steps to do so vary, based on the access type of the account (i.e., Secure -- SSH, vs. Non-secure -- FTP).

Phase 1: Creating an Upload Account in a Desired Storage Group

A Storage Group is an allocated space within NetStorage in which you maintain your content, and an Upload Account is used to regulate access to that Storage Group. Access to the API, as well as relevant credential information are maintained within an Upload Account.

If you do not already have an Upload Account, you must create one, and then later edit it to enable the API (as discussed in the sections that follow). Instructions on creating a new Upload Account can be found in the *NetStorage - Configuration Guide (for FileStore)* -- specifically the section, *"Managing Upload Accounts"* (a link to this document is maintained on page 6).

If you already have an Upload Account setup, you can use it as well. You simply edit it to enable access to the API, as discussed in the sections that follow.

Phase 2: Enabling the API/Gathering Information

This is accomplished via the **NetStorage:Configuration** utility available in the Luna Control Center. The actual process varies, based on the access type in place for the Upload Account (i.e., **Secure -- SSH**, vs. **Non-secure -- FTP**).

Enabling for a Secure (SSH) Upload Account

1. Launch your desired browser and access the Luna Control Center:

   https://control.akamai.com

2. Login using the **User ID** and **Password** that were configured for access to Net-Storage.
3. Select the appropriate account -- one configured for NetStorage use -- from the **Select Account** drop-down.

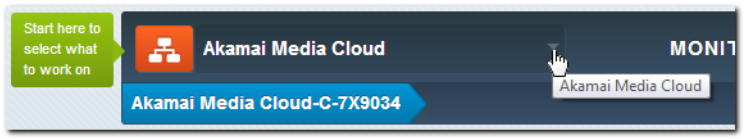
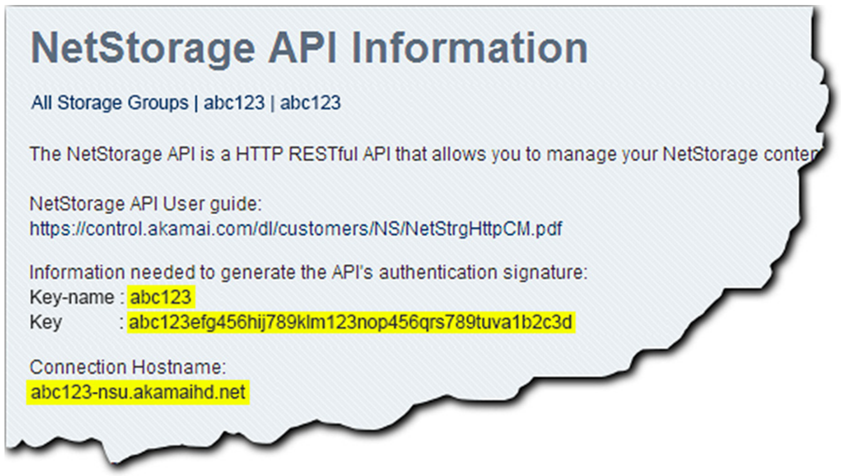

Start here to select what to work on
Akamai Media Cloud
MONIT
Akamai Media Cloud
Akamai Media Cloud-C-7X9034

Figure 1-1. The Select Account drop-down in Luna

4. If applicable, select the appropriate context from the **Context Selector** drop-down (i.e. the orange button accompanied by a comment bubble that prompts you to **"Start here to select what to work on"**).
5. Click the **Configure** mega-menu button and click the **"Configuration"** link beneath the **"NetStorage"** heading.
6. Locate the Storage Group housing the secure Upload Account to be used, mouse-over its **ACTION** drop-down and select **"View Details"**.
7. From the **UPLOAD ACCOUNTS** table, locate the applicable account, mouse-over its **ACTION** drop-down and select **"Edit"**.
8. In the **Account Details** section, click the **Enable NetStorage API** link, and click **OK** to confirm.
9. Click the newly revealed **NetStorage API Information** link (at right).
10. Various required information will be displayed. Make note of the following for use with the API:
    - **Key-name** - The actual "name" of the key (used in authenticating an API call)
    - **Key** - The "shared secret" value associated with the **Key-name** (used in authenticating an API call)
    - **Connection Hostname** - The "HOST" used in an API call



NetStorage API Information

All Storage Groups | abc123 | abc123

The NetStorage API is a HTTP RESTful API that allows you to manage your NetStorage conte

NetStorage API User guide:
https://control.akamai.com/dl/customers/NS/NetStrgHttpCM.pdf

Information needed to generate the API's authentication signature:
Key-name : abc123
Key : abc123efg456hij789klm123nop456qrs789tuva1b2c3d

Connection Hostname:
abc123-nsu.akamaihd.net

Figure 1-2. Gathering "Key" values from the NetStorage API Information interface in Luna

Enabling for a Non-Secure (FTP) Upload Account

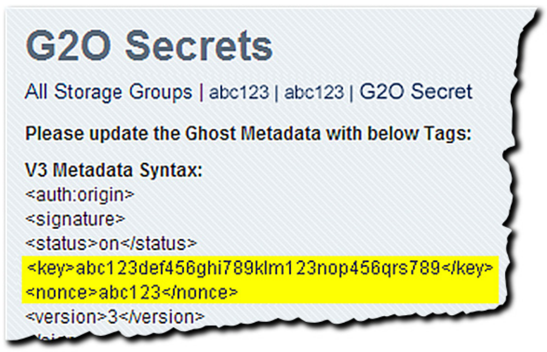
Step 1: Enable HTTP Upload Authentication (G2O)

This is the security used that will allow you to access the API. Contact your Akamai Representative for assistance in enabling it for use with the Upload Account that will be used in conjunction with the API.

Step 2: Information Gathering

Follow the steps below in order to obtain information required by the API.

1. Perform **Steps 1.** - **5.** in *Enabling for a Secure (SSH) Upload Account* on page 8.
2. Locate the Storage Group housing the non-secure Upload Account to be used, mouse-over its **ACTION** drop-down and select **"View Details"**
3. From the **UPLOAD ACCOUNTS** table, locate the applicable account, mouse-over its **ACTION** drop-down and select **"Edit"**.
4. Scroll to the bottom of the **Account Details** section, and ensure that HTTP **Upload Authentication (G2O)** is "Enabled" (see *Step 1: Enable HTTP Upload Authentication (G2O)*, above if it is not).
5. Click the **"View/Rotate G2O"** link.
6. In the **G2O Secrets** interface, make note of the following:
   - **Key ("<key>")** - The "shared secret" value associated with the **Key-name** (used in authenticating an API call).
   - **Key-name ("<nonce>")** - The actual "name" of the key, which is used in authenticating an API call.



G2O Secrets

All Storage Groups | abc123 | abc123 | G2O Secret

Please update the Ghost Metadata with below Tags:

V3 Metadata Syntax:

```
<auth:origin>
<signature>
<status>on</status>
<key>abc123def456ghi789klm123nop456qrs789</key>
<nonce>abc123</nonce>
<version>3</version>
```

Figure 1-3. Gathering the Key ("<key>") and the Key-name ("<nonce>") for use with a Non-secure Upload Account and the API

7. Generate your **"Connection Hostname"**. This is accomplished by combining the **Key-name** value with a specific URI (**"-nsu.akamaihd.net"**). For example, if your **Key-name** was **"abc123"**, your Connection Hostname would be:

   abc123-nsu.akamaihd.net

Phase 3: Upload Account Propagation

Any changes made to an Upload Account -- including enabling various functionality called out in the previous processes -- must be propagated to the NetStorage network before the account can be used with the API. The **CONTACTS** established for the account will be alerted via e-mail once the it has been updated and it is ready for use (This can take upwards of 90 minutes).

You can verify the current status of an Upload Account in the Luna Control Center.

Understanding API Basics - Important

Variable Entries and this API ("[ ]")

Throughout this chapter, references are made to "variables". This applies to mutable values that exist within the API call or the response output. Values shown in square brackets **("[ ]")** represent a variable that requires that you input a desired/specific value, or it indicates variable data may be displayed in response output. When defining a variable value in a call, *do not* include these characters as part of a value entry (i.e., unless noted to do otherwise).

The "HTTP/1.1" Suffix

The HTTP version in use is required in the syntax of a call, prefaced by a whitespace (i.e., as outlined in the Hypertext Transfer Protocol specification set by w3.org -- http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html -- see Section 5.1.2). This is currently "1.1". For example:

```
PUT /[path]/[file.ext] HTTP/1.1
```

Most languages/tools will offer automatic support for the inclusion of this value. However, you must ensure that this component is included in the API call.

Defining the HOST in a Call

Each API call requires that you define the HOST:

```
Host: [example]-nsu.akamaihd.net
```

This full value is your unique **"Connection Hostname"**. For details on obtaining this value, please see the section relevant to your Upload Account's access type:

- *Enabling for a Secure (SSH) Upload Account* on page 8, or
- *Enabling for a Non-Secure (FTP) Upload Account* on page 9

About the “Action” Header

This API uses a unified “action” HTTP header that specifies which action to perform on the file or directory name specified in the request URL. This header may include a variety of “fields” and acceptable HTTP methods include GET, POST, or PUT depending on the operation.

The Edge network can proxy all of the different methods currently used by various clients (e.g., GET, PUT, POST, DELETE, MOVE, MKCOL). To keep the API simple, the “action” header is authoritative. If you want to use action-specific HTTP methods, the Edge network can convert methods like DELETE and others to POST and include an “action” header.

Action Header “Version” and Authentication

The header itself includes a version number and any other arguments the operation requires. The version -- which at present is always **“1”** -- identifies the protocol, which eliminates backward compatibility concerns. Any request with the “action” HTTP header is treated as an API request and is authenticated with a shared key authentication scheme. If this authentication fails, the request is denied with an HTTP 403 error.

Action Header “Field” Syntax - Query String Encoding

Action headers can contain required and optional “fields”. Each field must be prefaced with an ampersand (“&”) -- including the “action” itself -- and its content must be input in standard “query string encoding” (URL encoding). For example, if the upload action also included a field entitled “target” which requires the variable, “/files/new file-1.txt”, it would be input as follows:

&action=upload&target=%2Ffiles%2Fnew%20file-1.txt

TIP: *Query string encoding also supports the use of a plus sign (“ + ”) to designate a whitespace.*

The PUT/POST Body Component

All APIs that require either a PUT or POST method must include a request message body. This body is delimited using the Content-Length: header or a Transfer-Encoding: chunked header. Most of these PUT/POST APIs do not specify any request body content, so an empty body should be sent using either of the following header formats:

- **"Content-Length: 0"**
- **"Transfer-Encoding: chunked"** (followed by an empty chunk-encoded body)

IMPORTANT: *This applies to all PUT/POST requests, except when incorporating the “upload” action. Specific usage requirements apply, and are discussed on page 24.*

Request Handling

The following numbered steps detail the flow of a request and how it is handled for an API call. Where **"Requirements"** are listed, these display activities that you must perform in the call itself, or when configuring the applicable header (prior to the call).

1. The **X-Akamai-ACS-Action** header is located.
   - **If the Header is Missing** - Request is not an API request. It may, however, still be a valid request that can be served.
   - **If the Header is Present** - The request is handled by the API (i.e., as either a valid operation or an error).
2. The **X-Akamai-ACS-Action** header is parsed using standard "query string" parsing rules.

NOTE: *Query string syntax already supports escaping arbitrary characters. Use care to escape at least spaces (" "), equal signs ("="), plus signs ("+"), and ampersands ("&") in any values you generate.*

3. The version is verified (i.e., the header must contain a "version" action header field, the value of which must be **"1"**, otherwise the request will fail).
   - **Requirements**:
     - Any legacy headers are denied. Unexpected **X-Akamai-*** headers may result in the request being denied to avoid potentially incorrect or ambiguous behavior from other APIs that may be, or may have been supported by the recipient servers. Below is an example of the appropriate syntax:

```
PUT /[CPCode]/[path]/[file.ext] HTTP/1.1
Host: [example]-nsu.akamaihd.net
X-Akamai-ACS-Action: version=1&action=upload&md5=0123456789abcdef01234
    56789abcdef&mtime=1260000000
[Signature headers (as described on page 14)]

[PUT body]
```

4. The request URL is checked for validity according to normal NetStorage rules.
   - **Requirements**:
     - This URL should begin with the same "`[CPCode]`" that you use with other NetStorage protocols. The URL will name the file or directory that will be queried or modified in this request.
5. The header field **"action"** is reviewed to determine the appropriate operation.
   - **Requirements**:
     - **An "Action" Must be Defined** - The following actions are supported for use (i.e., only these actions are valid; anything else will return an error):
       - **"Read-only" Actions**
         - **dir** - Return the directory structure (Usage details on page 17)

        - **download** - Download the file (Usage details on page 18)
        - **du** - Return the disk usage information for a directory (Usage details page 18)
        - **stat** - Return the stat structure (Usage details on page 22)
    - **"Update" Actions**
        - **delete** - Delete a file or symbolic link (Usage details on page 17)
        - **mkdir** - Create a directory (Usage details on page 19)
        - **mtime** - Change a file's mtime ("touch" - Usage details on page 19)
        - **quick-delete** - Quick-delete a directory (i.e., recursively delete a directory tree - Usage details on page 20)
        - **rename** - Rename a file or symbolic link (Usage details on page 20)
        - **rmdir** - Delete a directory (Usage details on page 21)
        - **symlink** - Create a symbolic link (Usage details on page 23)
        - **upload** - Upload a file or form (depending on user configuration - (Usage details on page 23)
- **Method Requirements** - **"Read-only"** actions must use method **GET**, and **"Update"** actions must use method **POST** or **PUT**, which are treated interchangeably. Any unexpected methods will generate an error.

6. The validity of the Signature Header Authentication in the **X-Akamai-ACS-Auth-Data/X-Akamai-ACS-Auth-Sign** headers is verified.
7. The defined action is processed.

Signature Header Authentication

All requests are authenticated using two special headers added by the client and checked by the NetStorage system. One of these headers contains basic authentication information about the specific request, and the other contains similar information encrypted with a shared secret (your Upload Account **"Key"**). This allows the origin server to perform several levels of authentication to ensure that the request is coming to NetStorage from an authorized client, without tampering.

Authentication Header Break-down

The two authentication headers follow the formats shown below:

```
X-Akamai-ACS-Auth-Data: [version], [0.0.0.0], [0.0.0.0], [time],
    [unique-id], [Key-name]
X-Akamai-ACS-Auth-Sign: [base-64-encoded-signature]
```

The information required for each variable value is described in the following table:

Field	Description
`[version]`	Indicates the authentication encryption format.
`[0.0.0.0]`	Reserved fields. Specify as **0.0.0.0**.
`[time]` 1)	The current epoch time (e.g., 1261440191 for Monday, December 21, 16:03:11, 2009).
`[unique-id]`	A unique ID chosen by the client with some randomness that will guarantee uniqueness for multiple headers generated at the same time for multiple requests.
`[Key-name]`	A simple internally-generated string that tells the server which key to use to authenticate the request. This makes it possible to transition from one key to another; during the transition the origin server will need to support more than one key. The Key-name must not contain special characters such as commas and spaces. For details on obtaining this value, please see the section relevant to your Upload Account's access type: • *Enabling for a Secure (SSH) Upload Account* on page 8, or • *Enabling for a Non-Secure (FTP) Upload Account* on page 9
`[base-64-encoded-signature]`	The base-64 encoded signature for the X-Akamai-ACS-Auth-Sign header is generated based on a selected algorithm. Please see the section, *The "base-64-encoded-signature"* on page 15 for complete details on this value.

1) The time on a client using the API must be within one (1) minute of the actual time. If the time varies more than this, then authentication will fail with an "HTTP 403" error. It is suggested that any client using the API run "ntp" (Network Time Protocol) to properly establish time.

The "base-64-encoded-signature"

Basic Make-Up

The base-64 encoded signature for the **X-Akamai-ACS-Auth-Sign** header is generated based on a specific algorithm (i.e., based on the **"[version]"** value specified in the **X-Akamai-ACS-Auth-Data** header). At present, versions "3" through "5" are supported by this API, though version "5" is preferred (version "4" is supported, but not preferred, and version "3" is deprecated and *should not be used*). Please refer to Knowledge Base below, to verify supported versions:

https://control.akamai.com/portal/kb/kbSearchDetails.jsf?articleId=5782

Depending on the version in use, the basic structure of the base-64-encoded-signature algorithm is as follows:

- **Version 3 - HMAC-MD5([key], [data] + [sign-string])**
- **Version 4 - HMAC-SHA1([key], [data] + [sign-string])**
- **Version 5 - HMAC-SHA256([key], [data] + [sign-string])**

In this algorithm, the following variable values are applied:

- **"[key]"** - The internally-generated **"Key"**. For details on obtaining this value, please see the section relevant to your Upload Account's access type:
  - *Enabling for a Secure (SSH) Upload Account* on page 8, or
  - *Enabling for a Non-Secure (FTP) Upload Account* on page 9.
- **"[data]"** - The full contents of the **X-Akamai-ACS-Auth-Data** header.
- **"[sign-string]"** - The string to sign is equal to the example below (full details on this value can be found in the next section -- *The "sign-string"*).

```
URL + "\n" + "x-akamai-acs-action:" + X-Akamai-ACS-Action value + "\n"
```

The "sign-string"

In the context of the API, the **"[sign-string]"** is formed by using:

1. The URL from the HTTP request line (the first line of the request) *without* the surrounding method (e.g., PUT, GET), protocol, or space character delimiters.
2. A single line feed character (represented here with a newline -- **"\n"** -- character)
3. The string "**x-akamai-acs-action:**" followed by the value of the **X-Akamai-ACS-Action** header (with any preceding and trailing whitespace *removed*)
4. A final line feed character (**"\n"**)

For example, with a request like this:

```
PUT /[path]/[file.ext] HTTP/1.1
Host: [example]-nsu.akamaihd.net
X-Akamai-ACS-Action: version=1&action=upload&md5=0123456789abcdef01234
56789abcdef&mtime=1260000000
```

The resulting sign-string would be as follows (i.e., where **"\n"** represents the line feed character (ASCII code 10)):

```
/[path]/[file.ext]\nx-akamai-acs-action:version=1&action=upload
    &md5=0123456789abcdef0123456789abcdef&mtime=1260000000\n
```

Example Authentication Headers

The X-Akamai-ACS-Auth-Data Header

Using the recommended "**`[version]`**" of **"5"**, if a request is generated at epoch time **1280000000** (i.e., the "**`[time]`**" variable) with a "**`[unique-id]`**" value of **"382644692"**, and a "**`[Key-name]`**" of **"key1"**, this authentication header would appear as follows:

```
X-Akamai-ACS-Auth-Data: 5, 0.0.0.0, 0.0.0.0, 1280000000, 382644692, key1
```

The X-Akamai-ACS-Auth-Sign Header

With the values set in the X-Akamai-ACS-Auth-Data Header, for the sake of example, assume that the "**`[key]`**" ("shared secret") value is **"abcdefghij"**, and the URL from the HTTP request for an upload action is "**`/[path]/[file.ext]`**". In addition, various optional upload action header fields are defined, and they too, must be included. As a result, the **HMAC-SHA256** would then be as follows:

```
HMAC-SHA256("abcdefghij",
    "5, 0.0.0.0, 0.0.0.0, 1280000000, 382644692, key1" +
    "/[path]/[file.ext]\n" +
    "x-akamai-acs-action:version=1&action=upload" +
    "&md5=0123456789abcdef0123456789abcdef" +
    "&mtime=1260000000\n")
```

NOTE: *In this particular example, there are exactly two newline and five space characters in the string that is signed using HMAC-SHA256*

Finally, the resulting X-Akamai-ACS-Auth-Sign header would look as follows:

```
X-Akamai-ACS-Auth-Sign: Ix98xZYkwygidinpmtKVk9+xPNn5QjozWDMROLjVWSo=
```

Both Headers in an Example Request

With both headers established, an example call incorporating them might look as follows (i.e., using an action of upload in this example -- along with its associated, optional action header fields):

```
PUT /dir1/dir2/file.html HTTP/1.1
Host: example-nsu.akamaihd.net
X-Akamai-ACS-Action: version=1&action=upload&md5=0123456789abcdef0123
    456789abcdef&mtime=1260000000
X-Akamai-ACS-Auth-Data: 5, 0.0.0.0, 0.0.0.0, 1280000000, 382644692, key1
X-Akamai-ACS-Auth-Sign: vuCWPzdEW5OUlH1rLfHokWAZAWSdaGTM8yX3bgIDWtA=

[Other PUT headers, if applicable]

[PUT body]
```

When the NetStorage FileStore server receives a request, it can use the information in the request to check the following:

- Both of the above specified headers exist.

- The version given in **X-Akamai-ACS-Auth-Data** is a supported version.
- It has the key corresponding to the "`[Key-name]`" given in **X-Akamai-ACS-Auth-Data**.
- The "`[time]`" given in **X-Akamai-ACS-Auth-Data** is within +/- 30 seconds of the current time.
- The NetStorage server *may* check that the given **X-Akamai-ACS-Auth-Data** header has not been used before.
- The signature matches the given **X-Akamai-ACS-Auth-Data** header, and the "sign-string" in the The **X-Akamai-ACS-Auth-Sign** header

If any of the above steps fail, the recipient server should reject the request.

TIP: *Your Account Representative can provide sample code to assist in implementing these steps.*

API Actions

All operations will be applied to the file or directory name specified in the request URL. Actions are listed in this section in alphabetic order.

NOTE: *If multiple paths are needed for an operation, the additional pathnames need to be specified via the "X-Akamai-ACS-Action" header.*

The "delete" Action

Include this action to delete an individual file or symbolic link.

Request Example

```
PUT /[CPCode]/[path]/[file.ext] HTTP/1.1
Host: [example]-nsu.akamaihd.net
X-Akamai-ACS-Action: version=1&action=delete
Content-Length: 0 or Transfer-Encoding: chunked (see page 11)
[Signature Headers]
```

Method Variables

Along with the "**`PUT/POST`**" method, specify the complete "**`[path]`**" to, and specify the target file/symbolic link to be deleted as the "**`[file.ext]`**" variable.

NOTE: *If the specified object does not exist, an HTTP 404 Not Found error may be returned.*

The "dir" Action

Use this action to return the structure for a selected directory.

Request Example

```
GET /[CPCode]/[Path] HTTP/1.1
Host: example-nsu.akamaihd.net
X-Akamai-ACS-Action: version=1&action=dir&format=xml
[Signature headers]
```

Method Variables

Along with the "**`GET`**" method, specify the appropriate "**`[CPCode]`**" (root) and the complete "**`[path]`**" to the target directory.

NOTE: *This action must be used to target a directory. If a file is included in the* "**`[path]`**"*, an error will be revealed (i.e., an "HTTP 412" will be displayed). The "stat" action is used to display file information (see page 22).*

Response Example

```
<stat directory="/dir1/dir2"]
    <file type="file" name="file.html" mtime="1260000000" size="1234567"
        md5="0123456789abcdef0123456789abcdef" />
    <file type="symlink" name="symlink.html" mtime="1260000000"
        target="file.html" />
    <file type="dir" name="dir3" mtime="1260000000" />
</stat>
```

NOTE: *The result of this request is returned as a standard XML document, but will contain one* **`[file]`** *tag per entry in the directory.*

The "download" Action

Include this action to download the specified file.

NOTE: *Download operations are allowed for files up to 1.8 GB in size. Larger files would need to be downloaded through the CDN configuration.*

Request Example

```
GET /[CPCode]/[path]/[file.ext] HTTP/1.1
Host: [example]-nsu.akamaihd.net
X-Akamai-ACS-Action: version=1&action=download
[Signature headers]
```

Method Variables

Along with the "**`GET`**" method, specify the appropriate "**`[CPCode]`**" (root) and any applicable sub-directories as the "**`[path]`**". End with the target "**`[file.ext]`**".

NOTE: *If the specified object does not exist, an HTTP 404 "Not Found" is returned.*

The "du" Action

Include this action to return disk usage information for the directory specified by the URL, including all files stored in any sub-directories that may exist.

Request Example

```
GET /[CPCode]/[path] HTTP/1.1
Host: [example]-nsu.akamaihd.net
X-Akamai-ACS-Action: version=1&action=du&format=xml
[Signature headers]
```

Method Variables

Along with the "**`GET`**" method, specify the appropriate "**`[CPCode]`**" (root) and the complete "**`[path]`**" to the target directory

Response Example

The result of this request is returned as XML in the following format:

```
<du directory="/dir1/dir2">
    <du-info files="12399999" bytes="383838383838">
</du>
```

The "mkdir" Action

Include this action to create a new directory.

Request Example

```
PUT /[CPCode]/[path]/[new_directory] HTTP/1.1
Host: example-nsu.akamaihd.net
X-Akamai-ACS-Action: version=1&action=mkdir
Content-Length: 0 or Transfer-Encoding: chunked (see page 11)
[Signature Headers]
```

Method Variables

Along with the "**`PUT/POST`**" method, input the applicable "**`[CPCode]`**" (root) as well as the "**`[path]`**" to the desired new directory location (if applicable). End with the "**`[directory]`**" variable set as the name for the new directory.

IMPORTANT: *A sub-directory and file in the same directory cannot have the same name (i.e., regardless of the file type/extension). For example, a single directory cannot contain a sub-directory named "baseball" as well as a file entitled "baseball.mp4". When creating a new directory, ensure that the name used does not already apply to a file that exists in the same location.*

The "mtime" Action

Incorporate this action to change a file's modification time ("touch").

Request Example

```
POST /[CPCode]/[path]/[file.ext] HTTP/1.1
Host: example-nsu.akamaihd.net
X-Akamai-ACS-Action: version=1&action=mtime&mtime=1260000000
Content-Length: 0 or Transfer-Encoding: chunked (see page 11)
[Signature Headers]
```

Method Variables

Along with the "**`PUT/POST`**" method, define the applicable "**`[CPCode]`**" (root) and define the complete "**`[path]`**" (if applicable). End with the target "**`[file.ext]`**".

Required Action Header Field

This action has a single required action header field:

- **"mtime=[#]" -** Set the "[#]" variable as the desired modification time for the target content (i.e., in UNIX epoch time).

The "quick-delete" Action

Used to perform a "quick-delete" of a selected directory (including all of its contents).

Request Example

```
POST /[CPCode]/[path]/[directory] HTTP/1.1
Host: example-nsu.akamaihd.net
version=1&action=quick-delete&quick-delete=imreallyreallysure
Content-Length: 0 or Transfer-Encoding: chunked (see page 11)
[Signature Headers]
```

Method Variables

Along with the "**`PUT/POST`**" method, define the applicable "**`[CPCode]`**" (root) followed by the complete "**`[path]`**" (if applicable). End with the name of the target directory as the "**`[directory]`**" variable.

Required Action Header Field

A single action header field is required:

- **"quick-delete=imreallyreallysure"** - This action header field *must be set to exactly this 18-character string* to minimize the chances of an inadvertent deletion.

NOTE: *The "quick-delete" action is disabled by default for security reasons, as it allows recursive removal of non-empty directory structures in a matter of seconds. If you wish to enable this feature, please contact your Account Representative with the NetStorage CPCode(s) for which you wish to use this feature.*

The "rename" Action

Include this action to rename a file or symbolic link. Request Example

```
POST /[CPCode]/[path]/[file.ext] HTTP/1.1
Host: example-nsu.akamaihd.net
X-Akamai-ACS-Action: version=1&action=destina-
tion=%2F12345%2F[path]%2F[new_file.ext]
Content-Length: 0 or Transfer-Encoding: chunked (see page 11)
[Signature Headers]
```

Method Variables

Along with the "**`PUT/POST`**" method, define the applicable "**`[CPCode]`**" (root), and input the complete "**`[path]`**" (if applicable). End with the filename of the file to be renamed as the "**`[file.ext]`**" variable.

Required Action Header Field

The action has one required action header field:

- **"destination=/[CPCode]/[path]/[file.ext]"** - Include the path to the target file/symlink to be renamed, as well as the new name for the object. Ensure that special characters (e.g., "/") are query string encoded, as noted earlier in this guide (see page 11).

IMPORTANT: *A sub-directory and file in the same directory cannot have the same name (i.e., regardless of the file type/extension). For example, a single directory cannot contain a sub-directory named "baseball" as well as a file entitled "baseball.mp4". When renaming a file/symlink, ensure that the name used does not already apply to a sub-directory that exists in the same location.*

NOTE: *A new sub-directory in the path -- one that exists off of the current "CPCode" root directory -- can be specified to also move the renamed object. However, you CANNOT specify a directory that exists off of a different CPCode root directory.*

The "rmdir" Action

Specify this action to delete an *empty* directory.

Request Example

```
POST /[path]/[target_directory] HTTP/1.1
Host: example-nsu.akamaihd.net
X-Akamai-ACS-Action: version=1&action=rmdir
Content-Length: 0 or Transfer-Encoding: chunked (see page 11)
[Signature Headers]
```

Method Variables

Within the "`PUT/POST`" method, specify the complete "`[path]`" to, and the "`[target_directory]`" name of the directory to be removed.

Non-empty Directories and the "rmdir" Action

Deleting a non-empty directory requires a combination of this action and the **`"delete"`** action, in order to empty the target of all contents:

1. **Remove All File Contents** - Use the **`"delete"`** action (see page 17) to remove all individual files/symbolic links that exist in the target directory.
2. **Remove All Sub-directories** - Any sub-directories that exist in the target directory must also be deleted -- this involves first accessing the sub-directory, and deleting all of its file contents via **`"delete"`** (as in **Step 1.** above), and then deleting the actual directory via **`"rmdir"`**.

TIP: *You can "recursively" delete a directory (i.e., delete it and all of its contents), by incorporating the* **`"quick-delete"`** *action (see page 20), but take careful note of the details for this action.*

The “stat” Action

Specify this action to return the stat structure of a named file/directory/symbolic link.

Request Example

```
GET /[path]/[target_object] HTTP/1.1
Host: [example]-nsu.akamaihd.net
X-Akamai-ACS-Action: version=1&action=stat&format=xml
[Signature headers]
```

Method Variables

Along with the “**`GET`**” method, specify the complete “**`[path]`**” to (if applicable), and the “**`[target_object]`**” name of the object to be targeted.

NOTE: *If the object does not exist, an “HTTP 404 Not Found” error is returned.*

Output Examples

The result of this request is returned as an XML document with the following formats (depending on the type of item targeted):

- **File Example**:

```
<stat directory="/dir1/dir2">
    <file type="file" name="file.html" mtime="1260000000" size="1234567"
        md5="0123456789abcdef0123456789abcdef" />
</stat>
```

- **Symbolic Link Example:**

```
<stat directory="/dir1/dir2">
    <file type="symlink" name="symlink.html" mtime="1260000000" tar
        get="file.html" />
</stat>
```

- **Directory Example:**

```
<stat directory="/dir1/dir2">
    <file type="dir" name="dir3" mtime="1260000000" />
</stat>
```

Be sure to support the standard XML quoting rules for attributes in which case a filename includes characters such as a quote (**"**) or an ampersand (**&**).

The “symlink” Action

Include this action to create a symbolic link

Request Example

```
POST /[path]/[file.ext] HTTP/1.1
Host: example-nsu.akamaihd.net
X-Akamai-ACS-Action: version=1&action=symlink&target=%2Fdir%2Flink.html
Content-Length: 0 or Transfer-Encoding: chunked (see page 11)
[Signature Headers]
```

Method Variables

Along with the “**PUT/POST**” method, specify the complete “**[path]**” (if applicable), and “**[file.ext]**” of the file to be the target of the symlink.

Required Action Header Field

This action has one required action header field:

- "**target=/[path]/[file.ext]**" - Include the complete path to the desired directory, as well as a desired file name (and extension) for the symlink. Ensure that special characters (e.g., “/”) are query string encoded, as noted earlier in this guide (see page 11).

IMPORTANT: *A sub-directory and symlink in the same directory cannot have the same name (i.e., regardless of the file type/extension). For example, a single directory cannot contain a sub-directory named “moved_new” as well as a symlink entitled “moved_new”. When creating a new symlink, ensure that the name used does not already apply to a sub-directory in the same location.*

The “upload” Action

Include this action to upload a file or form (depending on user configuration).

Request Example

```
PUT /[path]/[file.ext] HTTP/1.1
Host: [example]-nsu.akamaihd.net
X-Akamai-ACS-Action: version=1&action=upload
Content-Length:/Transfer-Encoding: (specific to this call - see page 24)
[Signature headers]
```

Method Variables

Within the “**PUT/POST**” method, define the complete “**[path]**” to the “**[file.ext]**” to be uploaded.

Optional Action Header Fields

Along with the standard “**upload**” action header content, the following optional action header fields can be included (i.e., appended to the end of the action header, separating each with an ampersand “&”).

- **index-zip=1** - Enable az2z processing to index uploaded “.zip” archive files for the “serve-from-zip” feature. See the *NetStorage - User’s Guide* for complete details on this feature (a link to this document is maintained on page 6).

- **mtime=[#]** - Set the "[#]" variable as the desired modification time for uploaded content using an applicable UNIX epoch time value. If not specified, the actual upload completion time will used instead.
- **size=[#]** - Enforce that the uploaded file is precisely "[#]" size (i.e., in bytes).
- **md5=[hash]** - Check and enforce that the md5sum is what is specified in this value. The hash must be specified in lowercase, hexadecimal notation. If this value is not specified, no content integrity checking will be performed.
- **sha1=[hash]** - Similar to the **md5** header, but uses the SHA-1 algorithm.
- **sha256=[hash]** - Similar to the **md5** header, but uses the SHA-256 algorithm.

NOTE: *If multiple hash verification headers are present, at least one MUST be enforced, but the implementation may choose to enforce multiple or all hashes present.*

- **upload-type=[type]** - Include this field to specify the upload type. The following are supported as the **"[type]"**:, and specify either of the following types:
  - **binary** - To specify a binary-encoded upload (e.g., **"&upload-type=binary"**)
  - **form** - To specify a form-encoded upload. In this case, one of two additional header fields should be included:
    - **field-name=[X]** - The upload is a multi-part form, and the content named in the field, **"[X]"** should be targeted for the upload (e.g., **"&upload-type=form&field-name=Upload"**).
    - **field-index=[#]** - The upload is a multi-part form, and the content specified in field number **"[#]"** should be targeted for the upload (e.g., **"&upload-type=form&field-name=1"**).

NOTE: *The "field-index" is zero-based (i.e., "0" would be set as the variable value for "field-index=[#]" to select the first field, "1" to select the second, etc.)*

NOTE: *If "upload-type=form" is specified, but neither the "field-name=[X]" nor "field-index=[#]" are specified, the API will try first to resolve those by looking up options in the appropriate data structure. If these values are not located, the API will default to the first form input field (index "0").*

NOTE: *If the "field-name=[X]" or "field-index=[#]" header fields are provided WITHOUT the preceding "upload-type=form" header field, they will be ignored.*

Applicable Usage Parameters

Various usage parameters apply, as illustrated in the points that follow:

- **Default Upload File Size Limitation** - Currently there is a default limit of 10 Gigabytes for a single file upload. Please contact your Account Representative if you have files that are larger than this limit, for help in uploading. This value is scheduled to be increased in the future.
- **Sub-directory/Filename Naming Conflicts** - A sub-directory and file in the same directory cannot have the same name (i.e., regardless of the file type/extension). For example, a single directory cannot contain a sub-directory named "baseball" as well as a file entitled "baseball.mp4". When uploading a new file,

ensure that the name used does not already apply to a sub-directory that currently exists in the same location.

- **Content-Length Header Usage** - As discussed in *The PUT/POST Body Component* on page 11, APIs that use the PUT method must include the Content-Length: header (i.e., or a Transfer-Encoding: chunked header). If you intend to include the Content-Length header, you must include the appropriate size when using the "upload" action. Such an HTTP request might look something like this:

```
PUT /[path]/[file.ext] HTTP/1.1
Host: [example]-nsu.akamaihd.net
X-Akamai-ACS-Action: version=1&action=upload&md5=atend&mtime=atend
Content-Length: [size in bytes]
[Signature headers]
```

- **Chunk-encoded PUT/POST Body Usage** - When using this format, the **"md5=[hash]"**, **"sha1=[hash]"**, **"sha265=[hash]"**, **"size=[#]"**, and/or **"mtime=[#]"** fields discussed above can also be specified as a "chunk trailer" after sending the file. To accomplish this:

  a. Specify the value **"atend"** in the original **X-Akamai-ACS-Action** header

  b. Send the body, and then re-send exactly the same **X-Akamai-ACS-Action** header at the end of the request, but with the actual values filled in.

  c. The two shared key authentication signature headers must then be recomputed and re-sent as chunk trailers as well.

  The authentication signature header timestamp enforcement is relaxed here to account for potential transfer buffering and/or transfer latencies: any time between the timestamp in the original request and the time this header is actually received plus 30 seconds will be accepted. Such an HTTP request might look something like this:

```
PUT /[path]/[file.ext] HTTP/1.1
Host: [example]-nsu.akamaihd.net
X-Akamai-ACS-Action: version=1&action=upload&md5=atend&mtime=atend
Transfer-Encoding: chunked
[Signature headers]
[Chunked PUT Body]
```

Appendix A: Cross-origin Resource Sharing (CORS) Support

The NetStorage HTTP API for FileStore supports the Cross-origin resource sharing (CORS) mechanism:

http://www.w3.org/TR/cors/

This mechanism allows JavaScript **"XMLHttpRequest"** requests from any site, in order to access NetStorage, provided that the necessary headers are sent.

Storage Access Key Handling

The underlying storage access key must always be maintained strictly in your control. As such, embedding your NetStorage key in, or transferring it to any client running on an un-trusted or third-party system is *not allowed*.

Here are some alternatives to embedding your key in a client:

- User enters your NetStorage key directly into a form (which assumes that the user of this script is one of your employees), or
- The script communicates with an authenticated external server to receive individual action and signature headers for each action to be performed by the script.

NOTE: *Securing data embedded within JavaScript programs is incredibly difficult, if not impossible, given the way browsers load and execute scripts. As such, we strongly discourage embedding the NetStorage secret in any JavaScript program or HTML page.*

Appendix B: Potential HTTP Error Codes

The table that follows lists the potential HTTP Error Codes that may be encountered when issuing calls via the NetStorage HTTP API.

Code	Description
400	**BAD REQUEST** (Can occur if the call was improperly formatted. Check the call structure and retry.)
401	**UNAUTHORIZED** (Proper authentication is required to process the request. This error code indicates that it was not properly given in the call. Please review the section on "Signature Header Authentication" (see page 13) and reformat the call as necessary.)
403	**FORBIDDEN** (The Akamai Edge server has denied access to the call. Please verify that the call is properly formatted and retry.)
404	**NOT FOUND** (Can occur if the targeted object can not be found in the specified path. Check the path provided and try again.)
406	**NOT ACCEPTABLE** (The response type being returned is not provided in the Accept-Type HTTP header in your Client request. See http://en.wikipedia.org/wiki/List_of_HTTP_header_fields for further details).
409	**CONFLICT** (Can occur if an attempt is made to create an object in which a non-directory element is included in a path (e.g., a file), when a targeted symlink points to another root, etc.)
412	**PRECONDITION FAILED** (The Akamai Edge Server that received the API call thinks that the call included a 'Precondition' specification which the server detected was not met. Check the format of the call and try again.)
422	**UNPROCESSABLE ENTITY** (Can occur when issues arise using the **"delete"** action -- the targeted object for delete can not be deleted (e.g., a non-empty directory).)
423	**LOCKED**
500	**INTERNAL SERVER ERROR** (Typically due to an Akamai internal error, Contact Akamai CCare.)
501	**NOT IMPLEMENTED** (Occurs when a server in the Akamai Edge network does not understand or will not support an HTTP method that it finds in an API call.)

Appendix C: Akamai Customer Support

Step 1: Open a Support Case

If you have a technical problem or question that is not covered in this documentation, or requires immediate attention, the quickest means of communicating it is to contact Akamai CCare by opening a support ticket:

1. Access and login to the **Luna Control Center**:

   https://control.akamai.com

2. Click the **Support** link in the top right corner of the interface.
3. In the **OPEN A SUPPORT CASE** pane, click the **Report a Technical Issue** link.
4. Follow the prompts revealed to submit your issue.

Step 2: Contact Akamai Customer Support

The table below offers contact information that can be used to contact an Akamai Representative for support.

NOTE: *Business support services are offered Monday - Friday, 7:30 AM - 8:30 PM.*

Operations Technical Support
Phone (Toll Free, 24/7):
• **1 (877) 4-AKATEC (252832)** - Click here for instructions on dialing toll-free outside the United States • **Contact your Akamai Account Manager**
E-mail
• ccare@akamai.com - For routine technical support issues • specialist@akami.com - For account assistance (resetting of passwords, setup of users, portal administration, etc.).