WebHDFS REST API

Table of contents

1 Document Conventions

`Monospaced`	Used for commands, HTTP request and responses and code blocks.
`<Monospaced>`	User entered values.
`[Monospaced]`	Optional values. When the value is not specified, the default value is used.
Italics	Important phrases and words.

2 Introduction

The HTTP REST API supports the complete FileSystem interface for HDFS. The operations and the corresponding FileSystem methods are shown in the next section. The Section HTTP Query Parameter Dictionary specifies the parameter details such as the defaults and the valid values.

2.1 Operations

- HTTP GET
  - `OPEN` (see FileSystem.open)
  - `GETFILESTATUS` (see FileSystem.getFileStatus)
  - `LISTSTATUS` (see FileSystem.listStatus)
  - `GETCONTENTSUMMARY` (see FileSystem.getContentSummary)
  - `GETFILECHECKSUM` (see FileSystem.getFileChecksum)
  - `GETHOMEDIRECTORY` (see FileSystem.getHomeDirectory)
  - `GETDELEGATIONTOKEN` (see FileSystem.getDelegationToken)
- HTTP PUT
  - `CREATE` (see FileSystem.create)
  - `MKDIRS` (see FileSystem.mkdirs)
  - `RENAME` (see FileSystem.rename)
  - `SETREPLICATION` (see FileSystem.setReplication)
  - `SETOWNER` (see FileSystem.setOwner)
  - `SETPERMISSION` (see FileSystem.setPermission)
  - `SETTIMES` (see FileSystem.setTimes)
  - `RENEWDELEGATIONTOKEN` (see DistributedFileSystem.renewDelegationToken)
  - `CANCELDELEGATIONTOKEN` (see DistributedFileSystem.cancelDelegationToken)
- HTTP POST
  - `APPEND` (see FileSystem.append)
  - `CONCAT` (see FileSystem.concat)
- HTTP DELETE

- `DELETE` (see FileSystem.delete)

2.2 FileSystem URIs vs HTTP URLs

The FileSystem scheme of WebHDFS is "`webhdfs://`". A WebHDFS FileSystem URI has the following format.

```
webhdfs://<HOST>:<HTTP_PORT>/<PATH>
```

The above WebHDFS URI corresponds to the below HDFS URI.

```
hdfs://<HOST>:<RPC_PORT>/<PATH>
```

In the REST API, the prefix "`/webhdfs/v1`" is inserted in the path and a query is appended at the end. Therefore, the corresponding HTTP URL has the following format.

```
http://<HOST>:<HTTP_PORT>/webhdfs/v1/<PATH>?op=...
```

2.3 HDFS Configuration Options

Below are the HDFS configuration options for WebHDFS.

Property Name	Description
`dfs.webhdfs.enabled`	Enable/disable WebHDFS in Namenodes and Datanodes
`dfs.web.authentication.kerberos.princ`	The HTTP Kerberos principal used by Hadoop-Auth in the HTTP endpoint. The HTTP Kerberos principal MUST start with 'HTTP/' per Kerberos HTTP SPNEGO specification.
`dfs.web.authentication.kerberos.keyta`	The Kerberos keytab file with the credentials for the HTTP Kerberos principal used by Hadoop-Auth in the HTTP endpoint.

3 Authentication

When security is *off*, the authenticated user is the username specified in the `user.name` query parameter. If the `user.name` parameter is not set, the server may either set the authenticated user to a default web user, if there is any, or return an error response.

When security is *on*, authentication is performed by either Hadoop delegation token or Kerberos SPNEGO. If a token is set in the `delegation` query parameter, the authenticated user is the user encoded in the token. If the `delegation` parameter is not set, the user is authenticated by Kerberos SPNEGO.

Below are examples using the `curl` command tool.

1. Authentication when security is off:

```
curl -i "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?[user.name=<USER>&]op=..."
```

2. Authentication using Kerberos SPNEGO when security is on:

```
curl -i --negotiate -u : "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?op=..."
```

3. Authentication using Hadoop delegation token when security is on:

```
curl -i "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?delegation=<TOKEN>&op=..."
```

4 Proxy Users

When the proxy user feature is enabled, a proxy user *P* may submit a request on behalf of another user *U*. The username of *U* must be specified in the `doas` query parameter unless a delegation token is presented in authentication. In such case, the information of both users *P* and *U* must be encoded in the delegation token.

1. A proxy request when security is off:

```
curl -i "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?[user.name=<USER>&]doas=<USER>&op=..."
```

2. A proxy request using Kerberos SPNEGO when security is on:

```
curl -i --negotiate -u : "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?doas=<USER>&op=..."
```

3. A proxy request using Hadoop delegation token when security is on:

```
curl -i "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?delegation=<TOKEN>&op=..."
```

5 File and Directory Operations

5.1 Create and Write to a File

- Step 1: Submit a HTTP PUT request without automatically following redirects and without sending the file data.

```
curl -i -X PUT "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?op=CREATE
                    [&overwrite=<true|false>][&blocksize=<LONG>][&replication=<SHORT>]
                    [&permission=<OCTAL>][&buffersize=<INT>]"
```

The request is redirected to a datanode where the file data is to be written:

```
HTTP/1.1 307 TEMPORARY_REDIRECT
Location: http://<DATANODE>:<PORT>/webhdfs/v1/<PATH>?op=CREATE...
Content-Length: 0
```

- Step 2: Submit another HTTP PUT request using the URL in the `Location` header with the file data to be written.

```
curl -i -X PUT -T <LOCAL_FILE> "http://<DATANODE>:<PORT>/webhdfs/v1/<PATH>?
op=CREATE..."
```

The client receives a `201 Created` response with zero content length and the WebHDFS URI of the file in the `Location` header:

```
HTTP/1.1 201 Created
Location: webhdfs://<HOST>:<PORT>/<PATH>
Content-Length: 0
```

Note that the reason of having two-step create/append is for preventing clients to send out data before the redirect. This issue is addressed by the "`Expect: 100-continue`" header in HTTP/1.1; see RFC 2616, Section 8.2.3. Unfortunately, there are software library bugs (e.g. Jetty 6 HTTP server and Java 6 HTTP client), which do not correctly implement "`Expect: 100-continue`". The two-step create/append is a temporary workaround for the software library bugs.

See also: `overwrite`, `blocksize`, `replication`, `permission`, `buffersize`, FileSystem.create

5.2 Append to a File

- Step 1: Submit a HTTP POST request without automatically following redirects and without sending the file data.

```
curl -i -X POST "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?op=APPEND[&buffersize=<INT>]"
```

The request is redirected to a datanode where the file data is to be appended:

```
HTTP/1.1 307 TEMPORARY_REDIRECT
```

```
Location: http://<DATANODE>:<PORT>/webhdfs/v1/<PATH>?op=APPEND...
Content-Length: 0
```

- Step 2: Submit another HTTP POST request using the URL in the `Location` header with the file data to be appended.

  ```
  curl -i -X POST -T <LOCAL_FILE> "http://<DATANODE>:<PORT>/webhdfs/v1/<PATH>?
  op=APPEND..."
  ```

  The client receives a response with zero content length:

  ```
  HTTP/1.1 200 OK
  Content-Length: 0
  ```

See the note in the previous section for the description of why this operation requires two steps.

See also: `buffersize`, FileSystem.append

5.3 Concatenate Files

- Submit a HTTP POST request.

  ```
          curl -i -X POST "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?
  op=CONCAT&sources=<PATHS>"
  ```

  The client receives a response with zero content length:

  ```
  HTTP/1.1 200 OK
  Content-Length: 0
  ```

See also: `sources`, FileSystem.concat

5.4 Open and Read a File

- Submit a HTTP GET request with automatically following redirects.

  ```
  curl -i -L "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?op=OPEN
                      [&offset=<LONG>][&length=<LONG>][&buffersize=<INT>]"
  ```

  The request is redirected to a datanode where the file data can be read:

  ```
  HTTP/1.1 307 TEMPORARY_REDIRECT
  Location: http://<DATANODE>:<PORT>/webhdfs/v1/<PATH>?op=OPEN...
  Content-Length: 0
  ```

The client follows the redirect to the datanode and receives the file data:

```
HTTP/1.1 200 OK
Content-Type: application/octet-stream
Content-Length: 22

Hello, webhdfs user!
```

See also: `offset`, `length`, `buffersize`, FileSystem.open

5.5 Make a Directory

- Submit a HTTP PUT request.

```
        curl -i -X PUT "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?
op=MKDIRS[&permission=<OCTAL>]"
```

The client receives a response with a `boolean` JSON object:

```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked

{"boolean": true}
```

See also: `permission`, FileSystem.mkdirs

5.6 Rename a File/Directory

- Submit a HTTP PUT request.

```
curl -i -X PUT "<HOST>:<PORT>/webhdfs/v1/<PATH>?op=RENAME&destination=<PATH>"
```

The client receives a response with a `boolean` JSON object:

```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked

{"boolean": true}
```

See also: `destination`, FileSystem.rename

5.7 Delete a File/Directory

- Submit a HTTP DELETE request.

```
curl -i -X DELETE "http://<host>:<port>/webhdfs/v1/<path>?op=DELETE
                              [&recursive=<true|false>]"
```

The client receives a response with a `boolean` JSON object:

```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked

{"boolean": true}
```

See also: `recursive`, FileSystem.delete

5.8 Status of a File/Directory

- Submit a HTTP GET request.

```
curl -i  "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?op=GETFILESTATUS"
```

The client receives a response with a `FileStatus` JSON object:

```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked

{
  "FileStatus":
  {
    "accessTime"      : 0,
    "blockSize"       : 0,
    "group"           : "supergroup",
    "length"          : 0,             //in bytes, zero for directories
    "modificationTime": 1320173277227,
    "owner"           : "webuser",
    "pathSuffix"      : "",
    "permission"      : "777",
    "replication"     : 0,
    "type"            : "DIRECTORY"    //enum {FILE, DIRECTORY}
  }
}
```

See also: FileSystem.getFileStatus

5.9 List a Directory

- Submit a HTTP GET request.

```
curl -i  "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?op=LISTSTATUS"
```

The client receives a response with a `FileStatuses` JSON object:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 427

{
  "FileStatuses":
  {
    "FileStatus":
    [
      {
        "accessTime"      : 1320171722771,
        "blockSize"       : 33554432,
        "group"           : "supergroup",
        "length"          : 24930,
        "modificationTime": 1320171722771,
        "owner"           : "webuser",
        "pathSuffix"      : "a.patch",
        "permission"      : "644",
        "replication"     : 1,
        "type"            : "FILE"
      },
      {
        "accessTime"      : 0,
        "blockSize"       : 0,
        "group"           : "supergroup",
        "length"          : 0,
        "modificationTime": 1320895981256,
        "owner"           : "szetszwo",
        "pathSuffix"      : "bar",
        "permission"      : "711",
        "replication"     : 0,
        "type"            : "DIRECTORY"
      },
      ...
    ]
  }
}
```

See also: FileSystem.listStatus

6 Other File System Operations

6.1 Get Content Summary of a Directory

- Submit a HTTP GET request.

```
curl -i "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?op=GETCONTENTSUMMARY"
```

The client receives a response with a `ContentSummary` JSON object:

```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked

{
  "ContentSummary":
  {
    "directoryCount": 2,
    "fileCount"     : 1,
    "length"        : 24930,
    "quota"         : -1,
    "spaceConsumed" : 24930,
    "spaceQuota"    : -1
  }
}
```

See also: FileSystem.getContentSummary

6.2 Get File Checksum

- Submit a HTTP GET request.

```
curl -i "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?op=GETFILECHECKSUM"
```

The request is redirected to a datanode:

```
HTTP/1.1 307 TEMPORARY_REDIRECT
Location: http://<DATANODE>:<PORT>/webhdfs/v1/<PATH>?op=GETFILECHECKSUM...
Content-Length: 0
```

The client follows the redirect to the datanode and receives a `FileChecksum` JSON object:

```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked

{
  "FileChecksum":
  {
    "algorithm": "MD5-of-1MD5-of-512CRC32",
    "bytes"    : "eadb10de24aa315748930df6e185c0d ...",
    "length"   : 28
  }
}
```

See also: FileSystem.getFileChecksum

6.3 Get Home Directory

- Submit a HTTP GET request.

```
curl -i "http://<HOST>:<PORT>/webhdfs/v1/?op=GETHOMEDIRECTORY"
```

  The client receives a response with a `Path` JSON object:

```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked

{"Path": "/user/szetszwo"}
```

See also: FileSystem.getHomeDirectory

6.4 Set Permission

- Submit a HTTP PUT request.

```
curl -i -X PUT "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?op=SETPERMISSION
                              [&permission=<OCTAL>]"
```

  The client receives a response with zero content length:

```
HTTP/1.1 200 OK
Content-Length: 0
```

See also: `permission`, FileSystem.setPermission

6.5 Set Owner

- Submit a HTTP PUT request.

```
curl -i -X PUT "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?op=SETOWNER
                              [&owner=<USER>][&group=<GROUP>]"
```

  The client receives a response with zero content length:

```
HTTP/1.1 200 OK
Content-Length: 0
```

See also: `owner`, `group`, FileSystem.setOwner

6.6 Set Replication Factor

- Submit a HTTP PUT request.

```
curl -i -X PUT "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?op=SETREPLICATION
                              [&replication=<SHORT>]"
```

The client receives a response with a `boolean` JSON object:

```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked

{"boolean": true}
```

See also: `replication`, FileSystem.setReplication

6.7 Set Access or Modification Time

- Submit a HTTP PUT request.

```
curl -i -X PUT "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?op=SETTIMES
                              [&modificationtime=<TIME>][&accesstime=<TIME>]"
```

The client receives a response with zero content length:

```
HTTP/1.1 200 OK
Content-Length: 0
```

See also: `modificationtime`, `accesstime`, FileSystem.setTimes

7 Delegation Token Operations

7.1 Get Delegation Token

- Submit a HTTP GET request.

```
curl -i "http://<HOST>:<PORT>/webhdfs/v1/?op=GETDELEGATIONTOKEN&renewer=<USER>"
```

The client receives a response with a `Token` JSON object:

```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked

{
  "Token":
  {
    "urlString": "JQAIaG9y..."
  }
}
```

See also: `renewer`, FileSystem.getDelegationToken

7.2 Renew Delegation Token

- Submit a HTTP PUT request.

```
curl -i -X PUT "http://<HOST>:<PORT>/webhdfs/v1/?op=RENEWDELEGATIONTOKEN&token=<TOKEN>"
```

The client receives a response with a `long` JSON object:

```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked

{"long": 1320962673997}           //the new expiration time
```

See also: `token`, DistributedFileSystem.renewDelegationToken

7.3 Cancel Delegation Token

- Submit a HTTP PUT request.

```
curl -i -X PUT "http://<HOST>:<PORT>/webhdfs/v1/?
op=CANCELDELEGATIONTOKEN&token=<TOKEN>"
```

The client receives a response with zero content length:

```
HTTP/1.1 200 OK
Content-Length: 0
```

See also: `token`, DistributedFileSystem.cancelDelegationToken

8 Error Responses

When an operation fails, the server may throw an exception. The JSON schema of error responses is defined in `RemoteException` JSON schema. The table below shows the mapping from exceptions to HTTP response codes.

8.1 HTTP Response Codes

Exceptions	HTTP Response Codes
`IllegalArgumentException`	`400 Bad Request`
`UnsupportedOperationException`	`400 Bad Request`
`SecurityException`	`401 Unauthorized`
`IOException`	`403 Forbidden`
`FileNotFoundException`	`404 Not Found`
`RumtimeException`	`500 Internal Server Error`

Below are examples of exception responses.

8.1.1 Illegal Argument Exception

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Transfer-Encoding: chunked

{
  "RemoteException":
  {
    "exception"    : "IllegalArgumentException",
    "javaClassName": "java.lang.IllegalArgumentException",
    "message"      : "Invalid value for webhdfs parameter \"permission\": ..."
  }
}
```

8.1.2 Security Exception

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json
Transfer-Encoding: chunked

{
  "RemoteException":
  {
    "exception"    : "SecurityException",
    "javaClassName": "java.lang.SecurityException",
    "message"      : "Failed to obtain user group information: ..."
  }
}
```

8.1.3 Access Control Exception

```
HTTP/1.1 403 Forbidden
Content-Type: application/json
Transfer-Encoding: chunked

{
  "RemoteException":
  {
    "exception"    : "AccessControlException",
    "javaClassName": "org.apache.hadoop.security.AccessControlException",
    "message"      : "Permission denied: ..."
  }
}
```

8.1.4 File Not Found Exception

```
HTTP/1.1 404 Not Found
Content-Type: application/json
Transfer-Encoding: chunked

{
  "RemoteException":
  {
    "exception"    : "FileNotFoundException",
    "javaClassName": "java.io.FileNotFoundException",
    "message"      : "File does not exist: /foo/a.patch"
  }
}
```

9 JSON Schemas

All operations, except for `OPEN`, either return a zero-length response or a JSON response . For `OPEN`, the response is an octet-stream. The JSON schemas are shown below. See draft-zyp-json-schema-03 for the syntax definitions of the JSON schemas.

9.1 Boolean JSON Schema

```
{
  "name"      : "boolean",
  "properties":
  {
    "boolean":
    {
      "description": "A boolean value",
      "type"       : "boolean",
      "required"   : true
    }
  }
}
```

See also: MKDIRS, RENAME, DELETE, SETREPLICATION

9.2 ContentSummary JSON Schema

```
{
  "name"      : "ContentSummary",
  "properties":
  {
    "ContentSummary":
    {
      "type"      : "object",
      "properties":
      {
        "directoryCount":
        {
          "description": "The number of directories.",
          "type"       : "integer",
          "required"   : true
        },
        "fileCount":
        {
          "description": "The number of files.",
          "type"       : "integer",
          "required"   : true
        },
        "length":
        {
          "description": "The number of bytes used by the content.",
          "type"       : "integer",
          "required"   : true
        },
        "quota":
        {
          "description": "The namespace quota of this directory.",
          "type"       : "integer",
          "required"   : true
        },
        "spaceConsumed":
        {
          "description": "The disk space consumed by the content.",
          "type"       : "integer",
          "required"   : true
        },
        "spaceQuota":
        {
          "description": "The disk space quota.",
          "type"       : "integer",
          "required"   : true
        }
      }
    }
  }
}
```

See also: GETCONTENTSUMMARY

9.3 FileChecksum JSON Schema

```
{
  "name"      : "FileChecksum",
  "properties":
  {
    "FileChecksum":
    {
      "type"      : "object",
      "properties":
      {
        "algorithm":
        {
          "description": "The name of the checksum algorithm.",
          "type"       : "string",
          "required"   : true
        },
        "bytes":
        {
          "description": "The byte sequence of the checksum in hexadecimal.",
          "type"       : "string",
          "required"   : true
        },
        "length":
        {
          "description": "The length of the bytes (not the length of the string).",
          "type"       : "integer",
          "required"   : true
        }
      }
    }
  }
}
```

See also: GETFILECHECKSUM

9.4 FileStatus JSON Schema

```
{
  "name"      : "FileStatus",
  "properties":
  {
    "FileStatus": fileStatusProperties      //See FileStatus Properties
  }
}
```

See also: GETFILESTATUS, FileStatus

9.4.1 FileStatus Properties

JavaScript syntax is used to define `fileStatusProperties` so that it can be referred in both `FileStatus` and `FileStatuses` JSON schemas.

```
var fileStatusProperties =
{
  "type"      : "object",
  "properties":
  {
    "accessTime":
    {
      "description": "The access time.",
      "type"       : "integer",
      "required"   : true
    },
    "blockSize":
    {
      "description": "The block size of a file.",
      "type"       : "integer",
      "required"   : true
    },
    "group":
    {
      "description": "The group owner.",
      "type"       : "string",
      "required"   : true
    },
    "length":
    {
      "description": "The number of bytes in a file.",
      "type"       : "integer",
      "required"   : true
    },
    "modificationTime":
    {
      "description": "The modification time.",
      "type"       : "integer",
      "required"   : true
    },
    "owner":
    {
      "description": "The user who is the owner.",
      "type"       : "string",
      "required"   : true
    },
    "pathSuffix":
    {
      "description": "The path suffix.",
      "type"       : "string",
      "required"   : true
    },
    "permission":
    {
      "description": "The permission represented as a octal string.",
      "type"       : "string",
      "required"   : true
    },
    "replication":
    {
      "description": "The number of replication of a file.",
      "type"       : "integer",
      "required"   : true
    },
   "type":
    {
      "description": "The type of the path object.",
      "enum"       : ["FILE", "DIRECTORY"],
      "required"   : true
    }
  }
};
```

9.5 FileStatuses JSON Schema

A `FileStatuses` JSON object represents an array of `FileStatus` JSON objects.

```

{
  "name"      : "FileStatuses",
  "properties":
  {
    "FileStatuses":
    {
      "type"      : "object",
      "properties":
      {
        "FileStatus":
        {
          "description": "An array of FileStatus",
          "type"       : "array",
          "items"      : fileStatusProperties      //See FileStatus Properties
        }
      }
    }
  }
}
```

See also: `LISTSTATUS`, FileStatus

9.6 Long JSON Schema

```

{
  "name"      : "long",
  "properties":
  {
    "long":
    {
      "description": "A long integer value",
      "type"       : "integer",
      "required"   : true
    }
  }
}
```

See also: RENEWDELEGATIONTOKEN,

9.7 Path JSON Schema

```
{
  "name"      : "Path",
  "properties":
  {
    "Path":
    {
      "description": "The string representation a Path.",
      "type"       : "string",
      "required"   : true
    }
  }
}
```

See also: GETHOMEDIRECTORY, Path

9.8 RemoteException JSON Schema

```
{
  "name"      : "RemoteException",
  "properties":
  {
    "RemoteException":
    {
      "type"      : "object",
      "properties":
      {
        "exception":
        {
          "description": "Name of the exception",
          "type"       : "string",
          "required"   : true
        },
        "message":
        {
          "description": "Exception message",
          "type"       : "string",
          "required"   : true
        },
        "javaClassName":                                     //an optional property
        {
          "description": "Java class name of the exception",
          "type"       : "string",
        }
      }
    }
  }
}
```

9.9 Token JSON Schema

```
{
  "name"      : "Token",
  "properties":
  {
    "Token":
    {
      "type"      : "object",
      "properties":
      {
        "urlString":
        {
          "description": "A delegation token encoded as a URL safe string.",
          "type"       : "string",
          "required"   : true
        }
      }
    }
  }
}
```

See also: `GETDELEGATIONTOKEN`, the note in Delegation.

10 HTTP Query Parameter Dictionary

10.1 Access Time

Name	`accesstime`
Description	The access time of a file/directory.
Type	long
Default Value	-1 (means keeping it unchanged)
Valid Values	-1 or a timestamp
Syntax	Any integer.

See also: `SETTIMES`

10.2 Block Size

Name	`blocksize`
Description	The block size of a file.
Type	long
Default Value	Specified in the configuration.

Valid Values	> 0
Syntax	Any integer.

See also: `CREATE`

10.3 Buffer Size

Name	`buffersize`
Description	The size of the buffer used in transferring data.
Type	int
Default Value	Specified in the configuration.
Valid Values	> 0
Syntax	Any integer.

See also: `CREATE`, `APPEND`, `OPEN`

10.4 Delegation

Name	`delegation`
Description	The delegation token used for authentication.
Type	String
Default Value	<empty>
Valid Values	An encoded token.
Syntax	See the note below.

Note that delegation tokens are encoded as a URL safe string; see `encodeToUrlString()` and `decodeFromUrlString(String)` in `org.apache.hadoop.security.token.Token` for the details of the encoding.

See also: Authentication

10.5 Destination

Name	`destination`
Description	The destination path used in RENAME.
Type	Path

Default Value	<empty> (an invalid path)
Valid Values	An absolute FileSystem path without scheme and authority.
Syntax	Any path.

See also: `RENAME`

10.6 Do As

Name	`doas`
Description	Allowing a proxy user to do as another user.
Type	String
Default Value	null
Valid Values	Any valid username.
Syntax	Any string.

See also: Proxy Users

10.7 Group

Name	`group`
Description	The name of a group.
Type	String
Default Value	<empty> (means keeping it unchanged)
Valid Values	Any valid group name.
Syntax	Any string.

See also: `SETOWNER`

10.8 Length

Name	`length`
Description	The number of bytes to be processed.
Type	long
Default Value	null (means the entire file)

Valid Values	>= 0 or null
Syntax	Any integer.

See also: `OPEN`

10.9 Modification Time

Name	`modificationtime`
Description	The modification time of a file/directory.
Type	long
Default Value	-1 (means keeping it unchanged)
Valid Values	-1 or a timestamp
Syntax	Any integer.

See also: `SETTIMES`

10.10 Offset

Name	`offset`
Description	The starting byte position.
Type	long
Default Value	0
Valid Values	>= 0
Syntax	Any integer.

See also: `OPEN`

10.11 Op

Name	`op`
Description	The name of the operation to be executed.
Type	enum
Default Value	null (an invalid value)
Valid Values	Any valid operation name.
Syntax	Any string.

See also: Operations

10.12 Overwrite

Name	`overwrite`
Description	If a file already exists, should it be overwritten?
Type	boolean
Default Value	false
Valid Values	true | false
Syntax	true | false

See also: `CREATE`

10.13 Owner

Name	`owner`
Description	The username who is the owner of a file/directory.
Type	String
Default Value	<empty> (means keeping it unchanged)
Valid Values	Any valid username.
Syntax	Any string.

See also: `SETOWNER`

10.14 Permission

Name	`permission`
Description	The permission of a file/directory.
Type	Octal
Default Value	755
Valid Values	0 - 777
Syntax	Any radix-8 integer (leading zeros may be omitted.)

See also: `CREATE`, `MKDIRS`, `SETPERMISSION`

10.15 Recursive

Name	`recursive`
Description	Should the operation act on the content in the subdirectories?
Type	boolean
Default Value	false
Valid Values	true | false
Syntax	true | false

See also: `RENAME`

10.16 Renewer

Name	`renewer`
Description	The username of the renewer of a delegation token.
Type	String
Default Value	<empty> (means the current user)
Valid Values	Any valid username.
Syntax	Any string.

See also: `GETDELEGATIONTOKEN`

10.17 Replication

Name	`replication`
Description	The number of replications of a file.
Type	short
Default Value	Specified in the configuration.
Valid Values	> 0
Syntax	Any integer.

See also: `CREATE`, `SETREPLICATION`

10.18 Sources

Name	`sources`
Description	A list of source paths.
Type	String
Default Value	<empty>
Valid Values	A list of comma seperated absolute FileSystem paths without scheme and authority.
Syntax	Any string.

See also: `CONCAT`,

10.19 Token

Name	`token`
Description	The delegation token used for the operation.
Type	String
Default Value	<empty>
Valid Values	An encoded token.
Syntax	See the note in Delegation.

See also: `RENEWDELEGATIONTOKEN`, `CANCELDELEGATIONTOKEN`

10.20 Username

Name	`user.name`
Description	The authenticated user; see Authentication.
Type	String
Default Value	null
Valid Values	Any valid username.
Syntax	Any string.

See also: Authentication