IBM Aspera HTTP Gateway 2.2

Contents

HTTP Gateway Admin Guide for Linux

IBM Aspera HTTP Gateway 2.2

Welcome to the HTTP Gateway documentation, where you can find information about how to install, maintain, and use the HTTP Gateway.

Getting started

Introduction

The HTTP Gateway is a service that allows a web application to leverage IBM Aspera technology to transfer data without requiring the end user to install IBM Aspera Connect software.

Common tasks

Installation

Troubleshooting and support

`gatewayconfig.properties` Reference

The default HTTP Gateway server settings and **ascp** settings are set in the `config/default.properties` file. You can override those default settings by creating the `gatewayconfig.properties` file and placing it in the same folder as the HTTP Gateway binary (by default `/opt/aspera/httpgateway/config/gatewayconfig.properties`).

Downloads and release notes

IBM Aspera downloads site

Introduction

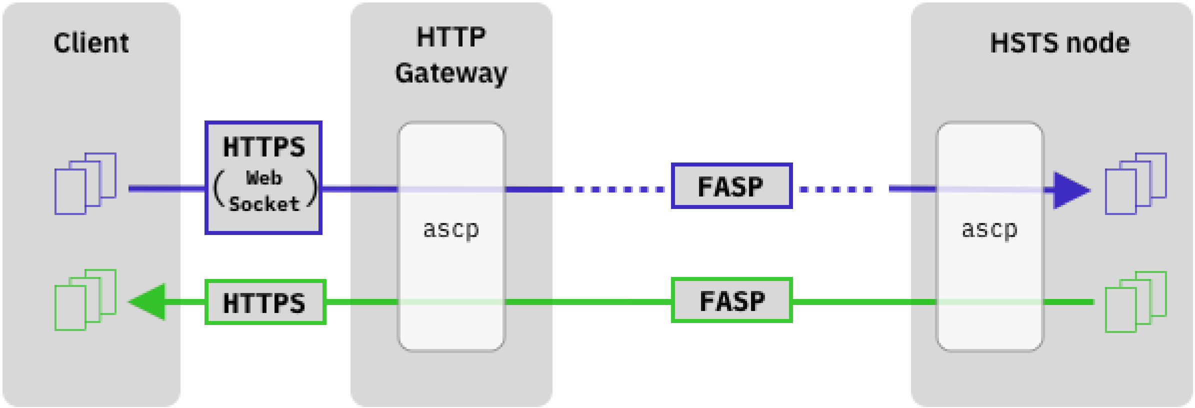
The HTTP Gateway is a service that allows a web application to leverage IBM Aspera technology to transfer data without requiring the end user to install IBM Aspera Connect software.

HTTP Gateway acts as a reverse proxy between a supported, modern web browser and an IBM Aspera High-Speed Transfer (HSTS) server, also known as a *node*. HTTP Gateway allows upload and download operations to the node by leveraging native, web-browser capabilities using HTTP.

HTTP Gateway is capable of transferring data using both HTTP and HTTPS. For security reasons, IBM Aspera highly recommends using HTTPS.

For downloads, HTTP Gateway uses the web browser's download manager. For uploads, HTTP Gateway uses a WebSocket connection.

The client connects to the HSTS node through the HTTP Gateway, downloading files over HTTPS and uploading files through a HTTPS (WebSocket) connection. The HTTP Gateway handles file transfer to and from the HSTS node. For best results, co-locate the HTTP Gateway as close as possible to your end users in order to keep the distance between the client and the HTTP Gateway short.



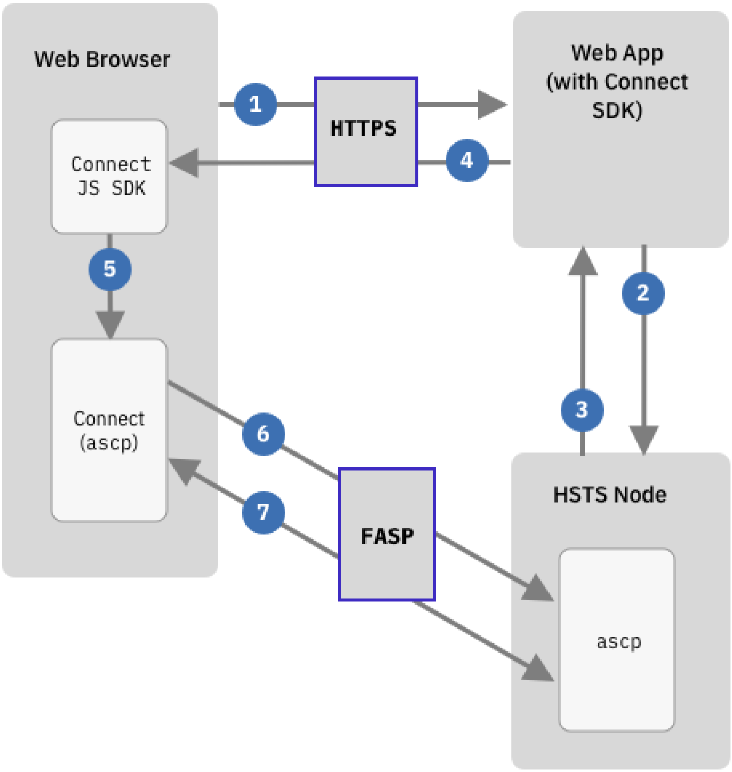
In Comparison with IBM Aspera Connect

IBM Aspera *FASP* transfers require two **ascp**-enabled machines (client and HSTS node) to establish a *FASP* transfer session. Web applications leveraging *FASP* for transfers can either:

- require their end users to install the Connect plug-in browser extension, which handles the *FASP* transfer session with the HSTS node.
- require their end users to transfer files (over HTTP/HTTPS) through HTTP Gateway, which handles the *FASP* transfer session with the HSTS node.

Transferring files with a HSTS node using Connect

A client makes a transfer with an HSTS node using Connect.



1 The web browser makes a transfer request to the web application.

2 The web application creates a transferSpec and submits it to the HSTS node.

3 The HSTS Node returns a transfer token.

4 The web application inserts the transfer token into the transferSpec and passes the transferSpec to the browser.

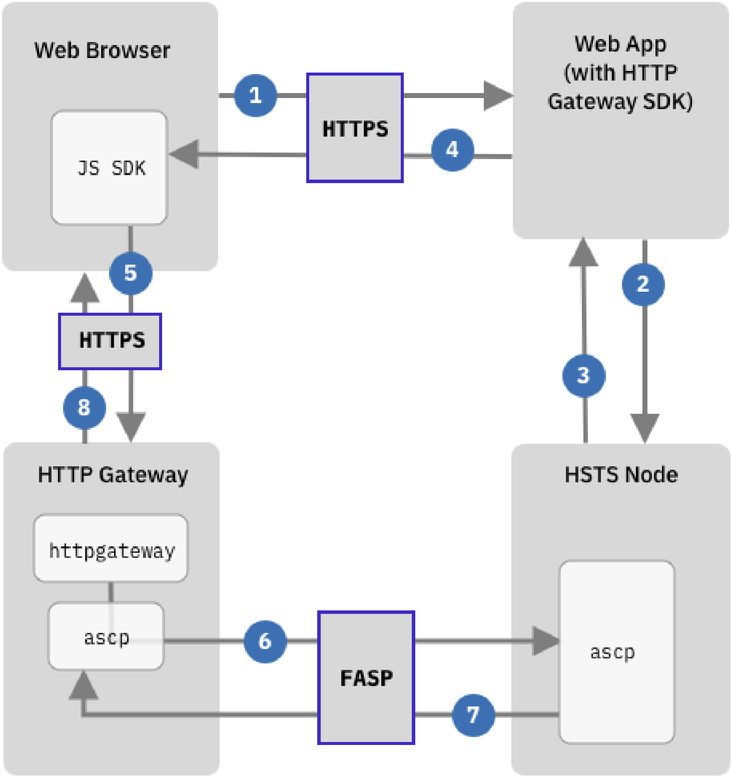
5 The browser passes the transferSpec to Connect.

6 Connect initiates a *FASP* transfer session with the HSTS node through ascp using the transfer token.

7 Connect performs the transfer with the HSTS node.

Receiving files from a HSTS node using HTTP Gateway

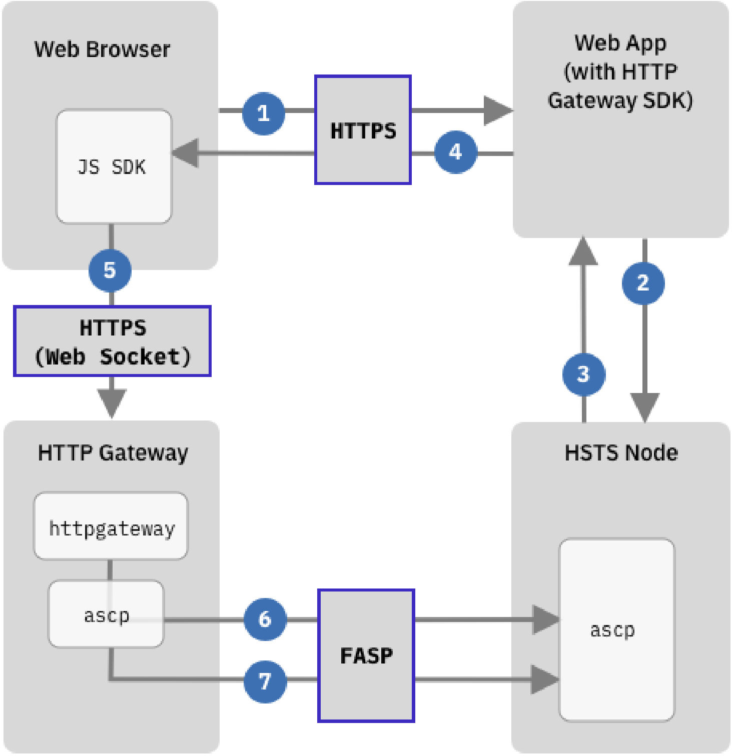
A client downloads files from a HSTS node using HTTP Gateway.



1. The web browser makes a transfer request to the web application.
2. The web application creates a transferSpec and submits it to the HSTS node.
3. The HSTS Node returns a transfer token.
4. The web application insert the transfer token into the transferSpec and passes the transferSpec to the browser.
5. The browser passes the transferSpec to HTTP Gateway.
6. HTTP Gateway initiates a *FASP* transfer session with the HSTS node through ascp using the transfer token.
7. HTTP Gateway retrieves requested files from the HSTS node.
8. HTTP Gateway sends requested files to the web browser download manager.

Sending files to a HSTS node using HTTP Gateway

A client uploads files to a HSTS node using HTTP Gateway.



1 The web browser makes a transfer request to the web application.

2 The web application creates a transferSpec and submits it to the HSTS node.

3 The HSTS Node returns a transfer token.

4 The web application inserts the transfer token into the transferSpec and passes the transferSpec to the browser.

5 The browser passes the transferSpec to HTTP Gateway and uploads chunked files.

6 HTTP Gateway initiates a *FASP* transfer session with the HSTS node through ascp using the transfer token.

7 HTTP Gateway transfers the files to the HSTS node.

Limitations

HTTP Gateway-based transfers are limited by the same limitations of HTTP/HTTPS transfers and the native download manager of the client's web-browser.

General Limitations

- Since uploads and downloads leverage HTTP/HTTPS between the web browser and HTTP Gateway, the transfer performance depends on the performance of HTTP/HTTPS, which can be affected by distance between clients and servers, and by other network-related issues. For this reason, IBM Aspera highly recommends following best practice by deploying HTTP Gateway as close as possible to end users.
- Empty (0-byte) files are not supported for uploads and downloads.
- HTTP Gateway does not support resuming transfers.

Download Limitations

When downloading more than one file, HTTP Gateway bundles the files in-memory. Bundling the files allows end users to download multiple files at once as one archive. Bundling the files also allows preserving a directory structure in the archive.

The total size of the archive cannot be communicated to the web browser, because files are bundled and transferred in-memory. Therefore, the download manager cannot show progress based on the total size.

Upload Limitations

- Since web browsers do not have an upload manager, IBM Aspera provides an upload mechanism through a JavaScript SDK. The upload mechanism allows sending multiple files as chunks, allowing the web page implementing the SDK to send large amounts of data. Because the web page sends the chunked data in the background, the user must stay on the same web page until the upload finishes.

  Note: You can find the JavaScript SDK documentation on the IBM Developer website.
- HTTP Gateway supports uploading only files and not directories.

Installation

Installing HTTP Gateway

Install HTTP Gateway on any server running a supported OS that can communicate with both the clients and the desired IBM Aspera High-Speed Transfer Server (HSTS) nodes.

About this task

While you can install HTTP Gateway on the same server as the HSTS nodes (if the servers has enough resources), IBM Aspera recommends running HTTP Gateway on a separate server. Installing on separate servers allows you to dedicate resources and to deploy the HTTP Gateway hosts close to clients.

Procedure

1. Before installing HTTP Gateway, review the system requirements in the HTTP Gateway release notes.
2. Download the HTTP Gateway RPM package.
3. Install HTTP Gateway:

   ```
   rpm -Uvh ibm-aspera-httpgateway-version.x86.64.rpm
   ```
4. Create the `gatewayconfig.properties` to configure HTTP Gateway settings.

   Copy the `default.properties` file as a base for the `gatewayconfig.properties`.

   ```
   cp /opt/aspera/httpgateway/config/default.properties /opt/aspera/httpgateway/config/
   gatewayconfig.properties
   ```

   Note:

   - The `gatewayconfig.properties` file overrides the settings in the `config/default.properties`.
   - The `gatewayconfig.properties` must live in a `config` folder at the same level as the **`aspera-httpgateway`** binary.
   - The `gatewayconfig.properties` file is preserved on upgrade.
5. Edit the `gatewayconfig.properties` file.

   IBM Aspera recommends securing your system by providing your own trusted root certificates for HTTPS. For more information, see "Configuring HTTP Gateway to Use HTTPS Scheme" on page 9.

For more information on the `gatewayconfig.properties` options, see “gatewayconfig.properties Reference” on page 11.

6. Start the `aspera_httpgateway` service:

   ```
   service aspera_httpgateway start
   ```

   For a `systemd` OS, run:

   ```
   systemctl start aspera_httpgateway
   ```

Upgrading HTTP Gateway

Upgrade HTTP Gateway to the latest version.

Procedure

1. Download the new installer.
2. Stop HTTP Gateway services:

   ```
   service aspera_httpgateway stop
   ```

   For a `systemd` OS, run:

   ```
   systemctl stop aspera_httpgateway
   ```

3. Install the new version:

   ```
   rpm -Uvh ibm-aspera-httpgateway-version.x86.64.rpm
   ```

4. Check the server status after upgrade:

   ```
   service aspera_httpgateway status
   ```

   For a `systemd` OS, run:

   ```
   systemctl status aspera_httpgateway
   ```

Uninstalling HTTP Gateway

Stop HTTP Gateway services and uninstall the package.

Procedure

1. Stop HTTP Gateway services:

   ```
   service aspera_httpgateway stop
   ```

   For a `systemd` OS, run:

   ```
   systemctl stop aspera_httpgateway
   ```

2. Uninstall HTTP Gateway:

   ```
   rpm -e ibm-aspera-httpgateway
   ```

Installing HTTP Gateway behind a reverse proxy

You can install HTTP Gateway behind a reverse proxy.

Nginx Configuration Example

The Nginx configuration example in this section has been successfully used with the current version of HTTP Gateway.

In this example:

- Nginx is installed on the same server as the HTTP Gateway server.
- To avoid double-encryption, the Nginx to HTTP Gateway connection is over HTTP, not HTTPS connection. Nginx settings require that the client connect over HTTPS.
- Nginx terminates the TLS connection with the client.
- HTTP Gateway listens on port 5080 using HTTP, while Nginx listens on port 6443 using HTTPs.

To use the example, replace the IP addresses and hostnames:

```
user  nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
  worker_connections  1024;
}

http {
   include       /etc/nginx/mime.types;
   default_type  application/octet-stream;

   map $remote_addr $proxy_forwarded_elem {
       # IPv4 addresses can be sent as-is
       ~^[0-9.]+$          "for=$remote_addr";

       # IPv6 addresses need to be bracketed and quoted
       ~^[0-9A-Fa-f:.]+$   "for=\"[$remote_addr]\"";

       # Unix domain socket names cannot be represented in RFC 7239 syntax
       default             "for=unknown";
   }

   map $http_forwarded $proxy_add_forwarded {
       # If the incoming Forwarded header is syntactically valid, append to it
       "~^(,[ \\t]*)*([!#$%&'*+.^_`|~0-9A-Za-z-]+=([!#$%&'*+.^_`|~0-9A-Za-z-]+|\"([\\t \\x21\
\x23-\\x5B\\x5D-\\x7E\\x80-\\xFF]|\\\\[\\t \\x21-\\x7E\\x80-\\xFF])*\"))?(;([!#$%&'*+.^_`|~0-9A-
Za-z-]+=([!#$%&'*+.^_`|~0-9A-Za-z-]+|\"([\\t \\x21\\x23-\\x5B\\x5D-\\x7E\\x80-\\xFF]|\\\\[\\t \
\x21-\\x7E\\x80-\\xFF])*\"))?)*([ \\t]*,([ \\t]*([!#$%&'*+.^_`|~0-9A-Za-z-]+=([!#$%&'*+.^_`|
~0-9A-Za-z-]+|\"([\\t \\x21\\x23-\\x5B\\x5D-\\x7E\\x80-\\xFF]|\\\\[\\t \\x21-\\x7E\\x80-\\xFF])*
\"))?(;([!#$%&'*+.^_`|~0-9A-Za-z-]+=([!#$%&'*+.^_`|~0-9A-Za-z-]+|\"([\\t \\x21\\x23-\\x5B\\x5D-\
\x7E\\x80-\\xFF]|\\\\[\\t \\x21-\\x7E\\x80-\\xFF])*\"))?)*)?)*$" "$http_forwarded,
$proxy_forwarded_elem";

       # Otherwise, replace it
       default "$proxy_forwarded_elem";
   }

   log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                     '$status $body_bytes_sent "$http_referer" '
                     '"$http_user_agent" "$http_x_forwarded_for"';


   log_format upstream_time '$remote_addr - $remote_user [$time_local] '
                             '"$request" $status $body_bytes_sent '
                             '"$http_referer" "$http_user_agent" $ssl_protocol $ssl_cipher '
                             'rt=$request_time uct=$upstream_connect_time uht=
$upstream_header_time urt=$upstream_response_time';

   access_log  /var/log/nginx/access.log  main;
   keepalive_timeout  240;
   gzip  off;

   upstream http_gateway_backend {
       # HTTP Gateway listens on port 5080
       server 127.0.0.1:5080;
   }

   server {
      access_log /var/log/nginx/http-gateway-access.log upstream_time;
      error_log /var/log/nginx/http-gateway-error.log debug;

      # The reverse proxy listens on port 6443
      listen 6443 ssl so_keepalive=on;
      server_name http-gateway.example.com;
      ssl_certificate /opt/aspera/common/apache/conf/server.crt;
      ssl_certificate_key /opt/aspera/common/apache/conf/server.key;
      client_max_body_size 0;
      proxy_read_timeout  600s;

      location /aspera/http-gwy/ {
          proxy_buffering off;
          proxy_request_buffering off;

          proxy_pass http://http_gateway_backend;

          proxy_set_header Host $host;
          proxy_set_header X-Real-IP $remote_addr;
          proxy_set_header X-Forwarded-Host $host:$server_port;
          proxy_set_header X-Forwarded-Server $host;
          proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
          proxy_set_header X-Forwarded-Proto $scheme;
          proxy_set_header Forwarded "$proxy_add_forwarded;proto=$scheme";

          # WebSocket support
          proxy_http_version 1.1;
          proxy_set_header Upgrade $http_upgrade;
          proxy_set_header Connection "upgrade";
      }
   }
}
```

Note: When behind a reverse proxy, HTTP Gateway adds additional fields (`x_real_ip`, `x_forwarded_for`, `x_forwarded_for`, `x_forwarded_for`, `forwarded`) to the `client_connection` section of the **ascp** transfer tags to provide external applications with transfer parameters and context.

For more information on tags, see "HTTP Gateway Transfer Tags" on page 15.

Configuration

Configuring HTTP Gateway to Use HTTPS Scheme

By default, Gateway is enabled to use HTTPS. Point HTTP Gateway to your own trusted root certificates instead of the included self-signed certificates.

Procedure

1. Open the `gatewayconfig.properties` file in a text editor.
2. Set `serverconfig.scheme` to `https`.
3. Provide the full paths to your certificate public and private keys in the `serverconfig.cert` and `serverconfig.key` options.
4. Save your changes.

Results

Note: If you install HTTP Gateway behind a reverse proxy, you can configure HTTP Gateway to use HTTP in order to avoid double encryption-decryption operations between reverse proxy and HTTP Gateway. Instead, use the reverse proxy to communicate securely with the client using HTTPS.

Restricting access to only authorized HSTS nodes

By default, HTTP Gateway allows clients to initiate transfer sessions with any HSTS node specified in the transfer specification as long as provided credentials are valid. For increased security, you can limit the nodes with which clients are allowed to initiate transfer sessions.

In the `gatewayconfig.properties` configuration file, edit the `transferconfig.authorized_hsts_nodes` property and set the value to a comma-separated list of HSTS nodes. Nodes can be IP addresses or FQDN's.

To allow transfers with all nodes, include the asterisk (*) wildcard as an independent entry in the list. Entries also respect regular expression matching.

For example:

```
["*"]                                     # allow access to any node
["aspera.ibmaspera.com", "*"]             # allow access to any node
["aspera.ibmaspera.com", "192.168.0.*"]   # allow access to example.com and any node in the
192.168.0.* IP range
["10.0.1.1", "aspera.ibmaspera.com"]      # allow access to only 10.0.1.1 and
aspera.ibmaspera.com
["10.0.1.1", ".*ibmaspera.com"]           # allow access to only 10.0.1.1 and to any subdomain
of ibmaspera.com
```

The `remote_host` property in the transfer specification must be an exact match of an entry in the `transferconfig.authorized_hsts_nodes`, unless the entry includes a wildcard.

For example:

`authorized_hsts_nodes`	**`remote_host` in the transfer specification**	**Result**
"ibmaspera.com"	"api.ibmaspera.com"	Transfer not allowed
"ibmaspera.com"	"ibmaspera.com"	Transfer allowed
".*ibmaspera.com"	"api.ibmaspera.com	Transfer allowed
".*ibmaspera.com"	"ibmaspera.com"	Transfer allowed

Run `ascp` transfer sessions as non-root user

About this task

Important: The configuration only applies when HTTP Gateway is running as `root`.

By default, HTTP Gateway runs **ascp** transfer sessions as the `root` user. If you are running the service as `root`, you can run transfer sessions as another user by changing the `ascpconfig.run_as_user` and `ascpconfig.run_as_group` properties in the gatewayconfig.properties configuration file.

If you choose to run transfer sessions as another user, that user must have correct permissions on the server. These instructions show how to configure HTTP Gateway to start transfer sessions as a new `httpgateway` user instead of as `root`.

Procedure

1. Create the `httpgateway` user and group:

   ```
   groupadd --gid 9000 httpgateway
   useradd -M --uid 9000 --gid 9000 httpgateway
   ```

2. Edit the gatewayconfig.properties configuration file (by default, `/opt/aspera/httpgateway/config/gatewayconfig.properties`) to use the `httpgateway` user and group for the `ascpconfig.run_as_user` and `ascpconfig.run_as_group` properties:

```
## This property will execute the ascp command as the user provided
ascpconfig.run_as_user=httpgateway
## This property will execute the ascp command as the group provided
ascpconfig.run_as_group=httpgateway
```

3. Give the `httpgateway` user ownership and write permissions to the `config` folder:

   ```
   chown -R httpgateway:httpgateway /opt/aspera/httpgateway/config
   chmod -R 755 /opt/aspera/httpgateway/config
   ```

4. Give the `httpgateway` user ownership and write permissions to the `SendFileListDir` folder (configurable in gatewayconfig.properties):

   ```
   chown root:httpgateway "/opt/aspera/httpgateway/SendFileListDir/"
   chmod 775 "/opt/aspera/httpgateway/SendFileListDir/"
   ```

5. Give the `httpgateway` user ownership and write permissions to the logs (folder configurable in gatewayconfig.properties):

   ```
   chown root:httpgateway /opt/aspera/httpgateway/aspera/log/
   chmod 775 /opt/aspera/httpgateway/aspera/log/
   ```

6. Start the `aspera_httpgateway` service:

   ```
   service aspera_httpgateway restart
   ```

   For a `systemd` OS, run:

   ```
   systemctl restart aspera_httpgateway
   ```

Results

The `root` user starts HTTP Gateway, but **ascp** transfer sessions use the `httpgateway` user.

Optionally, you can also start HTTP Gateway with the `httpgateway` user (and not as `root`).

Appendix

`gatewayconfig.properties` Reference

The default HTTP Gateway server settings and **ascp** settings are set in the `config/default.properties` file. You can override those default settings by creating the `gatewayconfig.properties` file and placing it in the same folder as the HTTP Gateway binary (by default `/opt/aspera/httpgateway/config/gatewayconfig.properties`).

`default.properties`

```
serverconfig.host=0.0.0.0
serverconfig.port=443
serverconfig.scheme=https
serverconfig.cert=/opt/aspera/httpgateway/config/server.crt
serverconfig.key=/opt/aspera/httpgateway/config/server.key
serverconfig.cors=["*"]
serverconfig.min_tls_version=1.2
serverconfig.max_tls_version=1.3
serverconfig.inbound_data_timeout=120
serverconfig.sys_log_disabled=false

ascpconfig.dir=/opt/aspera/httpgateway/aspera/
ascpconfig.private_key_path=/opt/aspera/httpgateway/aspera/asperatokenauthidrsa
ascpconfig.debug_enabled=false
ascpconfig.log_dir=/opt/aspera/httpgateway/aspera/log/
ascpconfig.destination_root_dir=/
ascpconfig.send_stdio_tar_total_bytes_and_files=false
ascpconfig.run_as_user=root
ascpconfig.run_as_group=root
```

```
transferconfig.zip_compression=6
transferconfig.source_file_list_tmpdir=/opt/aspera/httpgateway/SendFileListDir
transferconfig.cap_transfer_policy=fair
transferconfig.cap_max_transfer_rate=-1
transferconfig.cap_min_transfer_rate=0
transferconfig.management_server_port=9999
transferconfig.authorized_hsts_nodes=["*"]
```

`serverconfig`

This section configures the server:

Option	Description	Default
`host`	The host address that HTTP Gateway binds to.	`0.0.0.0`
`port`	The host port host port that HTTP Gateway listens to.	`443`
`scheme`	The HTTP scheme, which can be either `http` or `https`.	`https`
`cert`	The full path to the public key, required when using the `https` scheme.	`/opt/aspera/httpgateway/config/server.crt`
`key`	The full path to the private key, required when using the `https` scheme.	`/opt/aspera/httpgateway/config/server.key`
`cors`	The Cross-Origin Request Policy that determines from where people are allowed to connect to HTTP Gateway. The default `["*"]` allows connections from anywhere, but setting it to `["example.com", "10.0.0.0"]` only allows connections from those addresses.	`["*"]`
`min_tls_version`	The minimum accepted TLS version. HTTP Gateway supports TLS 1.0, 1.1, 1.2, and 1.3.	`1.2`
`max_tls_version`	The maximum accepted TLS version. HTTP Gateway	`1.3`

Option	Description	Default
	supports TLS 1.0, 1.1, 1.2, and 1.3.	
inbound_data_timeout	HTTP Gateway considers the websocket connection STALLED after this duration, and closes the connection.	120
sys_log_disabled	Prevent HTTP Gateway from attempting to write to syslog. Logs are written to stdout instead.	false

ascpconfig

This section configures settings for the *FASP* transfer session the HTTP Gateway initiates with a node using the **ascp** binary:

Option	Description	Default
dir	The full path to directory containing configuration files for **ascp**. The directory must contain these files: • **ascp** (binary) • aspera-license • aspera.conf	/opt/aspera/httpgateway/aspera/
private_key_path	The full path to the private key used by **ascp** to authenticate the transfer to the node.	/opt/aspera/httpgateway/aspera/asperatokenauthidrsa
debug_enabled	Determines whether HTTP Gateway runs **ascp** in debug mode. If enabled, you are required to specify the directory for the logs (log_dir).	false
log_dir	The directory to hold ascp logs if debug mode is enabled.	
destination_root_dir	The destination root directory for uploaded files. If the transferSpec does not specify the destination root directory, files are uploaded to this directory. This directory are relative to the destination node docroot. For example, if the node docroot is /usr/project1 and the destination_root_dir	/

Option	Description	Default
	is /aspera, HTTP Gateway uploads files to /user/project/aspera.	

transferconfig

This section configures transfer settings:

Option	Description	Default
zip_compression	The ZIP compression ratio used for downloading multiple files at once.	6
source_file_list_tmpdir	The directory that holds a temporary file that stores the downloading-files list, when the total size of the list is greater than 4k bytes.	/opt/aspera/httpgateway/SendFileListDir
cap_transfer_policy	The highest transfer policy allowed. This overwrites the transferSpec setting For example, if the cap_transfer_policy is set to fair and the transfer policy in the transferSpec is set to low,HTTP Gateway respects the transferSpec setting. But if the transfer policy in the transferSpec is set to high, HTTP Gateway continues to use the fair set in cap_transfer_policy.	fair
cap_max_transfer_rate	The max transfer rate allowed. This overwrites the transferSpec setting.	-1
cap_min_transfer_rate	The minimum transfer rate allowed. This overwrites the transferSpec setting.	0
scp_passphrase	The passphrase used to authenticate the ascp private key (not set by default).	
management_server_port	The port used to set the **ascp** management port, which HTTP Gateway uses to manage **ascp** transfer sessions and to listen for transfer information.	9999
authorized_hsts_nodes	The list (in array format) of HSTS nodes with which clients are allowed to initiate transfer sessions. You can designate IP addresses or FQDN's.	["*"]

Option	Description	Default
	To allow transfers with all nodes, use the asterisk (*) wildcard: ["*"]. To use the asterisk (*) wildcard, it must be the only entry in the list.	

HTTP Gateway Transfer Tags

HTTP Gateway provides additional information to **ascp** transfer tags to provide external applications with transfer parameters and context.

Tag Example

Note: When behind a reverse proxy, HTTP Gateway adds additional fields (`x_real_ip`, `x_forwarded_for`, `x_forwarded_for`, `x_forwarded_for`) to the `client_connection` section of the **ascp** transfer tags to provide external applications with transfer parameters and context.

```
{
  "aspera": {
    "xfer_id": "9c03f82d-53e1-4c85-b6fe-96c9f88a3dc3",
    "http-gateway": {
      "uuid": "9c03f82d-53e1-4c85-b6fe-96c9f88a3dc3",
      "client_connection": {
        "peer_ip_address": "127.0.0.1",
        "peer_tcp_port": "40580",
        "local_ip_address": "127.0.0.1",
        "local_tcp_port": "5080",
        "x_real_ip": "66.211.105.2",
        "x_forwarded_for": "66.211.105.2",
        "x_forwarded_host": "http-gateway.example.com:6443",
        "x_forwarded_proto": "https",
        "forwarded": "for=66.211.105.2;proto=https",
        "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/79.0.3945.130 Safari/537.36"
      },
      "transfer_request": {
        "direction": "send",
        "target_rate": "100000",
        "file_count": "3",
        "encryption_at_rest": "no",
        "file_obfuscation": "no"
      },
      "archiver": {
        "method": "none",
        "compression_ratio": "N/A"
      }
    }
  }
}
```

Tags Reference

Tag	Description	Reverse-proxy specific?	Example
`xfer_id`	Unique ID assigned to the transfer	No	`9c03f82d-53e1-4c85-b6fe-96c9f88a3dc3`
`uuid`	Same as `xfer_id`	No	`9c03f82d-53e1-4c85-b6fe-96c9f88a3dc3`
`peer_ip_address`	The IP address of the client that initiated communication with HTTP Gateway	No	`127.0.0.1`

Tag	Description	Reverse-proxy specific?	Example
`peer_tcp_port`	The TCP port of the client that initiated communication with HTTP Gateway	No	`40580`
`local_ip_address`	The IP address of the HTTP Gateway server that received and served the request	No	`127.0.0.1`
`local_tcp_port`	The TCP port of the HTTP Gateway server that received and served the request	No	`6443`
`x_forwarded_for`	Representation of the HTTP X-Forwarded-For header; typically represents the original IP address of the client that established the connection with the reverse proxy	Yes	`66.211.105.2`
`x_real_ip`	Representation of the HTTP X-Real-IP header; typically represents the original IP address of the client that established the connection with the reverse proxy	Yes	`66.211.105.2`
`x_forwarded_host`	Representation of the HTTP X-Forwarded-Host header; typically represents the hostname and port used by the client to connect to the reverse proxy	Yes	`http-gateway.example.com:6443`
`x_forwarded_proto`	Representation of the HTTP X-Forwarded-Proto header; typically represents the protocol (typically `https`) used between the client and the reverse proxy	Yes	`https`
`forwarded`	Representation of the HTTP Forwarded header; typically provides information about the real IP address of the client that established the connection with the	Yes	`for=66.211.105.2;proto=https`

Tag	Description	Reverse-proxy specific?	Example
	reverse proxy, and some optional information **Note:** This is an optional HTTP header defined in RFC 7239: Section 4.		
`user_agent`	Represents a value passed by the client software (usually web browsers) that connects to the HTTP Gateway server or to the reverse proxy	No	`Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36`
`direction`	For user upload, the direction is from HTTP Gateway server to the HSTS node (`send`). For user download, the direction is from the HSTS node to the HTTP Gateway server (`receive`)	No	`send`
`target_rate`	The original target rate requested by the web application	No	`100000`
`file_count`	The number of files in the transfer	No	`3`
`encryption_at_rest`	Whether encryption-at-rest was requested by the web application	No	`no`
`file_obfuscation`	Whether file obfuscation was requested by the web application	No	`no`
`method`	The archive method, none when uploading, and `zip` when downloading multiple files at once	No	`none`
`compression_ratio`	ZIP compression ratio used for downloading multiple files at once	No	`N/A`

HTTP Gateway Performance Metrics

HTTP Gateway is designed to introduce no bottlenecks, but performance results depend on the resources available.

Performance depends on:

- Distance between the end user and the server hosting the HTTP Gateway service

Note: For better performance, IBM Aspera highly recommends following best practice by deploying HTTP Gateway as close as possible to end users.

- Number of files transferred
- Size of files transferred
- Number of concurrent transfers
- Current network conditions
- Total memory available on the client
- Total memory available on the server hosting the HTTP Gateway service

IBM Aspera has successfully uploaded and downloaded multiple-GB, large files through the HTTP Gateway. IBM Aspera has also successfully uploaded and downloaded 1,000 files at once. These metrics are not hard limits, but serve to inform baseline expectations.

Installing HTTP Gateway as a non-root user

Install HTTP Gateway as a non-root user.

About this task

Important:

- The changes in this procedure do not persist on upgrade. You need to perform the same operations after each upgrade of HTTP Gateway.
- Running processes as a non-root user prevents HTTP Gateway from listening on a privileged TCP port (`1-1023`). This may not be a problem if HTTP Gateway is located behind a load-balancer that binds to 443 and redirects to HTTP Gateway on an unprivileged port.

Procedure

1. As the `root` user, create a new system user with the name `httpgateway`.
2. Create a system group with the name `httpgateway`, and add the `httpgateway` user to it.
3. Allow `httpgateway` user to run applications with **sudo**:

   ```
   usermod -aG wheel httpgateway
   ```

4. Log in as the `httpgateway` user.
5. Install HTTP Gateway with sudo:

   ```
   sudo rpm -Uvh ibm-aspera-httpgateway-version.rpm
   ```

6. Create the `gatewayconfig.properties` to configure HTTP Gateway settings.

   Copy the `default.properties` file as a base for the `gatewayconfig.properties`.

   ```
   cp /opt/aspera/httpgateway/config/default.properties /opt/aspera/httpgateway/config/
   gatewayconfig.properties
   ```

   Note:

   - The `gatewayconfig.properties` file overrides the settings in the `config/default.properties`.
   - The `gatewayconfig.properties` must live in a `config` folder at the same level as the **`aspera-httpgateway`** binary.
   - The `gatewayconfig.properties` file is preserved on upgrade.

7. Edit the `gatewayconfig.properties` file.

   IBM Aspera recommends securing your system by providing your own trusted root certificates for HTTPS. For more information, see “Configuring HTTP Gateway to Use HTTPS Scheme” on page 9.

For more information on the gatewayconfig.properties options, see "gatewayconfig.properties Reference" on page 11.

8. Start the aspera_httpgateway service:

   ```
   service aspera_httpgateway start
   ```

   For a systemd OS, run:

   ```
   systemctl start aspera_httpgateway
   ```

Running HTTP Gateway processes as a non-root user

You can run HTTP Gateway as a non-root user.

About this task

Important:

- The changes in this procedure do not persist on upgrade. You need to perform the same operations after each upgrade of HTTP Gateway.
- Running processes as a non-root user prevents HTTP Gateway from listening on a privileged TCP port (1-1023). This may not be a problem if HTTP Gateway is located behind a load-balancer that binds to 443 and redirects to HTTP Gateway on an unprivileged port.
- When running HTTP Gateway process as a non-root user, you cannot run **ascp** as another user by configuring ascpconfig.run_as_user and ascpconfig.run_as_group. That configuration only applies when HTTP Gateway is running as root.

Procedure

1. As the root user, stop the **aspera_httpgateway** service:

   ```
   service aspera_httpgateway stop
   ```

   For a systemd OS, run:

   ```
   systemctl stop aspera_httpgateway
   ```

2. Create a system user, such as httpgateway, that you want to run the **aspera_httpgateway** and **ascp** processes.
3. Create a system group, such as httpgateway, and add the user to it.
4. Change the HTTP Gateway config folder permissions:

   Using the httpgateway group as an example:

   ```
   chown root:httpgateway /opt/aspera/httpgateway/config
   chmod 775 /opt/aspera/httpgateway/config
   ```

5. Check if the http-gateway.pid files exists at:

   /opt/aspera/httpgateway/config/http-gateway.pid

   If it exists, delete the file.
6. Change the permissions of the directory defined in transferconfig.source_file_list_tmpdir (default is /opt/aspera/httpgateway/SendFileListDir) in the gatewayconfig.properties file:

   Using the httpgateway group and the default directory as an example:

   ```
   chown root:httpgateway /tmp/SendFileListDir
   chmod 775 /tmp/SendFileListDir
   ```

7. Change the permissions of the log directory defined in `ascpconfig.log_dir` (default is `/opt/aspera/httpgateway/aspera/log`) in the `gatewayconfig.properties` file:

   Using the `httpgateway` group and the default log directory as an example:

   ```
   chown root:httpgateway /opt/aspera/httpgateway/aspera/log
   chmod 775 /opt/aspera/httpgateway/aspera/log
   ```

8. Edit the `/etc/systemd/system/multi-user.target.wants/aspera_httpgateway.service` file:

   Find the line:

   ```
   ExecStart=/bin/bash -ce "/opt/aspera/httpgateway/aspera-httpgateway start > /opt/aspera/
   httpgateway/httpgateway.log 2>&1"
   ```

   Replace the line with:

   ```
   ExecStart=/bin/bash -ce "sudo -u httpgateway /opt/aspera/httpgateway/aspera-httpgateway
   start > /opt/aspera/httpgateway/httpgateway.log 2>&1"
   ```

9. Start the **`aspera_httpgateway`** service:

   ```
   service aspera_httpgateway start
   ```

   For a `systemd` OS, run:

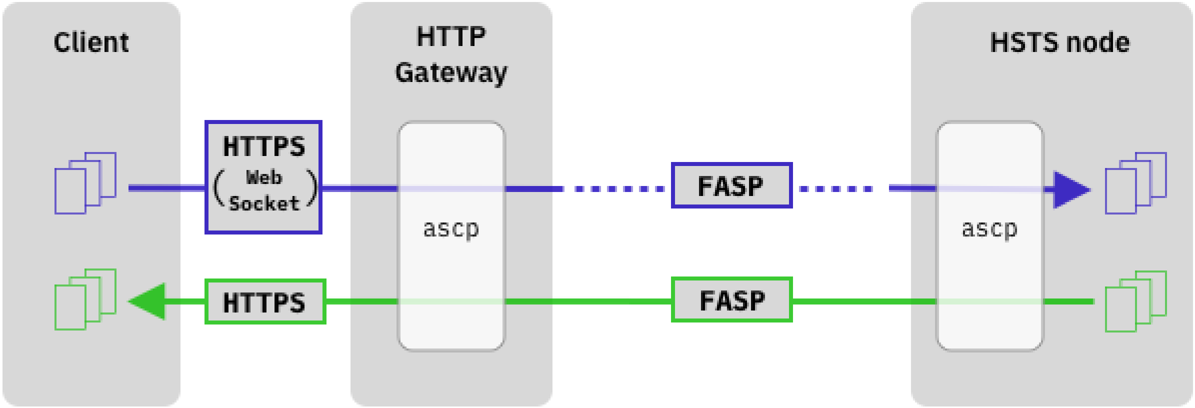
   ```
   systemctl start aspera_httpgateway
   ```

HTTP Gateway defaults to target rate set by the HSTS node

HTTP Gateway accepts a transfer specification to start uploads and downloads. If the transfer specification does not include the `transfer_rate` field, HTTP Gateway defaults to using the transfer rate defined by the HSTS node.

There are two legs to transfer using HTTP Gateway:

- Between the client and HTTP Gateway through HTTPS
- Between HTTP Gateway and the HSTS node through FASP



The `transfer_rate` field in the transfer specification configures the speed of transfer between HTTP Gateway and the HSTS node through FASP. The `transfer_rate` field is important, because it sets expectations for the speed of the transfer. If you do not set the `transfer_rate` in the transfer specification, HTTP Gateway defaults to using the transfer rate defined by the HSTS node independent of the speed of your HTTPS transfer speed, which may be set at a slower rate than what your bandwidth can handle.

To understand the implications, compare these two scenarios:

Scenario one: `transfer_rate` is not set

Your average upload speed is 1 GB/s

The `transfer_rate` defined by the HSTS node is 400 MB/s.

There is no `transfer_rate` set in the transfer specification.

Your expected transfer rate is 400 MB/s.

To increase your transfer rate, change the settings in the `aspera.conf` configuration file on the node.

Scenario two: `transfer_rate` is not set

Your average upload speed is 1 GB/s

The `transfer_rate` defined by the HSTS node is 400 MB/s.

The `transfer_rate` in the transfer specification is set to 800 MB/s

Your expected transfer rate is 800 MB/s.

Troubleshooting

Transfers fail because the temporary file list directory no longer exists

When a the total size of the downloading-files list of a transfer is greater than 4k bytes, HTTP Gateway temporarily stores the list of files in a temporary directory defined in the `gatewayconfig.properties` file. If the temporary directory is deleted (for example, by a system process) and HTTP Gateway is unable to recreate the temporary directory at the specified location, the transfer will fail.

By default, HTTP Gateway stores the temporary file list at `/opt/aspera/httpgateway/SendFileListDir`. If you change the location (using the `gatewayconfig.properties` file), make sure the directory does not get deleted by any process on the system.