IBM Aspera faspio Gateway 1.3.2

Contents

Introduction

IBM Aspera faspio Gateway is a lightweight software component for high-speed bidirectional data transport. Using the patented Aspera FASP protocol, faspio achieves speeds of up to 5 Gb/sec of aggregated FASP traffic over unmanaged networks.

IBM Aspera faspio Gateway fully utilizes available bandwidth to transfer data in byte-order sequence at the maximum possible speed with near-zero latency. It removes the barriers of size, distance, and complexity to move data between on-premises and cloud infrastructures. Provides significant improvements in performance and service quality when transferring data between highly remote or dispersed locations in unfavorable network conditions, such as high latency and packet loss.

Aspera faspio Gateway is a software component that can be integrated quickly and easily with existing applications that use a TCP/UDP connection for their data flow. It improves nearly all server-to-server TCP-based data flows regardless of the distance and network conditions.

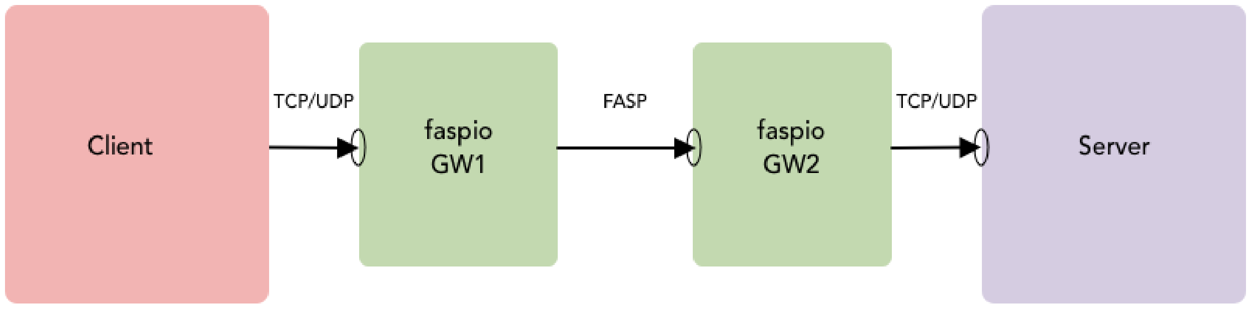
IBM Aspera faspio Gateway acts as a transport layer proxy between TCP/UDP and Aspera FASP.

Usage

Gateway Client/Server Usage

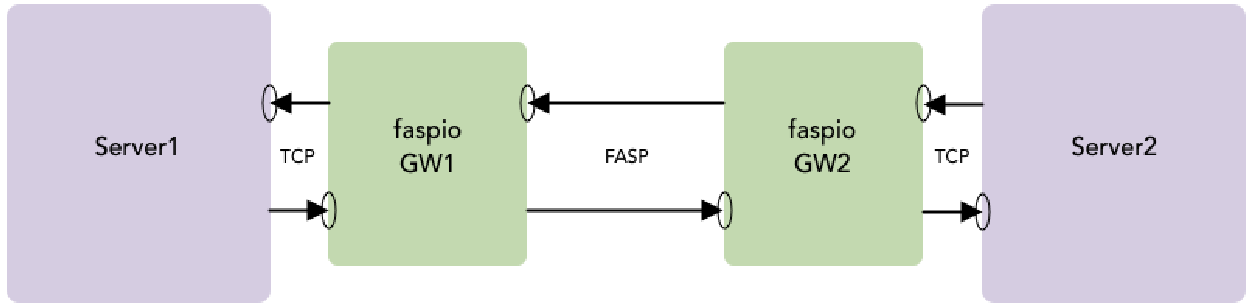
In this configuration, two faspio Gateways are used to bridge TCP/UDP connections from TCP/UDP clients to a TCP/UDP server over FASP:

Note: UDP supports only one connection at a time.



Gateway Server/Server Usage

For some use cases, such as DB replication or messaging services (like MQ or Event Streams), communication must be established by both sides. In this mode, each server initiates a connection to the other:



For information on how to configure these two cases, see "Configuring the Gateway" on page 5.

License

The faspio Gateway requires a license for using the Gateway as a FASP server. No license is required to use the Gateway as a client.

Perpetual license

- If you purchased faspio Gateway, you can obtain your license key by emailing the IBM Aspera license team at **aspera-license@ibm.com** with the site ID and the order number for your purchase, the part numbers and quantities that are required, and the email addresses to which the licenses must be sent.
- If you encounter any issue with the license you obtained, send an email to **aspera-licenses@wwpdl.vnet.ibm.com**.

Per-license limits

License limit	Limit value
Maximum total per-process aggregate bandwidth (in and out traffic combined)	5 Gbps
Maximum number of bridges per gateway	50 bridges per gateway
Maximum number of concurrent sessions per gateway	20 sessions per gateway

License errors

If you start the Gateway with an invalid license, you might see one of the following errors.

- If you do not have a license, faspio Gateway reports:

```
The current license does not allow for FASP to run in server-mode (i.e.
bridge.local.protocal="fasp")
[2022-01-25 12:26:42.899] [  gateway  ] [ error ] license-checking failure - "no license
loaded"
```

- If your license is expired, faspio Gateway reports:

```
The current license does not allow for FASP to run in server-mode (i.e.
bridge.local.protocal="fasp")
[2022-01-25 12:26:42.899] [  gateway  ] [ error ] license-checking failure - "license expired"
```

PDFs

1.3.1

- IBM Aspera faspio Gateway 1.3.1 Guide
- IBM Aspera faspio Gateway 1.3.1 Release Notes

1.3

- IBM Aspera faspio Gateway 1.3 Guide
- IBM Aspera faspio Gateway 1.3 Release Notes

1.2

- IBM Aspera faspio Gateway 1.2 Guide
- IBM Aspera faspio Gateway 1.2 Release Notes

1.1

- IBM Aspera faspio Gateway 1.1 Guide
- IBM Aspera faspio Gateway 1.1.1 Release Notes
- IBM Aspera faspio Gateway 1.1.0 Release Notes

1.0

- IBM Aspera faspio Gateway 1.0 Guide
- IBM Aspera faspio Gateway 1.0.1 Release Notes

Installation and configuration

Installing on macOS

Download and install IBM Aspera faspio Gateway on your macOS system.

To install faspio Gateway on your macOS machine:

1. Ensure that you're on a supported version of macOS. Supported versions are listed in the release notes.
2. Download the macOS `.pkg` installer from IBM Passport Advantage: https://www.ibm.com/products/software

   The name of the installer will be similar to:

   ```
   ibm-faspio-gateway_1.1.0-version_macOS.pkg
   ```
3. Run the installer by opening the file and following the prompts.
4. If you are using faspio Gateway as a FASP server, you must provide a valid license at `/usr/local/etc/faspio/aspera-license`.

   For more information about configuration and client versus server usage, see "Configuring the Gateway" on page 5.

The locations of installed files are shown below:

Executable

```
/usr/local/bin/faspio-gateway
```

IBM Aspera license

```
/usr/local/etc/faspio/aspera-license
```

Configuration files

```
/usr/local/etc/faspio/gateway.toml
/usr/local/etc/faspio/logging.toml
```

Launchd service configuration file

```
/Library/LaunchDaemons/com.ibm.aspera.faspio-gateway.plist
```

IBM SWID file

```
/usr/local/share/faspio-gateway/iso-swid/ibm.com_IBM_Aspera_faspio_Gateway-1.0.0.swidtag
```

IBM software licenses

```
/usr/local/share/faspio-gateway/license/*
```

Installing on Linux

Download and install IBM Aspera faspio Gateway on your Linux system.

To install faspio Gateway on your Linux machine:

1. Ensure that you're on a supported version of Linux. Supported versions are listed in the release notes.
2. Download the Linux installer from IBM Passport Advantage: https://www.ibm.com/products/software
3. Choose the `.deb` or `.rpm` package as appropriate to your system. The names of the installers follow the syntax:

   ```
   ibm-faspio-gateway_1.2.0-version_arch.deb
   ibm-faspio-gateway_1.2.0-version_arch.rpm
   ```
4. Install from either the `.deb` or `.rpm` package:
   - `.deb`:

     ```
     sudo apt install ./ibm-faspio-gateway_1.2.0-version_arch.deb
     ```
   - `.rpm`:

     ```
     sudo yum install ./ibm-faspio-gateway_1.2.0-version_arch.rpm
     ```
5. If you are using faspio Gateway as a FASP server, you must provide a valid license at `/usr/local/etc/faspio/aspera-license`.

   For more information about configuration and client versus server usage, see "Configuring the Gateway" on page 5.

The locations of installed files are shown below:

Executable

```
/usr/local/bin/faspio-gateway
```

IBM Aspera license

```
/usr/local/etc/faspio/aspera-license
```

Configuration files

```
/usr/local/etc/faspio/gateway.toml
/usr/local/etc/faspio/logging.toml
```

SystemD service configuration file

```
/usr/local/lib/systemd/system/faspio-gateway.service
```

IBM SWID file

```
/usr/local/share/faspio-gateway/iso-swid/ibm.com_IBM_Aspera_faspio_Gateway-1.0.0.swidtag
```

IBM software licenses

```
/usr/local/share/faspio-gateway/license/*
```

Installing on Windows

Download and install IBM Aspera faspio Gateway on your Windows system.

To install faspio Gateway on your Windows machine:

1. Ensure that you're on a supported version of Windows and are logged in with an account that has administrator privileges. Supported versions are listed in the release notes.
2. Download the Windows `.msi` installer from IBM Passport Advantage: https://www.ibm.com/products/software The name of the installer will be similar to:

   ```
   ibm-faspio-gateway_version_win64.msi
   ```
3. Open the `.msi` file. The installer wizard launches. Follow the prompts to complete the installation.
4. If you are using faspio Gateway as a FASP server, you must provide a valid license at `C:\Program Files\IBM\faspio Gateway\config\aspera-license`.

   For more information about configuration and client versus server usage, see "Configuring the Gateway" on page 5.

The locations of installed files are shown below. All files are found under:

```
C:\Program Files\IBM\faspio Gateway\
```

Executable

```
bin\faspio-gateway
```

IBM Aspera license

```
config\aspera-license
```

Configuration files

```
config\gateway.toml
config\logging.toml
```

IBM SWID file

```
iso-swid\ibm.com_IBM_Aspera_faspio_Gateway-1.0.0.swidtag
```

IBM software licenses

```
license\*
```

Configuring the Gateway

Configure two or more faspio Gateway servers to connect to each other.

The Gateway configuration file

The IBM Aspera faspio Gateway configuration file, `gateway.toml`, is located here:

Linux, macOS:

```
/usr/local/etc/faspio/gateway.toml
```

Windows:

```
C:\Program Files\IBM\faspio Gateway\config\gateway.toml
```

The `gateway.toml` file that is included in the installation is provided as a template. Modify this file to specify your ports, hostnames, and so on.

Every time that you modify `gateway.toml` to make changes for your configuration, you must restart the IBM faspio Gateway service. For information on starting and stopping the Gateway service, see "Launching faspio Gateway" on page 12.

Note: The version of the FASP protocol that is included in faspio Gateway uses a single UDP port. Whatever port you configure for your FASP connection over the WAN must have the same UDP port open on any firewalls along the connection path.

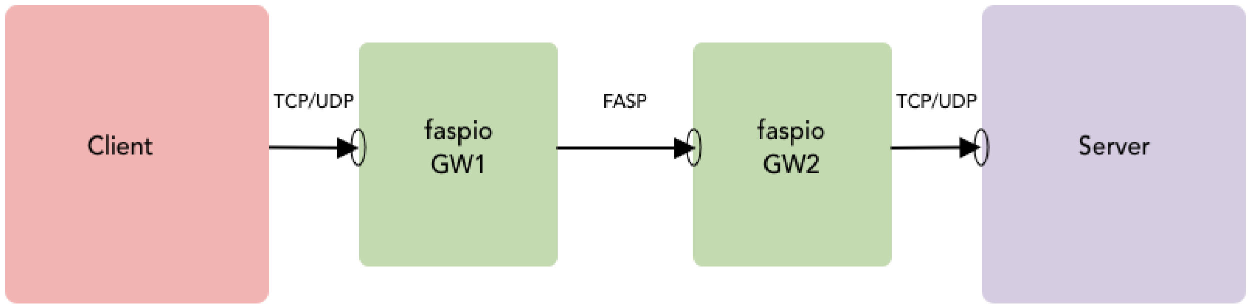
Configuration examples

Important: For simplicity, the examples in this section do not include security settings. After understanding how to configure the connection between servers, make sure to understand how to secure your servers. Security is enabled by default and you need to properly configure security settings before you can use your Gateway servers.

Example: Client to Server

In this configuration, two Gateways are used to establish a FASP bridge between two TCP/UDP connections, a TCP/UDP client, and a TCP/UDP server:

Note: UDP supports only one connection at a time. For some use cases, such as video or streaming services, communication must be established by the client side.



Given a server listening on port 12345, configure your client to point to Gateway 1 (`GW1 IP`) and port 12345:

GW1 Configuration

```
[[bridge]]
   [bridge.local]
       protocol = "tcp"
           host = "0.0.0.0"
           port = 12345

   [bridge.forward]
       protocol = "fasp"
           host = "GW2"
           port = 12345
```

GW2 Configuration

```
[[bridge]]
   [bridge.local]
       protocol = "fasp"
           host = "0.0.0.0"
           port = 12345

   [bridge.forward]
       protocol = "tcp"
           host = "Server"
           port = 12345
```

protocol
The transfer protocol type, either `"tcp"` or `"fasp"`. The protocol type must be quoted.

host
The hostname or IP address, always quoted.

port
Either of these:

- Port number (quoting optional, except for the port range).
- Known port/service name (quoting required), such as `"http"` or `"ftp-data"`.

You can specify a range or a list of numbered ports.

Port range:

- Use a hyphen and quotation marks; for example, `"100-110"`.
- The range of port numbers must be ascending (low to high); for example, `"100-110"`, but not `"100-90"`.
- The size of the range cannot exceed 50 ports.
- If a port range is used, the range for the local port and the forward port must be the same size.
- The ranges do not need to consist of the same ports. For example, the local range might be `"501-509"` and the forward range might be `"511-519"`; however, the forward range might not be `"511-520"`.

List of ports:

- Use commas and quotation marks; for example, `"12345, 12347, 12349, 12351"`.

Port range configuration: Many to one port

You can now define a range of ports in the configuration file (toml) on the local interface, and a single port on the forward interface. This allows you to add connections on a port range to a single port when forwarded.

Example: Many to one port

```
[[bridge]]
        name = "Bridge-many2one"
        [bridge.local]
                protocol = "fasp"
                port = "4142-4144"
                host = "0.0.0.0"
                tls_enabled = false

        [bridge.forward]
                protocol = "tcp"
                port = "12346"
                host = "127.0.0.1"
                tls_enabled = false
```

Example: List of ports

```
[[bridge]]
        name = "Bridge-port-list"
        [bridge.local]
                protocol = "fasp"
                port = "4142,4144"
                host = "0.0.0.0"
                tls_enabled = false

        [bridge.forward]
                protocol = "tcp"
                port = "12346,12348"
                host = "127.0.0.1"
                tls_enabled = false
```

Example: Port range

```
[[bridge]]
        name = "Bridge-port-range"
        [bridge.local]
                protocol = "fasp"
                port = "4142-4144"
                host = "0.0.0.0"
                tls_enabled = false

        [bridge.forward]
                protocol = "tcp"
                port = "12346-12348"
                host = "127.0.0.1"
                tls_enabled = false
```

Example: Forwarding to the first available host

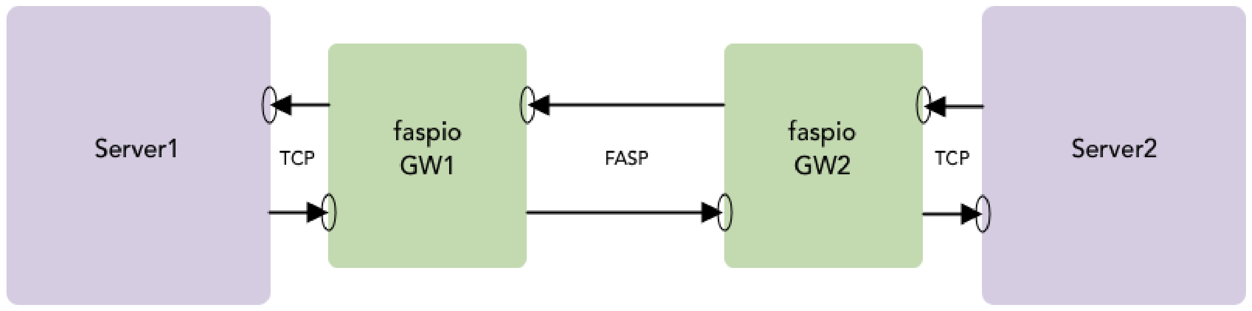
For `bridge.forward`, Gateway can loop through an array of specified hostnames or IP addresses and forward to the first available host it finds. For example,

```
[bridge.forward]
       protocol = "fasp"
           host = ["GW2", "10.0.0.2"]
           port = 12345
```

A hostname can resolve to multiple IP addresses. If a hostname is specified, either as a single entry or as an entry within the array, each of its IP addresses are tried until a connection is established.

Example: Server to Server

For some use cases, such as DB replication or messaging services (like MQ or Event Streams), communication must be established by both sides. In this mode, each server initiates a connection to the other:



GW1 configuration

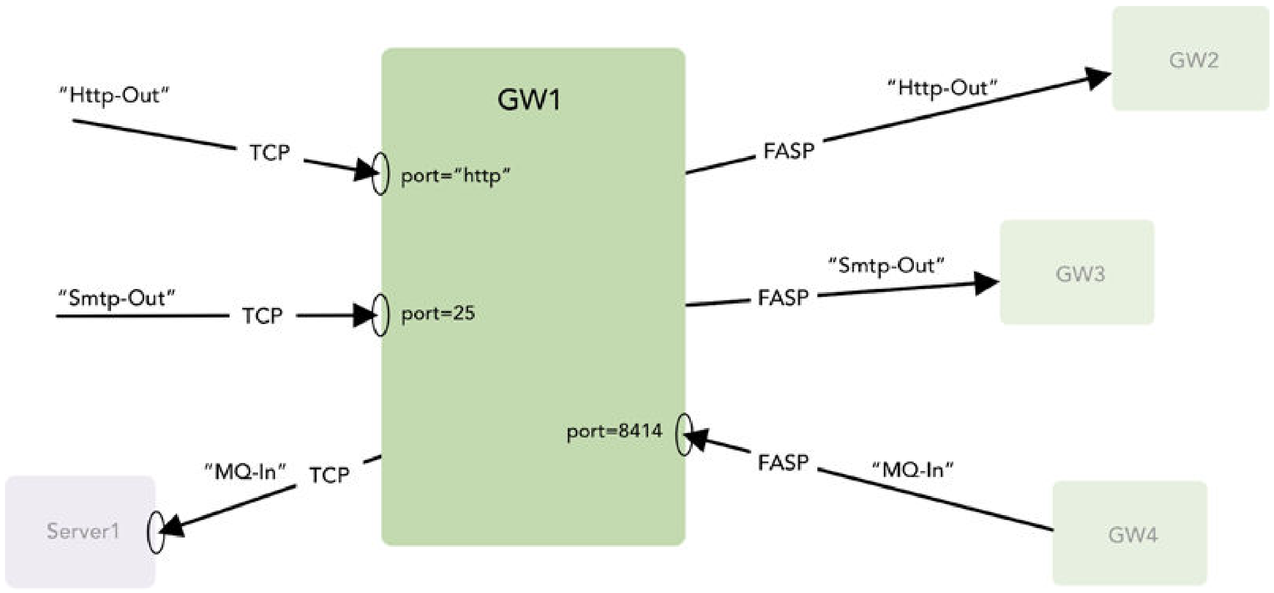
```
[[bridge]]
   name = "Outbound"
   [bridge.local]
       protocol = "tcp"
           host = "0.0.0.0"
           port = 12345

   [bridge.forward]
       protocol = "fasp"
           host = "GW2"
           port = 12345

[[bridge]]
   name = "Inbound"
   [bridge.local]
       protocol = "fasp"
           host = "0.0.0.0"
           port = 54321

   [bridge.forward]
       protocol = "tcp"
           host = "Server1"
           port = 54321
```

GW2 configuration

```
[[bridge]]
   name = "Outbound"
   [bridge.local]
       protocol = "tcp"
           host = "0.0.0.0"
           port = 54321

   [bridge.forward]
       protocol = "fasp"
           host = "GW1"
           port = 54321

[[bridge]]
   name = "Inbound"
   [bridge.local]
       protocol = "fasp"
           host = "0.0.0.0"
           port = 12345

   [bridge.forward]
       protocol = "tcp"
           host = "Server2"
           port = 12345
```

Gateway configuration options

Gateway can also be configured with multiple bridges, multiple ports, multiple destinations, and multiple services. The following image shows a single gateway that is configured with examples of these combinations.



Bridge: HTTP-Out

```
[[bridge]]
   name = "Http-Out"
   [bridge.local]
      protocol = "tcp"
        host =
"0.0.0.0"
        port = "http"

   [bridge.forward]
      protocol = "fasp"
        host = "GW2"
        port = "http"
```

Bridge: Smtp-Out

```
[[bridge]]
   name = "Smtp-
Out"
   [bridge.local]
      protocol =
"tcp"
        host =
"0.0.0.0"
        port = 25

   [bridge.forward]
      protocol =
"fasp"
        host =
"GW3"
        port = 25
```

Bridge: MQ-In

```
[[bridge]]
   name = "MQ-In"
   [bridge.local]
      protocol =
"fasp"
        host =
"0.0.0.0"
        port = 8414

   [bridge.forward]
      protocol =
"tcp"
        host =
"Server1"
        port = 8414
```

Binding to a specific address

This configuration allows you to bind the forwarding connection to a specific NIC on the machine. To achieve this, we can add the option `bind_address` in the config file.

Local parameters:

`host`

Deprecated and replaced with `bind_address`.

`protocol`

This is a required field.

`bind_address`

This is a required field. It can be an IP address or a hostname, and it specifies which interface to listen on. `127.0.0.1` for the loopback interface, `0.0.0.0` to listen on all interfaces.

`port`

This is a required field.

Forward parameters:

`host`

This is a required field. Represents the remote host to connect to. It can either be a single IP address or hostname, as well as an array of IP addresses or hostnames. The hostname can also resolve to multiple

IP addresses, the system will attempt to connect to the first IP address and then the other ones if that fails.

`protocol`

This is a required field.

`bind_address`

This is an optional field. It can be an IP address or a hostname and it specifies which interface to connect to.

`port`

This is a required field.

Example: Binding to a specific address

```
[bridge.local]
           protocol = "tcp"
       bind_address = "127.0.0.1"
               port = 12345
        tls_enabled = false
    [bridge.forward]
           protocol = "fasp"
               host = "localhost"
               port = 12345
        tls_enabled = false
```

Configuring Prometheus for monitoring

Prometheus gathers and saves its measurements as time series data. The information about the metrics is kept along with the timestamp at which it was captured. To enable Prometheus, update the configuration to `enabled = true`. To disable Prometheus, keep the default value `enabled = false`.

To monitor the faspio Gateway by using Prometheus, use the following configuration:

```
[admin]
               host = "127.0.0.1"
               port = "8081"
        tls_enabled = true    # Set this to false to disable TLS
         cert_chain = "tls/gw1_cert_chain.pem"

 prometheus_enabled = true
    prometheus_auth = "basic"

[admin.basic_auth]
           username = "test"
    password_digest = 
"$sha-512$esJ0jf7Jc4D8jvjjRauf0ExWrHXG6jvzGx4mIe6X3dh4A87bQEuLRJnY4md+l4teeMaubIK85EFUFoqdmak4Ig
==$Rhhk3CrSSyemgrBP"
```

You must configure the parameters that Prometheus uses to receive the data for the monitoring metrics. These are the parameters:

host
IP address or hostname of the interface to listen on. Default: `127.0.0.1`.

port
Port number to listen on. Default: `8080`.

tls_enabled
Secure Prometheus endpoint with TLS. Default: `true`.

cert_chain
Path to the SSL certificate file. The file must be in PEM format and it must have both the private key and the certificate. Default: `''`.

prometheus_enabled
Enable or disable exposing metrics through Prometheus. Default: `false`.

prometheus_auth
Specify the authentication type. Set it to "basic" or "none". Default: basic.

username
Name of the user that must be authenticated.

password_digest
The password digest of the user that must be authenticated.

Note: If TLS is enabled for a bridge and cert_key is not defined, the gateway defaults to certificate and cert_chain for the private key. For example, this allows the same cert_chain path to be used for both bridges and Prometheus.

Password digest

To generate the password digest, run the following command:

```
faspio-gateway digest
```

Follow the prompts so the output gives the password digest that you need to add to the gateway.toml along with the wanted username. Add the values under the [admin.basic_auth] section as in the following example,

Note: Create the [admin.basic_auth] section if doesn't exist in the gateway.toml file.

```
[admin.basic_auth]
           username = "test"
           password_digest = "$sha-512$BPd2kBmlW+f7pTc/pFPiSwBCOMYqpPqzyWN9VVPyY/
JssqdO0J5f6Zie6+MIbv9r8ntTPUv/mWHmbiRUYnwJmg==$EcsNAKGiEWTMbQwI"
```

For more information, see "Launching faspio Gateway" on page 12.

Securing the Gateway

The faspio Gateway uses Transport Layer Security (TLS) to secure your TCP connections and initiate key exchange for the FASP protocol.

Important: TLS is enabled by default on all your bridges. You must provide valid certificates to your Gateways before they can connect.

Configuring TLS

When two Gateways connect to each other, they use Mutual TLS (mTLS) authentication to ensure traffic is secure and trusted in both directions. mTLS requires a certificate chain, a certificate key, and a verification key on both servers. Here is a sample configuration of two servers configured with mTLS:

GW1 Configuration (Client)

```
      [[bridge]]
         [bridge.local]
            protocol = "tcp"
                host = "127.0.0.1"
                port = "2000-2001"
         tls_enabled = true
          cert_chain = "tls/
gw1_cert_chain.pem"
            cert_key = "tls/gw1_cert_key.pem"
              verify = "tls/verify.pem"

          [bridge.forward]
            protocol = "fasp"
                host = "Gateway2"
                port = "3000-3001"
         tls_enabled = true
          cert_chain = "tls/
gw1_cert_chain.pem"
            cert_key = "tls/gw1_cert_key.pem"
```

GW2 Configuration (Server)

```
      [[bridge]]
         [bridge.local]
            protocol = "fasp"
                host = "127.0.0.1"
                port = "3000-3001"
         tls_enabled = true
          cert_chain = "tls/
gw2_cert_chain.pem"
            cert_key = "tls/gw2_cert_key.pem"
              verify = "tls/verify.pem"

          [bridge.forward]
             protocol = "tcp"
                 host = "127.0.0.1"
                 port = "4000-4001"
         tls_enabled = true
          cert_chain = "tls/
gw2_cert_chain.pem"
```

GW1 Configuration (Client)

```
             verify = "tls/verify.pem"
host_verify_enabled = true
```

GW2 Configuration (Server)

```
             cert_key = "tls/gw2_cert_key.pem"
               verify = "tls/verify.pem"
```

cert_chain
: The relative path to the certificate chain signed by a valid CA.

cert_key
: The relative path to the private key matching the cert chain.

verify
: The relative path to the verification key that verifies the other server's chain is signed by the correct CA.

host_verify_enabled
: Optionally enforce additional security by requiring that the `host` field defined in the client's `bridge.forward` section matches the Common Name in the server's SSL certificate.

Disabling TLS

To disable TLS, add `tls_enabled = false` to the section for which connection you want to disable TLS. For example, if your faspio Gateway servers and HSTS servers run in the same private and secure network, you may consider disabling TLS for those sections only:

GW1 Configuration (Client)

```
      [[bridge]]
         [bridge.local]
            protocol = "tcp"
                host = "127.0.0.1"
                port = "2000-2001"
          tls_enabled = false

          [bridge.forward]
            protocol = "fasp"
                host = "Gateway2"
                port = "3000-3001"
         tls_enabled = true
          cert_chain = "tls/
gw1_cert_chain.pem"
            cert_key = "tls/gw1_cert_key.pem"
              verify = "tls/verify.pem"
host_verify_enabled = true
```

GW2 Configuration (Server)

```
      [[bridge]]
         [bridge.local]
            protocol = "fasp"
                host = "127.0.0.1"
                port = "3000-3001"
         tls_enabled = true
          cert_chain = "tls/
gw2_cert_chain.pem"
            cert_key = "tls/gw2_cert_key.pem"
              verify = "tls/verify.pem"

          [bridge.forward]
             protocol = "tcp"
                 host = "127.0.0.1"
                 port = "4000-4001"
          tls_enabled = false
```

Launching faspio Gateway

To launch IBM Aspera faspio Gateway, you start or stop the **`faspio-gateway`** service using one of these methods:

- Linux: Uses the **`systemd`** service manager. Start the service from the command line.
- Windows: Start the service from the command line or from the Windows Services panel.
- macOS: Uses the **`launchd`** service manager. Start the service from the command line.

Note: Each time you change your Gateway configuration—by modifying `gateway.toml`—you must restart the service. For information about configuring your system, see "Configuring the Gateway" on page 5.

Launching on macOS

The OS service manager on macOS is **launchd**, and the terminal command for controlling it is **launchctl**. You can load, unload, start, stop, or check the status of the **faspio-gateway** service by running the **launchctl** sub-commands.

launchctl sub-commands

- Load the service. Loading the service also starts it:

```
$ sudo launchctl load /Library/LaunchDaemons/com.ibm.aspera.faspio-gateway.plist
```

- Start the loaded service (such as a stopped service that is still loaded):

```
$ sudo launchctl start com.ibm.aspera.faspio-gateway
```

- Check the status of the service:

```
$ sudo launchctl list | grep com.ibm.aspera.faspio-gateway
```

- Stop the service:

```
$ sudo launchctl stop com.ibm.aspera.faspio-gateway
```

- Unload the service:

```
$ sudo launchctl unload /Library/LaunchDaemons/com.ibm.aspera.faspio-gateway.plist
```

Launching on Windows

To launch the faspio Gateway service on Windows, you start the service from either the command line or the Windows Services panel. Both methods require that you have admin privileges.

To start the service from the command line, run:

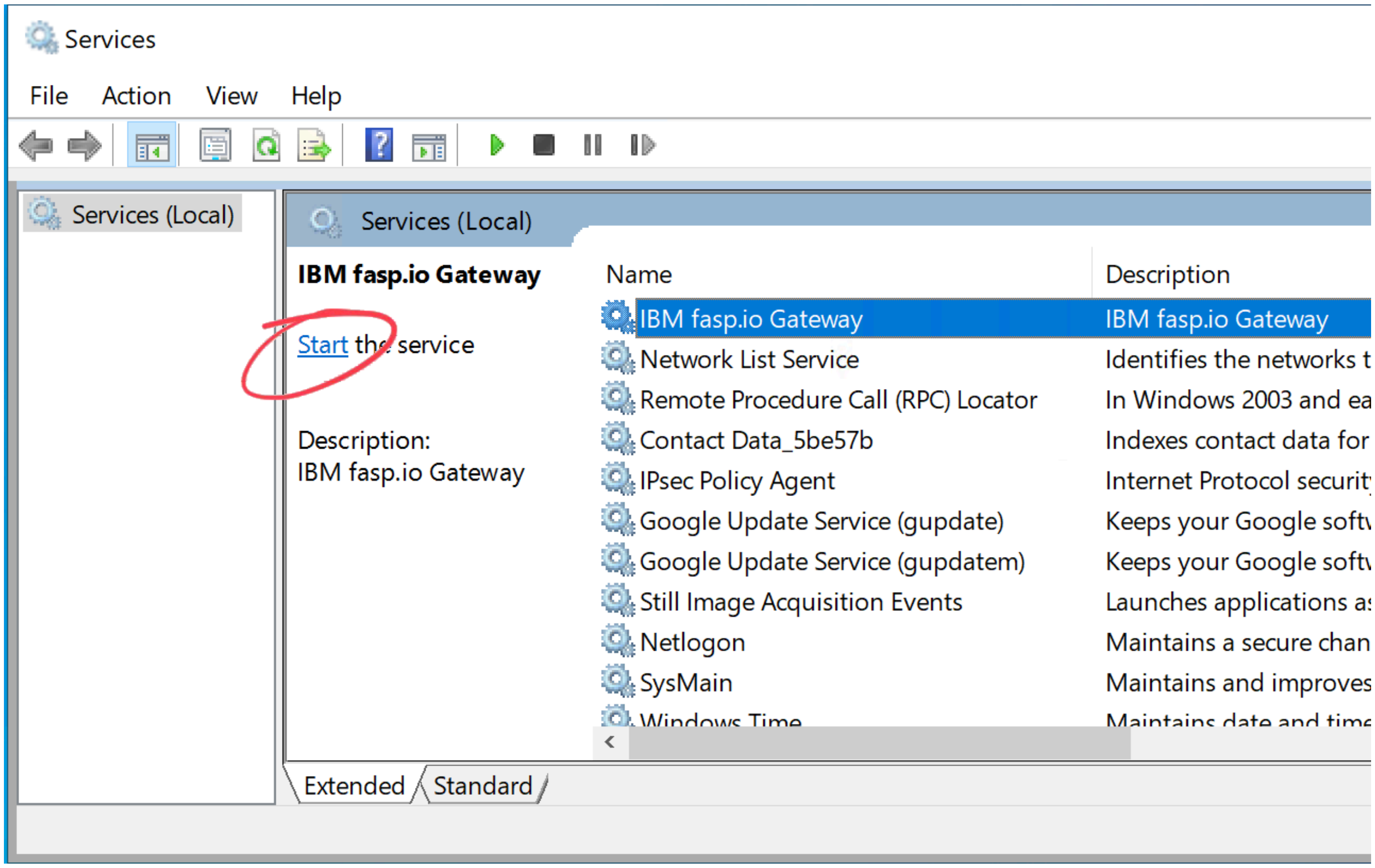
```
net start faspio-gateway
```

To stop the service:

```
net stop faspio-gateway
```

To start the service from the Windows UI, open the Services panel. To access it quickly, enter "services" in the search box next to the **Start** button:

In the display that appears, click **Services**. The Windows **Services** panel opens. In the list of services, find and select **IBM faspio Gateway**. To launch the service, click **Start the service**.



IBM fasp.io Gateway

Start the service

Description:
IBM fasp.io Gateway

Name	Description
IBM fasp.io Gateway	IBM fasp.io Gateway
Network List Service	Identifies the networks t
Remote Procedure Call (RPC) Locator	In Windows 2003 and ea
Contact Data_5be57b	Indexes contact data for
IPsec Policy Agent	Internet Protocol securit
Google Update Service (gupdate)	Keeps your Google softv
Google Update Service (gupdatem)	Keeps your Google softv
Still Image Acquisition Events	Launches applications a:
Netlogon	Maintains a secure chan
SysMain	Maintains and improves
Windows Time	Maintains date and time

Launching on Linux

The OS service manager used on Linux is **systemd**, and the terminal command for controlling it is **systemctl**. You can load, unload, start, stop, or check the status of the **faspio-gateway** service by running **systemctl** sub-commands. You can also use **journalctl** to check service logs.

systemctl sub-commands

- Start the service:

```
sudo systemctl start faspio-gateway
```

- Stop the service:

```
sudo systemctl stop faspio-gateway
```

- Enable the service (to restart after reboot):

```
sudo systemctl enable faspio-gateway
```

- Check service status:

```
sudo systemctl status faspio-gateway
```

- See service logs:

```
sudo journalctl --unit=faspio-gateway
```

Service not found error message

If you see the following error message when running one of these commands, `systemd` may not be monitoring the folder for the faspio Gateway service:

```
Unit faspio-gateway.service not found.
```

If you get the error, the following command usually resolves the problem:

```
sudo systemctl daemon-reload
```

Testing the Gateway

The tests below use the sample configuration from "Configuring the Gateway" on page 5.

Simple echo test through `netcat`

On both machines, run:

```
faspio-gateway
```

On the server machine (GW2), listen:

```
nc -v -l 12345
```

On the client machine (GW1), connect to the server (GW2):

```
nc GW2_ip_addr 12345
```

`netcat` should report a connection from your client machine.

Stream data through `netcat`

On both machines, run:

```
faspio-gateway
```

On the server machine (GW2), listen:

```
nc -v -l 12345 > /dev/null
```

On the client machine (GW1), connect and stream 1 GB of data to the server (GW2):

```
dd if=/dev/zero count=1024 bs=1M  | nc -v  GW2_ip_addr 12345 >/dev/null
```

`netcat` should report a successful transfer.

Appendices

Configuring the logging file

Logging Configuration File

Logging configuration is defined in `logging.toml`, which is located here:

Linux, macOS:

```
/usr/local/etc/faspio/logging.toml
```

Windows:

```
C:\Program Files\IBM\faspio Gateway\config\logging.toml
```

Error, warning, and info logs are sent to the console by default. In the default `logging.toml` file, four loggers are made available:

`gateway` – High-level logger for the gateway.
`s2s` – Stream-to-stream session class logger.
`faspio-cpp` – Logger for the Asio/C++ FASP SDK.
`faspio-c` – Logger for the FASP protocol.

Note: Every time you modify `logging.toml`, you must restart the faspio Gateway service.

For more information on how to configure logging, see the full reference at:

`https://github.com/guangie88/spdlog_setup`

Notes:

The `level` setting is optional for both sinks and loggers.
The `level` for error logging is `err`, not `error`.
The `_st` suffix means single-threaded.
The `_mt` suffix means multi-threaded.
`syslog_sink` is thread-safe by default. No `_mt` suffix is required.

The `spdlog` default logging format is:

```
[2014-10-31 23:46:59.678] [loggername] [info] message
```

For information on how to customize `spdlog` formatting, see:

`https://github.com/gabime/spdlog/wiki/3.-Custom-formatting`

Uninstalling on macOS

To uninstall faspio Gateway:

1. Unload the service:

   ```
   sudo launchctl unload /Library/LaunchDaemons/com.ibm.aspera.faspio-gateway.plist
   ```

2. Delete all the faspio installed files listed in the section above (executable, configuration files, `launchd` service config file, SWID file, and license files).

Uninstalling on Linux

To uninstall faspio Gateway, run:

- `.deb`:

  ```
  sudo apt uninstall ./ibm-faspio-gateway
  ```

- `.rpm`:

  ```
  sudo yum uninstall ./ibm-faspio-gateway
  ```

Note: Previous versions of faspio Gateway (1.1 and before) are called `ibm-fasp.io-gateway`.

Uninstalling on Windows

To uninstall faspio Gateway:

1. From the **Start** menu, go to **Settings > System > Apps & Features**.
2. Select the **IBM faspio Gateway** app. The **Uninstall** button appears.
3. Click **Uninstall** and confirm.

Note: Previous versions of faspio Gateway (1.1 and before) are called `ibm-fasp.io-gateway`.

Beta REST API guide

Introduction

The introduction of the faspio Gateway Beta Rest API offers the ability to manage gateways and control the way bridges are created.

This guide provides basic examples (with sample code) for managing, creating, and getting information on bridges.

Beta API reference

See the faspio Gateway Beta API reference (OAS 3 format) and developer guide available from the IBM API Hub on the IBM Developer website.