STOMP Message Transport for the Swaptacular Messaging Protocol

Evgeni Pandurksi

2023-07-04

Overview

Each Swaptacular node should run one or more publicly accessible servers, allowing peer Swaptacular nodes to connect to these servers as clients, and post messages.

This document specifies how Swaptacular nodes can use a subset of the STOMP 1.2 protocol, to interoperably send Swaptacular Messaging Protocol (SMP) messages from one peer node to another.

Note: The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

A Subset of the STOMP Protocol

> STOMP[1] is a simple interoperable protocol designed for asynchronous message passing between clients via mediating servers. It defines a text based wire-format for messages passed between these clients and servers.

Swaptacular nodes must support, both as servers, and as clients, the following subset of the STOMP 1.2 specification:

- The STOMP 1.2 commands `STOMP`, `CONNECT`, `CONNECTED`, `SEND`, `RECEIPT`, `ERROR`, and `DISCONNECT` MUST be fully supported.
- Support for STOMP *subscriptions* and *transactions* is OPTIONAL. That is: `SUBSCRIBE`, `UNSUBSCRIBE`, `MESSAGE`, `ACK`, `NACK`, `BEGIN`, `COMMIT`, and `ABORT` commands may not be implemented. Swaptacular nodes MUST NOT presume that their peer nodes will understand these commands.

[1] Simple Text Oriented Messaging Protocol: https://stomp.github.io/

- In addition to the requirements stated in the STOMP 1.2 specification, every `SEND` command MUST include the following headers:

  receipt Specifies a message ID.

  type Specifies the type of the SMP message.

  Here is a non-exhaustive list of possible message types:

  - `ConfigureAccount`
  - `PrepareTransfer`
  - `FinalizeTransfer`
  - `RejectedConfig`
  - `RejectedTransfer`
  - `PreparedTransfer`
  - `FinalizedTransfer`
  - `AccountUpdate`
  - `AccountPurge`
  - `AccountTransfer`

  content-type Specifies the MIME type of the message body.

  Every Swaptacular node MUST support the JSON serialization format (the `application/json` MIME type), and MAY support additional serialization formats and MIME types.

  persistent MUST have the value `true`.

Each Swaptacular node MUST run one or more publicly accessible servers that support the above described STOMP subset.

Note: In addition to servers that support the above described STOMP subset, Swaptacular nodes MAY run servers that support other message transport protocols. When some other message transport protocol is supported by both the server and the client, they MAY agree to use it instead.

STOMP Connections

When a Swaptacular node wants to send some SMP messages to a peer Swaptacular node, the sending node opens a client STOMP connection to one (or more) of the receiving node's servers, and issues a `SEND` command for each of the messages. The client MUST consider a message to be successfully delivered, only after a `RECEIPT` command has been received from the server, confirming that the message has been processed[2].

The client MAY decide to keep the STOMP connection open for any length of time, and the server SHOULD NOT terminate the connection unilaterally, without a reason.

[2] Every STOMP `RECEIPT` command confirms the delivery of all preceding messages.

STOMP connections MUST be secured by using **Transport Layer Security** version 1.3 or higher. Both the client and the server must present a certificate, which the other side verifies before proceeding with the connection. That is:

- Clients MUST require servers to authenticate themselves by presenting a trusted certificate chain. Clients SHOULD NOT perform *hostname verification*[3].
- Servers MUST require clients to authenticate themselves by presenting a trusted certificate chain.

STOMP Servers Manifest Files

Every Swaptacular node MUST publicly provide a *STOMP servers manifest file*, which describes the STOMP servers that the node runs. The RECOMMENDED name for this file is `stomp.toml`.

STOMP Servers Manifest Files are regular TOML[4] files that contain values for the following configuration keys:

servers A list of server addresses in the form `"hostname:port"`.

> The `hostname` can be a fully qualified domain name, or an IP address; `port` specifies the TCP port that the servers listens on. To initiate a new connection, the client SHOULD randomly choose one of the server addresses from the list. Note that the list MAY contain the same server address more than once, which would simply increase the probability for that address to be chosen by clients.

host A value for the `host` header in `CONNECT`[5] commands.

> The client MUST substitute all occurrences of the string `${NODE_ID}` in the value, with the ID of the client's Swaptacular node. For example, if the value is `"/${NODE_ID}"`, and the client's node ID is `12345678`, then the client must send the header `"host:/12345678"` with each `CONNECT` command to the server.

login An *optional* value for the `login` header in `CONNECT` commands.

> Servers SHOULD NOT require clients to include a `login` header (an username) in `CONNECT` commands.
>
> The client MUST substitute all occurrences of the string `${NODE_ID}` in the value, with the ID of the client's Swaptacular node.

[3] The *hostname verification* involves looking at the certificate sent by the server, and verifying that the `dnsName` in the `subjectAltName` field of the certificate matches the host portion of the URL used to make the connection.

[4] Tom's Obvious Minimal Language: https://toml.io/en/

[5] The STOMP protocol specification requires servers to handle the `STOMP` command in the same manner as the `CONNECT` command. Therefore, everything said in this section applies to the `STOMP` command as well.

passcode An *optional* value for the `passcode` header in `CONNECT` commands.

Servers SHOULD NOT require clients to include a `passcode` header (a password) in `CONNECT` commands.

destination A value for the `destination` header in `SEND` commands.

The client MUST substitute all occurrences of the string `${NODE_ID}` in the value, with the ID of the client's Swaptacular node.

accepted-content-types An *optional* list of supported MIME types for the message bodies, starting with the most preferable.

Support for the `application/json` MIME type is implied. Therefore, an empty (or missing) list means that only `application/json` is supported.

STOMP Servers Manifest Files MAY contain additional key/value pairs, which are not described in this document.

Example `stomp.toml` file:

```
servers = [
  "server1.example.com:1234",
  "server2.example.com:1234",
  "201.202.203.204:2345",
]
host = "/"
destination = "/exchange/${NODE_ID}"
accepted-content-types = [
  "application/vnd.google.protobuf",
  "application/msgpack",
]
not-described-here = true
```