

Cookie Protocol Specification

1. Overview

This document specifies version 1.0 of the Cookie Protocol (CP). CP emulates an application layer protocol by utilizing the PHY protocol.

First, any client has to make itself known to the server by sending a cookie request message. The switch stores the respective information for any sender. This specification does not detail how the switch stores this information internally.

Only after receipt of a cookie, a can send commands to the server. Each command has its own workflow of messages exchanged between client and server.

2. Protocol behavior

2.1. Requesting cookies

A client requests a cookie from the server by sending a cookie request message. The client waits for a cookie response message to be returned. When no response is received within two seconds the request is considered lost and should be resent to the server. If three consecutive cookie request messages time out, an exception has to be returned to the client app.

There shall be at most one active cookie for any client at any point in time. A client is uniquely identified by the clients IP address and UDP port number. If a client requests a cookie any previously issued cookies shall be invalidated. A server can limit the total amount of active cookies. In case more cookies are requested, the server will indicate a failure in the response message.

The cookie lifetime shall be 600 seconds. The server has to invalidate any cookie that has exceeded its lifetime and shall not accept any commands for this cookie anymore. The client shall request a new cookie before issuing any new commands.

The cookie response shall indicate whether the server has accepted the cookie request message or occurrence of a failure. If the server has accepted the request, it shall return a cookie within the cookie response message to the client. The client has to provide this cookie in all future command messages. If the response indicates a failure, an exception shall be returned to the client app. The client app shall not be able to send any commands to the sever in case of a failure. The client can request a cookie again after a five seconds timeout.

2.2. Issuing commands

Once the client has an active cookie, it can send arbitrary many command messages to the server as long as the cookie is valid. Any command message shall contain the cookie.

2.2.1. Status command

The status command is used to query the server about the current session status. The server shall return at least the number of successful processed commands for this client and the time to live of the session cookie.

2.2.2. Print command

The print command is used to print messages to the screen of the server.

3. Message Format

All messages are encoded as ASCII text strings. Messages consist of a number of *fields* separated by whitespaces.

The following notational conventions are used in describing the messages:

- String literals are rendered in typewriter text.
- Whitespace is denoted by `<WS>`.
- Fields of a message other than literals are indicated by `<fieldname>`.
- The notation `<msg>` denotes a string of ASCII printable characters **including** whitespaces.
- `[<Fields>]` in square brackets are optional.

3.1. Cookie Request Message

A client requests a cookie by sending a cookie request message to the server. The **cookie request** message has the following format:

```
cp<WS>creq
```

3.2. Cookie Response Message

The **cookie response** message has one of the two following formats:

1. If the server accepts the cookie request, it shall respond with a cookie response message:

```
cp<WS>cres<WS>ACK<WS><cookie>
```

On receipt of this message, the client is allowed to send command messages to the server.

2. If the server does not accept the cookie request, it shall respond with a cookie response message indicating failure. The server should indicate the type of failure using suitable error codes.

```
cp<WS>cres<WS>NAK[<WS><error>]
```

On receipt of this message, the client can restart the cookie request process by sending another cookie request message or abort operation. Requesting a cookie after receipt of a failure message shall be delayed by five seconds. The client shall not send any command message without receiving a cookie first.

3.3. Command Messages

The **command** message has the following format:

```
cp<WS>command<WS><id><WS><cookie><WS><command>[<WS><message>]<WS><checksum>
```

The command message is sent from a client to the server from which the client previously requested a cookie. The fields in the message are defined as follows:

- **id:** Every command is uniquely identified by a command id. The client can start with an arbitrary non-negative id. The id of any message needs to be greater than the id of the previous message. The maximum id is 65535.
- **command:** The command can be either “status” or “print”.
- **cookie:** The cookie is the cookie value received from the server.
- **message:** The message depends on the command the client sends. The message field is omitted for status commands. In case of a print command, the text to print to the server screen is contained in the message field.
- **checksum:** The checksum is used to protect the message from modifications. CP uses CRC (cyclic redundancy check).

3.4. Command response message

The **command response** message has the following format:

```
cp{WS}comres{WS}{id}{WS}{success}{WS}{length} [{WS}{message}] {WS}{crc}
```

The command response message is sent from the server to the client in response to an incoming command message. The fields in the message are defined as follows:

- **id:** The server uses the same id the client used in the command message to match response message to command message. The maximum id is 65535.
- **success:** This can be either “ok” or “error”.
- **length:** The length of the message field.
- **message:** The message depends on the which command the client has issued. For a print command the message field is omitted. For a status command the message field contains at least two lines in JSON formatting for the number of successfully processed command messages and the time-to-live value of the current cookie.
- **checksum:** The checksum is used to protect the message from modifications. CP uses CRC (cyclic redundancy check).