1 Overview

A Command consists of multiple words on a single line and will be specified in the following format:

payload [<block_offset>] put [<payload_offset>]

In this example, 'payload' and 'put' specify the command. Arguments (block_offset and payload_offset in this example) are enclosed in angle brackets and additionally in square brackets, if they are optional.

The responses of the DTN-Daemon are of the form:

<status_code> <message>

If a command was successful, the DTN-Daemon responds with status code 200, which represents API_STATUS_OK. Some commands require additional information to be sent by the client or the server. If the client has to send more information, the server responds with the status code 100 (API_STATUS_CONTINUE) and waits for additional data from the client. Otherwise, if the server has to send additional information, he responds with the status code 200 (API_STATUS_OK) and continues sending the data. In these cases, the command specifies how the data is interpreted and how the end of the data can be recognized.

Offsets into the bundle or data will always start at 0 if not stated otherwise, e. g., block 0 is the first block of the bundle.

After the client established a connection, the DTN-Daemon will send its banner, for example:

IBR-DTN 0.6.4 (build 44739M) API 1.0

To switch to the ExtendedAPI, the client sends the command:

protocol extended

and receives the answer:

200 SWITCHED TO EXTENDED

1.1 Bundle Register

The DTN-Daemon holds a bundle register for the client. All commands starting with bundle and payload apply to the bundle in this register. For example, you can load a bundle into this register with the bundle load command, modify it and commit it for sending.

1.2 Bundle Notification

If the client has its endpoint set or created a registration, the DTN-Daemon will notify him if a bundle arrived for this registration and put it into the clients bundle queue. The notification is of the form:

602 NOTIFY BUNDLE <timestamp> <seq_nr> [<fragment_offset>] <source_eid>

holding all information necessary to identify the bundle unambiguously. Note that this notification can also arrive at the client, between him sending a command and the daemon sending its answer.

1.3 Status Codes

100 API_STATUS_CONTINUE

200 API_STATUS_OK

201 API_STATUS_CREATED

202 API_STATUS_ACCEPTED

302 API_STATUS_FOUND

400 API_STATUS_BAD_REQUEST

401 API_STATUS_UNAUTHORIZED

403 API_STATUS_FORBIDDEN

404 API_STATUS_NOT_FOUND

405 API_STATUS_NOT_ALLOWED

406 API_STATUS_NOT_ACCEPTABLE

409 API_STATUS_CONFLICT

500 API_STATUS_INTERNAL_ERROR

501 API_STATUS_NOT_IMPLEMENTED

503 API_STATUS_SERVICE_UNAVAILABLE

505 API_STATUS_VERSION_NOT_SUPPORTED

2 Commands

2.1 set

set endpoint <endpoint_affix>

Sets the nodes EID to a concatenation of the daemons EID and a given string. Subsequently, if a Bundle arrives for this EID the Client will be notified.

endpoint_affix: a string that will be concatenated to the EID of the DTN-Daemon

2.2 registration

registration add <endpoint>

Creates a registration for a given EID. Subsequently, if a Bundle arrives for this EID the Client will be notified.

endpoint: The EID to subscribe to.

registration del <endpoint>

Deletes a registration for a given EID.

endpoint: The EID to unsubscribe from.

registration list

Sends a list with all registrations of the client.

The first line of the daemons response starts with the API_STATUS_OK code. Subsequently, the daemon sends all of the clients registrations, each on a single line, terminated by an empty line.

Example communication:

```
registration list
200 REGISTRATION LIST
dtn://test/1
dtn://abc
```

---

registration save <time>

Makes a registration persistent for a given time.

The daemon responds with the API_STATUS_OK code, followed by the words 'REGISTRATION SAVE' and a string of printable ASCII characters, that identifies the registration uniquely.

time: The time in seconds that the registration shall be saved.

Example communication:

```
registration save 60
200 REGISTRATION SAVE pEOlcirMMTUVBZs
```

---

registration load <id>

Loads a persistent registration by a given ID. The registration can only be loaded if it is not assigned to another client.

id: The unique ID of the registration, that was returned by 'registration save'.

2.3 neighbor

neighbor list

Sends a list with all neighbors of the DTN-Daemon.

The first line of the daemons response starts with the API_STATUS_OK code. Subsequently, the daemon sends all of its neighbors, each on a single line, terminated by an empty line.

Example communication:

```
neighbor list
200 NEIGHBOR LIST
dtn://test-server
dtn://morpheus
dtn://abc.dtn
```

2.4 bundle

bundle get [<format>]

Sends the content of the bundle register to the client in a chosen format (default: plain). The available data formats are described in section **??**. The daemon responds with the status code API_STATUS_OK on the first line and continues sending the whole bundle in the chosen format.

format: Either 'binary' or 'plain'.

Example communication:

```
bundle get plain
200 BUNDLE GET PLAIN 374770842 4 dtn://test/1
Processing flags: 16
Timestamp: 374770842
Sequencenumber: 4
Source: dtn://test/1
Destination: dtn://test/2
Reportto: dtn:none
Custodian: dtn:none
Lifetime: 3600
Blocks: 1

Block: 1
Flags: LAST_BLOCK
Length: 4

dGVzdA==
```

---

bundle put [<format>]

Receives a bundle from the client in a given format (default: plain) and stores it in the bundle register. The daemon responds with the status code API_STATUS_CONTINUE and awaits a bundle in the given format. The available data formats are described in section **??**. After the whole bundle was transmitted the server sends the status code API_STATUS_OK to signal the success.

format: Either 'binary' or 'plain'.

Example communication:

```
bundle put plain
100 PUT BUNDLE PLAIN
Source: api:me
Destination: dtn://test/2
Processing flags: 16
Blocks: 1

Block: 1
Flags: LAST_BLOCK
Length: 4

dGVzdA==

200 BUNDLE IN REGISTER
```

bundle load queue

Loads a bundle from the queue into the bundle register.

bundle load <timestamp> <seq_nr> [<fragment_offset>] <source_eid>

Loads a bundle, identified by the given parameters, into the bundle register.

timestamp: The timestamp of the bundle to load.

seq_nr: The sequence number.

fragment_offset: The fragment offset.

source_eid: The sender EID.

bundle clear

Clears the bundle register.

bundle free

Removes the bundle in the bundle register from the internal storage and clears the bundle register.

bundle delivered <timestamp> <seq_nr> [<fragment_offset>] <source_eid>

Marks the bundle, identified by the given parameters, as delivered.

timestamp: The timestamp of the bundle.

seq_nr: The sequence number.

fragment_offset: The fragment offset.

source_eid: The sender EID.

bundle store

Stores the bundle in the bundle register in the internal storage.

bundle send

Accept the content of the bundle register to be an outgoing bundle.

bundle info

Send a summary of the bundle to the client. The summary corresponds to the plain format (section **??**) with omitted payload.

bundle block add [<block_offset>]

Adds a new block to the bundle in the bundle register. If no offset is given, the block is appended to the bundle. The daemon responds with the status code API_STATUS_CONTINUE and awaits a bundle in the plain format (section **??**). After the block was transmitted the server sends the status code API_STATUS_OK to signal the success.

block_offset: The offset where to insert the block.

Example communication:

```
bundle block add 0
100 BUNDLE BLOCK ADD
Block: 1
Length: 4

dGVzdA==
```

```
200 BUNDLE BLOCK ADD SUCCESSFUL
```

---

bundle block del <block_offset>

Removes a block from the bundle in the bundle register.

block_offset: The offset of the block that shall be removed.

2.5 payload

payload [<block_offset>] get [<data_offset> [<length>]]

Sends data of a block to a client beginning at <data_offset> (default: 0). If <length> is specified, the number of bytes that are send are limited, otherwise all bytes from <data_offset> to the end of the block are sent. The daemon responds at first with the status code API_STATUS_OK and continues sending the block in question in plain format (section **??**), while header fields except the length field may be omitted. If <data_offset> and/or <length> are given, a modified block is sent, that only hold the requested subset of the data.

block_offset: The offset of the block to get the data from. If omitted, the payload block is used.

data_offset: Offset into the data that shall be returned.

length: Length of the data that shall be returned.

Example communication:

```
payload 0 get 1 2
200 PAYLOAD GET
Length: 2

ZXM=
```

---

payload [<block_offset>] put [<data_offset>]

Writes data into a block of the bundle register, beginning at offset <data_offset> (default: 0). The server responds with the status code API_STATUS_CONTINUE and awaits a single Block in plain format (section **??**), while header fields except the length field may be omitted. On success, the server responds with the status code API_STATUS_OK.

block_offset: The offset of the block to write the data into. If omitted, the payload block is used.

data_offset: Offset where to write the new data.

Example communication:

```
payload 0 put 3
100 PAYLOAD PUT
Block: 1
Length: 2

ZXM=

200 PAYLOAD PUT SUCCESSFUL
```

---

payload [<block_offset>] append

Appends data to a block of the bundle register. The server responds with the status code API_STATUS_CONTINUE and awaits a single Block in plain format (section **??**). On success, the server responds with the status code API_STATUS_OK.

block_offset: The offset of the block to append the data to. If omitted, the payload block is used.

Example communication:

```
payload 0 append
100 PAYLOAD APPEND
Block: 1
Length: 2

ZXM=

200 PAYLOAD APPEND SUCCESSFUL
```

field specifier	value
Processing flags	unsigned decimal number
Timestamp	unsigned decimal number
Sequencenumber	unsigned decimal number
Source	EID
Destination	EID
Reportto	EID
Custodian	EID
Lifetime	unsigned decimal number
Fragment offset (optional)	unsigned decimal number
Application data length (optional)	unsigned decimal number

Table 1: Field specifiers for the header of the primary bundle block.

---

payload [<block_offset>] clear

Clears the data of a block of the bundle register.

block_offset: The offset of the block that shall be cleared. If omitted, the payload block is used.

payload [<block_offset>] length

Returns the length of a block of the bundle register..

block_offset: The offset of the block whose length shall be returned. If omitted, the payload block is used.

Example communication:

```
payload 0 length
200 PAYLOAD LENGTH
Length: 7
```

---

field specifier	value
Block (the block type)	unsigned decimal number
Flags (optional)	space seperated flag specifier according to table **??**
EID (optional, can appear multiple times)	EID
Length	unsigned decimal number

Table 2: Field Specifiers for the header of other blocks.

- LAST_BLOCK
- REPLICATE_IN_EVERY_FRAGMENT
- TRANSMIT_STATUSREPORT_IF_NOT_PROCESSED
- DELETE_BUNDLE_IF_NOT_PROCESSED
- DISCARD_IF_NOT_PROCESSED
- FORWARDED_WITHOUT_PROCESSED

Table 3: Specifier for the flags field of a block.

3 Data Formats

3.1 Plain Format

The plain data format is used to either transmit a whole bundle, the bundle with omitted payloads or single blocks. To transmit a bundle, all blocks are transmitted sequentially, starting with the primary bundle block. Each block consists of the block header, an empty line and its payload, followed by an additional empty line to mark its end. An exception to this is only the primary bundle block, which has no payload. If the payload for a block is not to be transmitted (in case of omitted payloads or for the primary bundle block), the payload and the second empty line are left out. The end of a bundle transmission is detected if the last processed block has its 'last block' flag set.

The header transmission consists of lines of the form:

<field>: <value>

where <field> specifies a field of the blocks header. The field specifiers for the primary bundle block are listed in table **??**, while the specifiers for other blocks are shown in table **??**. Field specifiers may appear in any order and some specifiers may be omitted. If the client sends 'api:me' as an EID parameter, it will be replaced with the clients endpoint when 'bundle send' is called.

The block payloads are base64 encoded and can occupy multiple lines.

Example for a bundle transmission:

```
Processing flags: 16
```

```
Timestamp: 375298751
Sequencenumber: 3
Source: dtn://tsrtp/1
Destination: dtn://tsrtp/2
Reportto: dtn:none
Custodian: dtn:none
Lifetime: 3600
Blocks: 1

Block: 1
Flags: LAST_BLOCK
Length: 64

dGVzdHRlc3R0ZXN0dGVzdHRlc3R0ZXN0dGVzdHRlc3R0ZXN0dGVzdHRlc3R0ZXN0
dGVzdHRlc3R0ZXN0dGVzdA==
```

Example for a bundle transmission with omitted payloads:

```
Processing flags: 16
Timestamp: 375298751
Sequencenumber: 3
Source: dtn://tsrtp/1
Destination: dtn://tsrtp/2
Reportto: dtn:none
Custodian: dtn:none
Lifetime: 3600
Blocks: 1

Block: 1
Flags: LAST_BLOCK
Length: 64
```

Example for the transmission of a single block:

```
Block: 1
Flags: LAST_BLOCK
Length: 64

dGVzdHRlc3R0ZXN0dGVzdHRlc3R0ZXN0dGVzdHRlc3R0ZXN0dGVzdHRlc3R0ZXN0
```

```
dGVzdHRlc3R0ZXN0dGVzdA==
```

3.2 Binary Format

The binary format corresponds to the format used by the TCP convergence layer and is not specified further in this document.