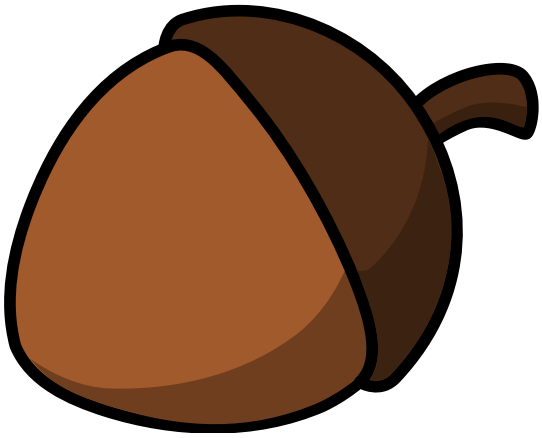

Denobo

Denobo Message Transfer Protocol (DMTP)
Version 0.9.0.1 (Alpha)

Preface – Intention

The intention of DMTP is to provide a means of transmitting serialised message data peer-to-peer between network-enabled actors in the Denobo multi-agent middleware. The protocol should support encryption and compression of packet payloads, or provide room to extend into these areas in a later version of the software. The protocol should also allow for the transmission of class files for dynamic loading by peers and implement security features to address the inherent security concerns involved with this.

As of this version of DMTP, only basic support is provided for encryption. Dynamic object loading is not implemented nor are any codes reserved for its implementation. **In short, software implementing this iteration of DMTP should not be considered a secure platform or be used for critical systems.**

Chapter II - Packet Codes

Packet codes represent possible values for the packet-code header in the message packet and are transmitted as a sequence of 3 ASCII digits (0-9).

1XX - Handshake/Initialisation Codes

Codes 100 – 199 (inclusive) are reserved for messages relating to connection handshaking and the transmission of metadata between connection peers.

100: GREETINGS

Packet code 100 is the GREETINGS code, and is sent from the peer initiating the connection (i.e. the connecting peer) to the one receiving the connection (the receiving peer) to identify itself as a Denobo peer node at the start of a peer-to-peer session.

Example Packet

```
DENOBO v0.9 (BENSON)
packet-code:100
body-length:0
```

Possible Responses

The connecting peer should be prepared for the following responses from the receiving peer:

- 101 (ACCEPTED): Peer accepted and session clear to begin.
- 102 (CREDENTIALS_PLZ): Authentication is required to access this peer. Connecting peer should reply with a 103 (CREDENTIALS) packet containing a username and password. Receiving peer should terminate the connection if a 103 packet is not provided within a timeout.
- 400 (NO): Peer rejected the connection without specifying why.
- 401 (TOO_MANY_PEERS): Peer rejected the connection because its maximum number of peers has been reached.
- 402 (NOT_A_SERVER): Peer rejected the connection because it is not configured to act as a receiving peer, only a connecting peer.

101: ACCEPTED

Packet code 101 is the ACCEPTED code, and is sent from the receiving peer to the connecting peer in response to a 100 (GREETINGS) or 103 (CREDENTIALS) packet to confirm that the session is clear to begin. Connecting peers receiving the 101 status code may begin transmitting 3XX packets containing live session data. Handshaking is considered to have ended after a 101 code is transmitted.

Example Packet

DENOBO v0.9 (BENSON)
packet-code:101
body-length:0

Possible Responses

- A packet with any live session code (3XX), session modification code (2XX) or error code (4XX) can be received directly following the sending of a 101 code. No Handshake/Initialisation codes (1XX) should be expected as a 101 code signals the end of handshaking and initialisation.

102: CREDENTIALS_PLZ

Packet code 102 is the CREDENTIALS_PLZ code, and is sent from the receiving peer to the connecting peer in response to a 100 (GREETINGS) packet to ask for a username and password as an additional authentication measure. The receiving peer is preconfigured with a bank of username/password hash pairs to authenticate against, which can be loaded from any data source.

Example Packet

DENOBO v0.9 (BENSON)
packet-code:102
body-length:0

Possible Responses

- 103 (CREDENTIALS): Connecting peer is offering the username and password contained in the packet body as authentication.
- 403 (NO_CREDENTIALS): Connecting peer does not have any credentials to offer and has terminated the session.
- No response after a reasonable timeout should be treated as a 403 packet.

103: CREDENTIALS

Packet code 103 is the CREDENTIALS code, and is sent from the connecting peer to the receiving peer in response to a 102 (CREDENTIALS_PLZ) packet to offer up its username and password as authentication. Username and password are transmitted in the clear in query string format.

Example Packet

```
DENOBO v0.9 (BENSON)
packet-code:101
body-length:25
username=foo&password=bar
```

Possible Responses

- 101 (ACCEPTED): Peer accepted and session clear to begin.
- 400 (NO): Receiving peer refused the connection without specifying why.
- 404 (BAD_CREDENTIALS): Receiving peer refused the connection because the credentials provided in the 103 packet were not acceptable.

2XX Session Modification Codes

Codes 200-299 (inclusive) are reserved for messages relating to the modification of the session state.

200: CHANGE_CONFIRMED

Packet code 200 is the CHANGE_CONFIRMED code, and can be sent from either peer in response to a 2XX code it receives from the other. Receiving a 200 code in reply to a 2XX code indicates that the receiving node is now operating in the same session state as the originator.

Example Packet

DENOBO v0.9 (BENSON)
packet-code:200
body-length:0

Possible Responses

- A packet with any live session code (3XX), session modification code (2XX) or error code (4XX) can be received directly following the sending of a 200 code. No Handshake/Initialisation codes (1XX) should be expected as handshaking and initialisation is already over.

201: SET_COMPRESSION

Packet code 201 is the SET_COMPRESSION code, and can be sent from either peer at any point in the session. Upon receipt of a 200 code, a peer should switch the compression algorithm it uses to compress packet payloads to the one specified in the message body.

Example Packet

DENOBO v0.9 (BENSON)
packet-code:201
body-length:[compression-technique-name-length]
[compression-technique-name]

Supported Compression Techniques

The following compression techniques are currently officially supported, see “Compression Techniques” chapter for more details:

- BASIC – Simple semi-adaptive Huffman coding based on byte frequencies with an in-built BSD checksum to verify data integrity.
- LZW – Lempel-Ziv-Welch compression, also with BSD checksum.

Possible Responses

- 200 (CHANGE_CONFIRMED): Recipient peer has changed its compression algorithm to the one specified.
- 400 (NO); Recipient peer refused to change compression algorithm without specifying why.
- 405 (UNSUPPORTED): Recipient peer does not support the proposed change in compression algorithm.

202: BEGIN_SECURE

Packet code 202 is the BEGIN_SECURE code, requesting that the receiving peer switch to using encrypted packets. The packet body will consist of a standardised (that is, pre-shared) large prime [1] to the power of a secret, randomly chosen large integer. This will allow for establishment of a shared secret key between the two connecting peers via a Diffie-Hellman key exchange.

Example Packet

DENOBO v0.9 (BENSON)
packet-code:202
body-length:[large-integer-length]
[large-integer]

Possible Responses

It is worth noting that the 202 (BEGIN_SECURE) code is the only 2XX packet code that should never receive a 200 (CHANGE_CONFIRMED) as a direct response.

- 203 (CONFIRM_SECURE): Recipient peer agrees with the proposed change in security state and is returning its contribution to the Diffie-Hellman shared secret key.
- 400 (NO): Recipient peer refused to change session state without specifying why. Session should revert to unencrypted state.
- 405 (UNSUPPORTED): Recipient peer does not support the proposed change in security state. Session should revert to unencrypted state.

203: CONFIRM_SECURE

Packet code 203 is the CONFIRM_SECURE code, instructing the receiving peer that it received and accepted a transmitted 202 BEGIN_SECURE code and is returning its contribution to the shared secret key as the packet body.

Example Packet

DENOBO v0.9 (BENSON)
packet-code:203
body-length:[large-integer-length]
[large-integer]

Possible Responses

- 200 (CHANGE_CONFIRMED): The shared secret key has been calculated and transmission of encrypted data may begin.
- 400 (NO): The session change has been abandoned but the originator did not specify why.

204: END_SECURE

Packet code 204 is the END_SECURE code, informing the receiving peer that the originator is about to stop transmitting encrypted packets and begin transmitting in plaintext again. This signals the receiver to discard its secret key. The sender however **should not discard their secret key** until they receive a 200 (CHANGE_CONFIRMED) code from the recipient.

Example Packet

DENOBO v0.9 (BENSON)
packet-code:204
body-length:0

Possible Responses

- 200 (CHANGE_CONFIRMED): Recipient peer understands that transmitted packets should no longer be encrypted and has discarded its secret key.
- 400 (NO): Recipient peer refuses to terminate the encrypted session state but did not specify why.

3XX Message Transmission/Agent Query Codes

Codes 300-399 (inclusive) are reserved for messages relating to message transmission and peer-to-peer queries.

300: PROPAGATE

Packet code 300 is the PROPAGATE code, instructing the receiving peer that the message contained in the packet body is propagating through the network. The receiving peer should pass the message to any attached handlers if and only if it is addressed to it before broadcasting it to any other connected peers **excluding** the originator.

Example Packet

DENOBO v0.9 (BENSON)
packet-code:300
body-length:[message-length]
[message]

Possible Responses

No direct response is expected to a 300 (PROPAGATE) packet.

301: POKE

Packet code 301 is the POKE code, instructing the receiving peer to reply with another 301 packet. The sole purpose of this is to establish whether or not the recipient agent is healthy and responding.

Example Packet

DENOBO v0.9 (BENSON)
packet-code:301
body-length:0

Possible Responses

- 301 (POKE): The agent replied to the poke with a poke of its own, indicating that the connection and agent are still healthy.

4XX Error Codes

Codes 400-499 (inclusive) are reserved for messages relating to errors encountered by either peer in response to a request they receive. Direct responses are never expected to any of these codes and therefore to the “Possible Responses” section has been omitted.

400: NO

A generic error was encountered by the peer. Peers should avoid using this code whenever a more specific error code is available, but **must** transmit an error code of some kind whenever a request is received that they are unable to deal with. The body of the message may contain additional information about the exact nature of the error.

Example Packet

```
DENOBO v0.9 (BENSON)
packet-code:400
body-length:0
```

401: TOO_MANY_PEERS

The peer is configured with a peer limit and that limit has been reached. A 401 code is sent in response to a 100 (GREETINGS) code rejecting the connection on this basis.

Example Packet

```
DENOBO v0.9 (BENSON)
packet-code:401
body-length:0
```

402: NOT_A_SERVER

The peer is configured to be able to make connections, but not accept them (i.e. this peer can only connect to peers, not receive connections from others). A 402 code is sent in response to a 100 (GREETINGS) code rejecting the connection on this basis.

Example Packet

```
DENOBO v0.9 (BENSON)
packet-code:402
body-length:0
```

403: NO_CREDENTIALS

The peer has not been configured with a username and password and as such is unable to provide the credentials requested. A 403 code is sent in response to a 102 (CREDENTIALS_PLZ) code acknowledging that it has no credentials to provide before closing the connection.

Example Packet

DENOBO v0.9 (BENSON)
packet-code:403
body-length:0

404: BAD_CREDENTIALS

The peer is rejecting the username and password passed to it in a 103 (CREDENTIALS) packet because they are incorrect, badly-formed or otherwise unacceptable. The connection will be closed.

Example Packet

DENOBO v0.9 (BENSON)
packet-code:404
body-length:0

405: UNSUPPORTED

The peer is rejecting the session state change proposed by a 2XX (session modification) code because it has no support for it. The peer that sent the 2XX code is now responsible for deciding whether or not to terminate the connection.

Example Packet

DENOBO v0.9 (BENSON)
packet-code:405
body-length:0

Chapter III – Compression Techniques

Denobo officially supports a number of compression techniques for use with the 201 (SET_COMPRESSION) packet code. This chapter specifies in detail the format of each.

Because the 201 packet is expected to have been transmitted beforehand, it is assumed that both peers know which compression technique is being used at any one time. Therefore, no identification bytes or 'magic numbers' are used to mark compressed data as using one compression technique or another.

Compression Technique 0: BASIC

Simple semi-adaptive Huffman coding based on byte frequencies. Compression takes place in stages:

- One pass is done on the data to collect byte frequency values.
- A Huffman tree is generated from these frequencies.
- Each leaf on the Huffman tree is back-traced to the root, giving a prefix code.
- Each prefix code is added as an entry in a lookup table.
- The lookup table is used to translate bytes into prefix codes, which are concatenated to create a compressed set of bytes.
- This set of bytes represents the compressed payload.

Additional metadata must be included alongside the compressed payload in order to enable its decompression by the receiving peer. Included metadata follows:

- A BSD checksum byte of the uncompressed data to verify data integrity.
- 4 bytes containing the length of the compressed data **in bits** (the compressed payload is not necessarily a discrete number of bytes in length).
- A serialised representation of the prefix code table (necessary for translation of bit codes back to bytes).

Formal Format Specification

A more detailed description of the format of a compressed BASIC data packet follows:

- **0x00:** BSD checksum byte.
- **0x01-0x04:** Length of compressed data in bits.
- **0x05-N:** Table entries of serialised prefix code table.
    - **Offset 0:** A zero byte (0x00) or a 255 byte (0xFF) if the entry is the last one in the table.
    - **Offset 1:** The value of the byte the prefix code represents.
    - **Offset 2:** The length of the prefix code **in bits**.
    - **Offset 3+**: The prefix code itself.
- **N**-*: The compressed payload.

Compression Technique 1: LZW

Compression is performed using the Lempel-Ziv-Welch algorithm. Compression takes place incrementally. As this is a well-known algorithm that is already specified, it will not be discussed in detail here. What will be discussed, however, is the precise format in which it is transmitted.

As it is possible to decompress LZW-coded data directly from compressed data no dictionary or lookup table of any kind is included with the transmitted payload. The receiving peer is expected to incrementally reconstruct the encoding dictionary directly from the compressed data stream.

For the sake of data integrity, a BSD checksum byte of the **uncompressed** data is prepended to the **beginning** of the compressed data packet. This byte is for verification purposes only and **should not** be read as part of the compressed data.

Formal Format Specification

A more detailed description of the format of a compressed LZW data packet follows:

- 0x00: BSD checksum byte.
- 0x01-*: The compressed payload.

Appendix

[1] - 1536-bit MODP Group (Taken from RFC 3526)

The 1536 bit MODP group has been used for the implementations for quite a long time, but was not defined in RFC 2409 (IKE). Implementations have been using group 5 to designate this group, we standardize that practice here.

The prime is: 2^1536 - 2^1472 - 1 + 2^64 * { [2^1406 pi] + 741804 }

Its hexadecimal value is:

```
FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D
C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F
83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
670C354E 4ABC9804 F1746C08 CA237327 FFFFFFFF FFFFFFFF
```

The generator is: 2.