Bid Emission and Exchange Protocol

Émmanuel Aacha, Vincent Guenat, Émeric Tosi

12 février 2015

Sommaire

1 Introduction **3**
- 1.1 Purpose . . . 3
- 1.2 Requirements . . . 3
- 1.3 Terminology . . . 3
- 1.4 Overall Operation . . . 5

2 Notational Conventions and Generic Grammar : Basic Rules **6**

3 Protocol Parameters **7**
- 3.1 Date/Time Formats . . . 7
- 3.2 Character Sets . . . 7

4 BEEP Message **8**
- 4.1 Message Types . . . 8
- 4.2 Message Headers . . . 8
- 4.3 Message Body . . . 8
- 4.4 Message Length . . . 8

5 Request **9**
- 5.1 Request-Line . . . 9

6 Response **10**
- 6.1 Status-Line . . . 10
- 6.2 Response Header Fields . . . 10

7 Entity **12**
- 7.1 Type . . . 12
- 7.2 Entity Length . . . 12

8 Connections **13**
- 8.1 Persistent Connections . . . 13
- 8.2 Message Transmission Requirements . . . 13

9 Method Definitions **14**
9.1 GET 14
9.2 PUT 14
9.3 DELETE 14
9.4 CONNECT 15

10 Status Code Definitions **16**
10.1 Positive (X) 16
10.2 Negative (-X) 16

11 Access Authentication **18**

12 Security Considerations **19**
12.1 Personal Information 19
12.2 Privacy Issues Connected to Accept Headers 19
12.3 Authentication Credentials and Idle Clients 19

13 Acknowledgments **20**

14 References **21**

15 Full Copyright Statement **22**

Chapitre 1

Introduction

1.1 Purpose

The Bid Emission and Exchange Protocol is a protocol for a distributed bid systems. It enables users using the client to bid for objects among a list. This protocol is referred to as the BEEP/0.1a.

1.2 Requirements

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 [34].

An implementation is not compliant if it fails to satisfy one or more of the MUST or REQUIRED level requirements for the protocols it implements. An implementation that satisfies all the MUST level requirements but not all the SHOULD level requirements for its protocols is said to be "conditionally compliant" ; one that satisfies all the MUST or REQUIRED level and all the SHOULD level requirements for its protocols is said to be "unconditionally compliant" ;

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 [34].

1.3 Terminology

This specification uses a number of terms to refer to the roles played by participants in, and objects of, the BEEP communication.

connection A transport layer virtual circuit established between two programs in order to establish a communication.

login IDs A combination of login and password.

message The basic unit of BEEP communication, consisting of a structured sequence of octets matching the syntax defined in section 4 and transmitted via the previously established connection.

request A BEEP request message, as defined in section 5.

response A BEEP response message, as defined in section 6.

resource A network data object or service that can be identified by a URI, as defined in section 3.2. Resources may be available in multiple representations (e.g. multiple languages, data formats, size, and resolutions) or vary in other ways.

entity The information transferred as the payload of a request or response. An entity consists of meta-information in the form of entity-header fields and content in the form of an entity-body, as described in section 7.

representation An entity included with a response that is subject to content negotiation, as described in section 12. There may exist multiple representations associated with a particular response status.

content negotiation The mechanism for selecting the appropriate representation when servicing a request, as described in section 12. The representation of entities in any response can be negotiated (including error responses).

client A program that establishes connections for the purpose of sending requests.

server An application program that accepts connections in order to service requests by sending back responses. Any given program may be capable of being both a client and a server ; our use of these terms refers only to the role being performed by the program for a particular connection, rather than to the program's capabilities in general. Likewise, any server may act as an origin server, proxy, gateway, or tunnel, switching behavior based on the nature of each request.

origin server The server on which a given resource resides or is to be created.

semantically transparent A cache behaves in a "semantically transparent" manner, with respect to a particular response, when its use affects neither the requesting client nor the origin server, except to improve performance. When a cache is semantically transparent, the client receives exactly the same response (except for hop-by-hop headers) that it would have received had its request been handled directly by the origin server.

upstream/downstream Upstream and downstream describe the flow of a message : all messages flow from upstream to downstream.

1.4 Overall Operation

BEEP is a request/response protocol. The client send a request to the server and the server answer with a response. All the messages (requests and responses) go through a connection between the server and the client. The connection MUST be established before any request other than those using the CONNECT method occurs. The request is composed of a method, an entity and the protocol version.

The response is composed of a status-line, response-headers and in some cases an entity.

Chapitre 2

Notational Conventions and Generic Grammar : Basic Rules

The following rules are used throughout this specification to describe basic parsing constructs. The US-ASCII coded character set is defined by ANSI X3.4-1986 [21].

OCTET = ¡any 8-bit sequence of data¿

CHAR = ¡any US-ASCII character (octets 0 - 127)¿

DIGIT = ¡any US-ASCII digit "0".."9"¿

INT = ¡basic signed integer type. cf ISO/IEC 9899¿

LONG INT = ¡long signed integer type. cf ISO/IEC 9899¿

FLOAT = ¡floating-point type. cf IEEE 754¿

SP = ¡US-ASCII SP, space (32)¿

CR = ¡US-ASCII CR, carriage return (13)¿

LF = ¡US-ASCII LF, linefeed (10)¿

LWS = ¡US-ASCII LWS, linear white space (9)¿

¡"¿ = ¡US-ASCII double-quote mark (34)¿

Chapitre 3

Protocol Parameters

3.1 Date/Time Formats

Full Date This application allows only one full date format : 24-11-2014 14 :41 :12 GMT This is a format from the RFC 3339.

Delta Seconds Delta Seconds represents a difference between server's starting time and current time.

3.2 Character Sets

BEEP uses the same definition of the term "character set" as that described for BEEP : The term "character set" is used in this document to refer to a method used with one or more tables to convert a sequence of octets into a sequence of characters. Note that unconditional conversion in the other direction is not required, in that not all characters may be available in a given character set and a character set may provide more than one sequence of octets to represent a particular character. This definition is intended to allow various kinds of character encoding, from simple single-table mappings such as US-ASCII to complex table switching methods such as those that use ISO-2022's techniques. However, the definition associated with a BEEP character set name MUST fully specify the mapping to be performed from octets to characters. In particular, use of external profiling information to determine the exact mapping is not permitted.

Chapitre 4

BEEP Message

4.1 Message Types

BEEP messages consist of requests from client to server and responses from server to client.

Request (section 5) and Response (section 6) messages use a generic message format for transferring entities (the payload of the message). Field names are case-sensitive. The field value MAY be preceded by any amount of LWS, though a single SP is preferred.

4.2 Message Headers

Headers are only response-headers. Headers end with a single CR. Each header field consists of a name followed by a colon (" :") and the field value. Header fields ends with a semi-colon (" ;"). The headers are case-sensitive.

4.3 Message Body

The message-body is used to transfer informations associated with a request or a response. The message body ends with a double CR.

4.4 Message Length

The message-length is the length of the headers if there is any plus the length of the message-body (cf entity section (8) for the length of an entity).

Chapitre 5

Request

5.1 Request-Line

The Request-Line begins with a method keyword, followed by the entity on which is applied this method.

The Method keyword indicates the method to be used on the resource identified by the Request-entity. The method is case-sensitive.

Method = "GET" cf. section 9.1

— "PUT" cf. section 9.2

— "DELETE" cf. section 9.3

— "CONNECT" cf. section 9.4

Chapitre 6

Response

After receiving and interpreting a request message, a server responds with an BEEP response message.

6.1 Status-Line

The first line of a Response message is the Status-Line, consisting of the protocol version followed by a numeric status code and its associated textual phrase, with each element separated by SP characters. No CR is allowed except in the final CR sequence.

Status-Line = BEEP-Version SP Status-Code SP Reason-Phrase CR

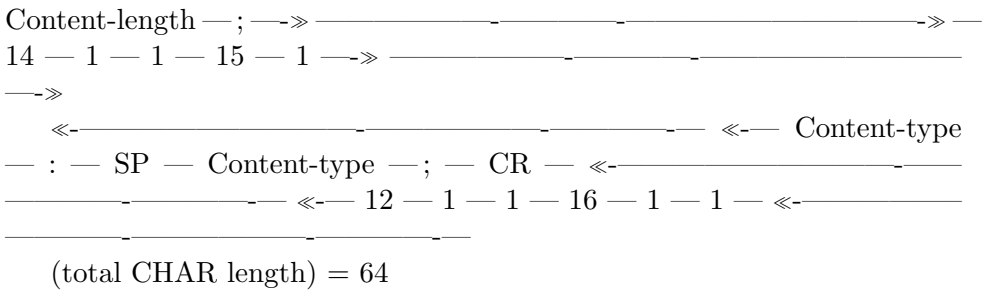
Status Code and Reason Phrase The Status-Code element is a signed 1-digit integer. This is the result code of the attempt to understand and satisfy the request. These codes are fully defined in section 10. The Reason-Phrase is intended to give a short textual description of the Status-Code. The Status-Code is intended for use by automate and the Reason-Phrase is intended for the human user. The client is not required to examine or display the Reason-Phrase.

Status-line length The CHAR length of the status-line is described below : ——————-———————-— — Status Code — SP — Reason Phrase — CR — ——————-———————-— — 2 — 1 — 12 — 1 — ——————-———————-—

(total CHAR length) = 16

6.2 Response Header Fields

The CHAR length of the response header line is described below : ——— ———-————-———————-≫ — Content-length — : — SP —



(total CHAR length) = 64

Content-Length It's the response body length.

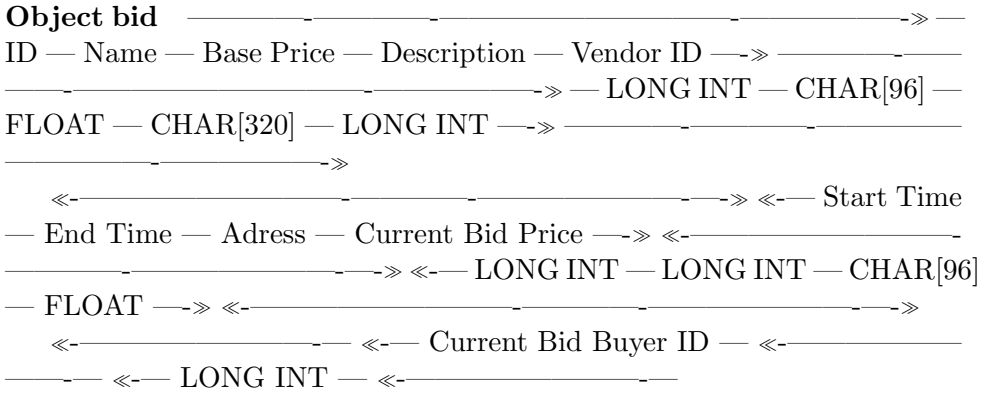
Content-Type It's the type of entity of the body (cf section 7).

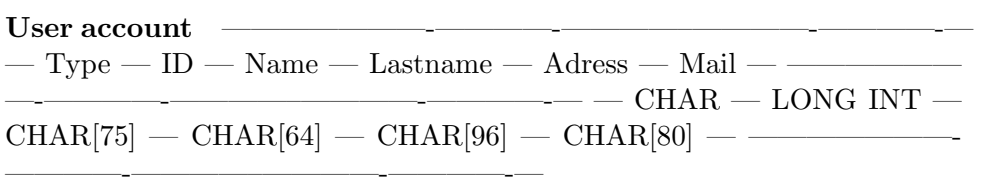
Chapitre 7

Entity

7.1 Type

Entity used by BEEP are composed of basic C vars.





7.2 Entity Length

Object bid An object bid is (in OCTET) 4 + 96 + 4 + 320 + 4 + 4 + 96 + 4 + 4 = 536

User account An user account is (in OCTET) 1 + 4 + 75 + 64 + 96 + 80 = 320

Chapitre 8

Connections

8.1 Persistent Connections

Purpose The purpose of the persistent connection is for the client to connect only once. That way it will reduce the number CONNECT requests, and clients don't have to send headers with their credentials for each request they made.

Overall Operation The client send a request using the CONNECT method with its login and password, if there is any, or a PUT request to create a user account. In the first case, CONNECT is used and the pair ID/password is correct. After that, the client is allowed to use the command to add, delete or bid for an object or to obtain the objects's list. If not, the client SHOULD use an other ID/password couple to connect or use the PUT request to create an account.

8.2 Message Transmission Requirements

One (or more) network connection is required if the server and clients don't run on the same system. BEEP also require the choosen connexion port availability.

Client Behavior if Server Prematurely Closes Connection If the server closes the connection, the client SHOULD shut down either. After that the client CAN restart its use.

Chapitre 9

Method Definitions

9.1 GET

The GET method enable the client to ask the server to send back an entity. This method requires X parameters and have the following length : ——————-——————-——————-— — GET — SP — OBJECT —; — CR — ——————-——————-——————-— — 3 — 1 — OBJECT — 1 — 1 — ——————-——————-——————-—

(total CHAR length) = 6 + (Object length)

9.2 PUT

The client requests the server to store the a requested entity sent in the core of the request. This method requires X parameters and have the following length : ——————-——————-——————-— — PUT — SP — OBJECT —; — CR — ——————-——————-——————-— — 3 — 1 — OBJECT — 1 — 1 — ——————-——————-——————-—

(total CHAR length) = 6 + (Object length)

9.3 DELETE

The client requests the server to delete the requested entity from his database. For the entity types, cf section 7. To delete an object or a user account, the client must be authenticed as the object's creator or an administrator. This method requires one parameters and have the following length : ——————-——————-——————-— — DELETE — SP — OBJECT —; — CR — ——————-——————-——————-— — 6 — 1 — OBJECT — 1 — 1 — ——————-——————-——————-—

(total CHAR length) = 9 + (Object length)

9.4 CONNECT

The CONNECT method is used by the client to establish a permanent connection with the server. This method requires two parameters and have the following length : ——————-———————————-—— CONNECT — SP — LOGIN —; — PASSWORD —; — CR — ———————-———————————-— — 7 — 1 — 27 — 1 — 26 — 1 — 1 — ——————-———————————-——

(total CHAR length) = 64

Chapitre 10

Status Code Definitions

The status code is the first part of the server answer for a request.

10.1 Positive (X)

0 OK The request has succeeded. The information returned with the response is dependent on the method used in the request, for example : — GET an entity corresponding to the requested resource — is sent in the response ; — — PUT an entity-header with the return status ; — — DELETE an entity-header with the return status ; — — CONNECT an entity-header with the return status ;

1 Created Result of a successful account or object bid creation.

10.2 Negative (-X)

-1 Bad Request The request could not be understood by the server due to malformed syntax. The client SHOULD NOT repeat the request without modifications.

-2 Not Created The request could not be understood by the server due to a problem with the ressource designed by the request. Result of a failed account or object bid creation.

-3 Internal Server Error The server encountered an unexpected condition which prevented it from fulfilling the request. Result of an unknown, unpredictable or internal (core) server error.

-4 Conflict Result of a concurrent object bid read and write access between more than one user and/or administrator.

-5 Forbidden The server understood the request, but is refusing to fulfill it. Authorization will not help and the request SHOULD NOT be repeated. Result of giving invalid connection ids, or call a method which needs rights but without them.

Chapitre 11

Access Authentication

It require to use the CONNECT method with the user's login IDs. The server sends back a response message composed of the status line.

Chapitre 12

Security Considerations

At the moment, no encryption module is in use, all the content is sent clearly through the Internet. Please be careful while using this version of the protocol. You MUST consider these severe security issues !

12.1 Personal Information

Your login, your password, and any informations you could send CAN be viewed by anyone who is watching your communications under this protocol.

12.2 Privacy Issues Connected to Accept Headers

.....

12.3 Authentication Credentials and Idle Clients

At the moment, all your credentials pass through one or more networks without protections. Anyone can take them from the BEEP connect request and/or from the answer of a GET request of user informations.

Chapitre 13

Acknowledgments

This specification makes heavy use of the generic definitions as they can be seen in the RFC 2616. We would like to thanks the author to allow us to use those definitions.

Moreover, we want to thanks our professors P. Torguet and A. Aoun for all their teaching and advising.

Chapitre 14

References

[1] R. Fielding, UC Irvine, J. Gettys, Compaq/W3C, J. Mogul, Compaq, H. Frystyk, W3C/MIT, L. Masinter, Xerox, P. Leach, Microsoft, T. Berners-Lee, W3C/MIT, "Hypertext Transfer Protocol", RFC 2616, June 1999.

Chapitre 15

Full Copyright Statement

For the general definitions and some help in the structuration of our rfc, we used the rfc 2616 (HTTP/1.1), as it is authorized by the copyright statement below.