GiJo Chat Protocol

Protocol Design & Implementation
COMP 3274
Set O

Prepared By:
Jordan Godau
Giwoun Bae

Prepared For:
D’Arcy Smith, MSc

Dec 3, 2021

Table of Contents

1.0 Executive Summary

The GiJo protocol was designed for transmitting messages between mutually exclusive client based applications over a network. This document shall describe the particular details of the protocol including the format, general usage, examples, as well as the purpose of certain design choices.

In general, the GiJo protocol has been designed with the basic responsibilities of most application layer protocols:

- Writing data from the network,
- Receiving data from other network elements
- Defining an interface for chat-based servers
- Error handling between multiple network elements.

While the current protocol has been designed to support text-based messages only, it is anticipated that future versions will accommodate different data objects such as imagery, audio, and other various file formats. Should this event occur, this document shall be updated to reflect the changes of the protocol.

2.0 Protocol Limitations

In its current state, the GiJo protocol should be explicitly used for transferring of simple ascii alpha-numeric messages. This recommendation should persist until more research has been completed by the authors to determine what types of support/semantics should be used to handle more complicated or advanced features in a protocol.

For instance, drastic changes may be necessary to accommodate handling many concurrent client requests to a server implementing GiJo - something that is (at this moment) known, but was not considered in the creation of the protocol.

Rather than present a protocol that ventures into the unknown, instead this protocol presents a simple framework for a more sophisticated future version of itself protocol - without compromising the functionality needed to implement the protocol in its current state.

3.0 Protocol

The following section describes the various components of the GiJo protocol. This includes the protocol message format, client/server specific behaviour, restrictions, constraints, and limitations. Additionally, examples shall be included to demonstrate and reinforce the intended use of the protocol.

3.1 Protocol Format

The following tables depict the protocol message header for GiJo, along with a description of it’s individual fields:

GiJo Message Header Block		
CMD	**SRCID**	**DESTID**
ENCODING	**LENGTH**	
CONTENT		

FIELD	**DESCRIPTION**	**SIZE (BYTES)**
CMD:	Used for providing instructions to the server.	8
SRCID:	ID assigned to the sender of the message	4
DSTID:	ID assigned to the intended recipient of the message	4
ENCODING:	Encoding type of the message (e.g. ASCII)	8
LENGTH:	The length of the transmitted messages in bytes	4
CONTENT:	The message being transmitted	512

3.2 Header Fields

The following Sections describe the protocol fields in greater details. This includes the range of specified values, as well as examples to demonstrate the intended use of the protocol message.

3.2.1 CMD Field

The command field is designated for providing instructions to a chat-based server. These instructions shall be sent from the client to the server, and shall never be altered by the server once the message has been transmitted from the client. It should be mentioned that the proposed CMD options shall be prescriptive in their instruction, however the implementation is up to the software designer. In essence, the *semantics* of the CMD shall be followed, however the exact way these instructions are carried out should be done in a way that best fits the software architecture.

At time of writing this document, three primary commands are proposed, which are described below:

CMD	DESCRIPTION
SEND:	Shall initiate transmission to the message to the <DSTID>
LOGOUT:	Shall disconnect the <SRCID> client from the network application Transmission of this message should contain no bytes written to the CONTENT field. Additionally, transmission of the LOGOUT CMD shall set the following fields to 0: - <DSTID> - <LENGTH>
LOGIN:	Shall connect the <SRCID> client to the network application Transmission of this message should contain no bytes written to the CONTENT field. Additionally, transmission of the LOGOUT CMD shall set the following fields to 0: - <DSTID> - <LENGTH>

SEND EXAMPLE

```
"SEND" SP SRCID SP DSTID CRLF ENCODING SP LENGTH CRLF CRLF CONTENT
```

Example is:

```
     SEND 234 223
     ASCII 31

     THIS IS THE CONTENT AS EXAMPLE.
```

LOGOUT EXAMPLE

```
"LOGOUT" SP SRCID SP 0 CRLF 0 SP 0 CRLF CRLF
```

Example is:

```
     LOGOUT 244 0
     0 0
```

LOGIN EXAMPLE

```
"LOGIN" SP SRCID SP 0 CRLF 0 SP 0 CRLF CRLF
```

Example is:

```
     LOGOUT 244 0
     0 0
```

3.2.2 SRCID Field

The source ID field shall contain a unique identifier for the client program transmitting a message.

A few key points must be noted regarding this field. The field is designed to hold up to a 4-digit value represented as a character sequence. While numeric IDs would be considered the most favourable choice, the GiJo protocol provides no specific details on *exactly what* designers should choose to represent the SRCID.

Nevertheless, in its most basic implementation it should provide the means to uniquely identify a client element in the network program. Thus, it is the responsibility of the programmer to implement proper design practices to ensure this field is utilized correctly.

3.2.3 DSTID Field

The source ID field shall contain a unique identifier for the client program intended to receive a message.

A few key points must be noted regarding this field. The field is designed to hold up to a <u>4-digit value represented as a character sequence</u>. While numeric IDs would be considered the most favourable choice, the GiJo protocol provides no specific details on *exactly what* designers should choose to represent the SRCID.

Nevertheless, in its most basic implementation the field should provide the means to <u>uniquely identify a client element in the network program</u>. Thus, it is the responsibility of the programmer to implement proper design practices to ensure this field is utilized correctly.

It should be noted that the GiJo protocol provides no recommendations or restrictions

3.2.4 ENCODING Field

The encoding field shall be used to define the intended encoding of the bytes contained in the CONTENT field. This option has been added to provide flexibility in accommodating future updates that may include transmission of more complex data objects using the GiJO protocol.

At the time of this document, the only recommended use for this field should be encoding types designed for transmission of character messages like utf-8 or ascii.

3.2.5 LENGTH Field

The LENGTH field of the protocol shall define the length of the message intended for transmission (contained in the CONTENT field). The value in this field shall be a character sequence representing a numeric value not exceeding 4 bytes in length.

Unless there is a valid reason to do so, the default value of this field shall be zero. In other words, if no bytes are transmitted in the CONTENT field of the message, the value of this field shall be zero.

3.2.6 CONTENT Field

The CONTENT field shall contain a message that the client application intends to SEND. The message shall be composed of any sequence of bytes, with the exception that the message size shall not exceed 512 bytes.

Furthermore, the bytes contained within the the content field shall have no restriction on the type of content being transmitted. It should be noted however that the intended purpose at the time of creation of this document is transmission of alphanumeric characters.

Additionally, some restrictions are provided as a suggestion regarding the CONTENT field and its relationship with other fields in the protocol header. Specifically:

1. **A message containing content size of 0 bytes should contain a LENGTH value of 0**

2. **A protocol message sent with the LOGOUT command should contain a message size of 0 bytes**

3. **A protocol message sent with the LOGIN command should contain a message size of 0 bytes**

4.0 Message Reception

Gijo does not define special response messages for server applications. Rather, it depends on the state of the originally sent message following server processing.

The following section describes the various states of messages received by a client network application. These states provide strict guidelines on interpretations of the different message states based on the contents received from a server network application implementing GiJo.

4.1 Protocol Message Augmentation Rules

The following rules are presented as strict guidelines outlining how certain properties of a GiJo message should be augmented to reflect certain crucial events throughout the lifetime of a message.

The rules are clearly defined below:

1. **Should a message be successfully transferred:**
   a. The CONTENT bytes shall be purged
   b. The LENGTH shall be set to zero

2. **Should a message be transmitted to an offline DSTID,**
   a. The DSTID of the message shall be set to 0

3. **Should the message fail for any unknown reason:**
   a. The CONTENT shall be purged
   b. The SIZE shall be set to zero
   c. The DSTID shall be set to zero

4.2 Message State Logic Table

Additionally, see the table below for a visual depiction of the combined logical conditions that represent the above described rules.

MEANING	**EXAMPLE(s)**	**Condition(s)**
Message has been sent successfully	SEND 1 2 0 SEND 3 2 0 SEND 45 21 0	SRCID > 0 DSTID > 0 LENGTH == 0
Message failed to send	SEND 1 0 0 SEND 3 0 0 SEND 45 0 0	SRCID > 0 DSTID == 0 LENGTH == 0
DSTID is offline	SEND 1 0 23 SEND 3 0 45 SEND 5 0 123	SRCID > 0 DSTID == 0 LENGTH > 0

Appendix A: ABNF Server Protocol Definition

```
PROTOCOL = SERVER

SERVER = CMD
CMD = SEND
USERID = %d48-57
SRCID = USERID
DSTID = USERID
LENGTH = 1*3%d48-57
CONTENT = 1*512(ALPHA)
ENCODING = "ASCII" / "UTF-8"

SEND = SENDCMD SP SRCID SP DSTID CRLF ENCODING SP LENGTH CRLF CRLF
CONTENT CRLF

SENDCMD = %x53%x45%x4E%x44

ALPHA          =  %x41-5A / %x61-7A   ; A-Z / a-z

         BIT            =  "0" / "1"

         CHAR           =  %x01-7F
                                ; any 7-bit US-ASCII character,
                                ;  excluding NUL

         CR             =  %x0D
                                ; carriage return

         CRLF           =  CR LF
                                ; Internet standard newline

         CTL            =  %x00-1F / %x7F
                                ; controls

         DIGIT          =  %x30-39
                                ; 0-9
```

```
DQUOTE         =  %x22
                       ; " (Double Quote)

HEXDIG         =  DIGIT / "A" / "B" / "C" / "D" / "E" / "F"

HTAB           =  %x09
                       ; horizontal tab

LF             =  %x0A
                       ; linefeed

LWSP           =  *(WSP / CRLF WSP)
                       ; Use of this linear-white-space rule
                       ;  permits lines containing only white
                       ;  space that are no longer legal in
                       ;  mail headers and have caused
                       ;  interoperability problems in other
                       ;  contexts.
                       ; Do not use when defining mail
                       ;  headers and use with caution in
                       ;  other contexts.

OCTET          =  %x00-FF
                       ; 8 bits of data

SP             =  %x20

VCHAR          =  %x21-7E
                       ; visible (printing) characters

WSP            =  SP / HTAB
                       ; white space
```

Appendix B: ABNF Server Protocol Definition

```
PROTOCOL = CLIENT

CLIENT = CMD
CMD = SEND / LOGIN / LOGOUT
USERID = %d48-57
SRCID = USERID
DSTID = USERID
LENGTH = 1*3%d48-57
CONTENT = 1*512(ALPHA)
ENCODING = "ASCII" / "UTF-8"

SEND = SENDCMD SP SRCID SP DSTID CRLF ENCODING SP LENGTH CRLF CRLF
CONTENT CRLF
LOGIN = LOGINCMD SP SRCID SP "0" SP "0"CRLF CRLF
LOGOUT = LOGOUTCMD SP SRCID SP 0CRLF 0SP 0CRLF CRLF

SENDCMD = %x53%x45%x4E%x44
LOGINCMD = %x4C%x4F%x47%x49%x4E
LOGOUTCMD = %x4C%x4F%x47%x4F%x55%x54

ALPHA          =  %x41-5A / %x61-7A   ; A-Z / a-z

         BIT            =  "0" / "1"

         CHAR           =  %x01-7F
                                ; any 7-bit US-ASCII character,
                                ;  excluding NUL

         CR             =  %x0D
                                ; carriage return

         CRLF           =  CR LF
                                ; Internet standard newline

         CTL            =  %x00-1F / %x7F
                                ; controls

         DIGIT          =  %x30-39
                                ; 0-9

         DQUOTE         =  %x22
                                ; " (Double Quote)
```

```
        HEXDIG         =  DIGIT / "A" / "B" / "C" / "D" / "E" / "F"

        HTAB           =  %x09
                               ; horizontal tab

        LF             =  %x0A
                               ; linefeed

        LWSP           =  *(WSP / CRLF WSP)
                               ; Use of this linear-white-space rule
                               ;  permits lines containing only
white

                               ;  space that are no longer legal in
                               ;  mail headers and have caused
                               ;  interoperability problems in other
                               ;  contexts.
                               ; Do not use when defining mail
                               ;  headers and use with caution in
                               ;  other contexts.

        OCTET          =  %x00-FF
                               ; 8 bits of data

        SP             =  %x20

        VCHAR          =  %x21-7E
                               ; visible (printing) characters

        WSP            =  SP / HTAB
                               ; white space
```