

my-Teams RFC SAN

Sergueï L, Antoine C, Nicolas M

May 2022

1 History

2 Introduction

2.1 client:

The client sent information is defined by a line starting with a command in uppercase letters, followed by a Space <SP>, followed by its arguments (if needed) separated by a Space <SP>, written between double-quotes, and terminated by the end-of-line code <LF>. The client command must not exceed 512 characters.

2.2 server:

The server reply is defined to contain the 3-digit code, followed by Space <SP>, followed by one line of text, a list of information for specific commands separated by a ';' and terminated by the end-of-line code <LF>.

2.3 code:

1XX: informative reply code
2XX: notifications code
3XX: success reply code
4XX: server error
5XX: error reply code
9XX: fatal error

1. <- : Request command from client to server
2. ->: Reply from server to client
3. Arguments preceded by a '?' are not mandatory.

3 Commands

1. [HELP](#)
2. [LOGIN](#)
3. [LOGOUT](#)
4. [USERS](#)
5. [USER](#)
6. [SEND](#)
7. [MESSAGES](#)
8. [SUBSCRIBE](#)
9. [SUBSCRIBED](#)
10. [UNSUBSCRIBE](#)
11. [USE](#)
12. [CREATE](#)
13. [LIST](#)
14. [INFO](#)
15. [QUIT](#)
16. [UNKNOWN](#)

3.1 Example

4 Notifications

1. [New Private message](#)
2. [New Channel message](#)
3. [Team created](#)
4. [Channel created](#)
5. [Thread created](#)

[back to commands](#)

5 RFC commands

1. HELP

Takes no argument.

Show help, displays the different commands and their uses.

Success:

-> [101](#)

2. LOGIN<SP>"username"

Takes one argument.

Set the username used by client.

Connect user if it already exists, create a new one otherwise.

Success:

-> [102](#)

-> [301](#)

-> [302](#)

Errors:

-> [501](#), [502](#), [511](#)

3. LOGOUT

Takes no argument.

Disconnect the client from the server.

Success:

-> [303](#)

Errors:

-> [503](#)

[back to commands](#)

4. USERS

Takes no argument.

Get the list of all users that exist on the domain.

Success:

-> [311](#)

Errors:

-> [501](#), [503](#)

5. USER<SP>"user uuid"

Takes a user uuid as argument.

Get information about the user.

Success:

-> [312](#)

Errors:

-> [501](#), [502](#), [503](#), [504](#)

-> [521](#)

6. SEND<SP>"user uuid"<SP>"message content"

Takes a user uuid and a message as arguments.

Send the message to the user.

Success:

-> [313](#)

Errors:

-> [501](#), [502](#), [503](#), [504](#)

-> [521](#)

[back to commands](#)

7. MESSAGES<SP>"user uuid"

Takes a user uuid as argument.
List all messages exchanged with the user.

Success:

-> [314](#)<SP>("received" or "sent")<SP>"message content" ; ... <LF>

Errors:

-> [501](#), [502](#), [503](#), [504](#)

-> [521](#)

8. SUBSCRIBE<SP>"team uuid"

Takes a team uuid as argument. Subscribe to the event of the team and its sub directories (enable reception of all events from a team).

Success:

-> [103](#)

-> [315](#)

Errors:

-> [501](#), [502](#), [503](#), [504](#)

-> [522](#)

9. SUBSCRIBED<SP>?"team uuid"

Takes no argument or a team uuid.
List all subscribed teams or list all users subscribed to a team.

Success:

-> [316](#)

-> [317](#)

Errors:

-> [501](#), [503](#), [504](#)

-> [522](#)

[back to commands](#)

10. UNSUBSCRIBE<SP>”team uuid”

Takes a team uuid as argument.
Unsubscribe from the team.

Success:

→ [104](#)

→ [318](#)

Errors:

→ [501](#), [502](#), [503](#), [504](#)

→ [522](#)

11. USE<SP>?”team uuid” <SP>?”channel uuid” <SP>?”thread uuid”

Takes a context uuid as argument.
Use specify the context team/channel/thread.

Success:

→ [319](#)

Errors:

→ [501](#), [503](#), [504](#)

→ [522](#), [523](#), [524](#)

12. CREATE

Takes one or two arguments depending on the context.
Based on what is being used create the sub resource.

Success:

The context is not defined:

<SP> “team name” <SP>? “team description”

→ [321](#)

Team uuid is defined:

<SP> “channel name” <SP>? “channel description”

→ [322](#)

Team uuid and channel uuid are defined:

<SP> “thread title” <SP>? “thread message”

→ [323](#)

Team uuid, channel uuid and thread uuid are defined:

<SP> “comment body”

→ [324](#)

Errors:

→ [501](#), [503](#), [504](#)

→ [512](#), [513](#)

[back to commands](#)

13. LIST

Takes no argument.

Based on what is being used create the sub resource.

Success:

The context is not defined.

-> [331](#)<SP>("team uuid"); ... <LF>

Team uuid is defined.

-> [332](#)<SP>("channel uuid"); ... <LF>

Team uuid and channel uuid are defined.

-> [333](#)<SP>("thread uuid"); ... <LF>

Team uuid, channel uuid and thread uuid are defined.

-> [334](#)<SP>("replie"); ... <LF>

Errors:

-> [501](#), [503](#)

14. INFO

Takes no argument.

Based on what is being used create the sub resource.

Success:

The context is not defined.

→ 341 <SP>"user uuid" <SP>"username" <SP>"team uuid" <SP>"channel uuid" <SP>"thread uuid"; ... <LF>

Team uuid is defined.

→ 342 <SP>"team name" <SP>"team uuid" <SP>"team description"; ... <LF>

Team uuid and channel uuid are defined.

→ 343 <SP>"channel name" <SP>"channel uuid" <SP>"channel description"; ... <LF>

Team uuid, channel uuid and thread uuid are defined.

→ 344 <SP>"thread name" <SP>"thread uuid" <SP>"thread comment"; ... <LF>

Errors:

→ 501, 503

15. QUIT

Takes no argument.

Quit the server.

Success:

→ 304

Errors:

→ 501, 503

16. UNKNOWN

The command does not exist.

→ 500

[back to commands](#)

6 Exemple

```
<- LOGIN <SP> "username"  
-> 501 - Missing double quotes
```

```
<- LOGIN <SP> "username"  
-> 301 - User created.
```

```
<- SEND <SP> "user uuid" <SP> "message"  
-> 313 - Message sent.
```

7 RFC Notification

1. New Private Message

Receive a user uuid and his private message.

Return:

→ 201<SP>(from)"uuid"<SP>(to)"uuid"<SP>"message content"<LF>

2. New Channel Message

Receive message notification from a channel.

Return:

→ 202<SP>(from)"uuid"<SP>"Team uuid"<SP>"Channel uuid"<LF>

3. New Thread Message

Receive message notification from a thread.

Return:

→ 202<SP>(from)"uuid"<SP>"Team uuid"<SP>"Channel uuid"<SP>"thread
uuid"<LF>

4. New Team created

Notify a team creation.

Return:

→ 211<SP>"Team uuid"<LF>

5. New Channel Created

Notify a channel creation.

Return:

→ 212<SP>"Team uuid"<SP>"channel uuid"<LF>

6. New Thread created

Notify a thread creation.

Return:

→ 213<SP>"Team uuid"<SP>"channel uuid"<SP>"thread uuid"<LF>

[back to commands](#)

8 Code

8.1 Information

- 101 - Displays the different commands and their uses.
- 102 - Already logged-in.
- 103 - Already subscribed.
- 104 - Not subscribed.

8.2 Success

8.2.1 Connection

- 301 - User created.
- 302 - User connected.
- 303 - User disconnected.
- 304 - User quit.

8.2.2 User

- 311 - List all users.
- 312 - Show user information.
- 313 - Message sent.
- 314 - List all exchanged messages
- 315 - Successfully subscribed.
- 316 - List all subscribed clients.
- 317 - List all subscribed teams.
- 318 - Successfully unsubscribed.
- 319 - New context specified.

[back to commands](#)

8.2.3 Create

- 321 - New team created.
- 322 - New channel created.
- 323 - New thread created.
- 324 - New reply created.

8.2.4 List

- 331 - List all existing teams.
- 332 - List all existing channels.
- 333 - List all existing threads.
- 334 - List all existing replies.

8.2.5 Info

- 341 - Display currently logged user infos.
- 342 - Display currently selected team infos.
- 343 - Display currently selected channel infos.
- 344 - Display currently selected thread infos.

8.2.6 Notifications

- 201 - Private message received.
- 202 - New channel message.
- 203 - New thread message.
- 211 - New Team created.
- 212 - New Channel created.
- 213 - New Thread created.

[back to commands](#)

8.3 Errors

- 501 - Missing double quotes
- 502 - Command missing arguments
- 503 - Not logged-in
- 504 - Too long command

- 511 - Already logged-in
- 512 - Already exists
- 513 - Not subscribed

- 521 - Wrong user uuid
- 522 - Wrong team uuid
- 523 - Wrong channel uuid
- 524 - Wrong thread uuid

[back to commands](#)