F-Chat Server Commands

From F-List Wiki

Contents

Commands sent by server

ADL

Sends the client the current list of chatops.
Syntax

```
>> ADL { "ops": [string] }
```

Raw sample

```
ADL {"ops": ["Silver", "Hiro", "Jamii", "Oskenso", "Aniko",
"King Mercy", "Hexxy", "Psy Chuan", "Playful Nips",
"LegionRed", "Glitter", "Feath", "Viona", "Ambellina", "Becca
Greene", "Victar"]}
```

AOP

The given character has been promoted to chatop.
Syntax

```
>> AOP { "character": string }
```

BRO

Incoming admin broadcast.
Syntax

```
>> BRO { "message": string }
```

CDS

Alerts the client that that the channel's description has changed. This is sent whenever a client sends a JCH to the server.

Syntax

```
>> CDS { "channel": string, "description": string }
```

Raw sample

```
CDS {"description": "[color=red]No actual roleplay in here. For discussion of anything at all,
please go to the correct channels.[/color] This is the channel for RP offers and announcements.",
"channel": "Looking for RP"}
```

Notes/Warnings

As with all commands that refer to a specific channel, official/public channels use the name, but unofficial/private/open private rooms use the channel ID, which can be gotten from ORS.

CHA

Sends the client a list of all public channels.

Syntax

```
>> CHA { "channels": [object] }
```

Raw sample

```
CHA {"channels": [{"name":"Hermaphrodites","mode":"both","characters":144},
{"name":"Avians","mode":"chat","characters":20},{"name":"World of
Warcraft","mode":"both","characters":77},{"name":"Ageplay","mode":"both","characters":174}
... }
```

Notes/Warnings

The channels object is a dictionary or associative array.

CIU

Invites a user to a channel.

Syntax

```
>> CIU { "sender":string,"title":string,"name":string }
```

Raw sample

```
CIU {"sender":"Jinni Wicked","title":"Test Room","name":"ADH-c7fc4c15c858dd76d860"}
```

Parameters
* sender: The character offering the invitation

- title: The display name for the channel.
- name: The channel ID/name.

Notes/Warnings
In F-Chat 1.0, this renders as: "[user]" + sender + "[/user] has invited you to join [session=" + title + "]" + name + "[/session]."

CBU

This command requires channel op or higher. Removes a user from a channel, and prevents them from re-entering.
Syntax

```
>> CBU {"operator":string,"channel":string,"character":string}
```

Raw sample

```
CBU {"operator":"Teal Deer","channel":"ADH-c7fc4c15c858dd76d860","character":"Pas un
Caractere"}
```

Parameters
operator: The chanop doing the banning.

channel: The channel from which the character is being banned.

character: The character being banned.
Notes/Warnings
As with all commands that refer to a specific channel, official/public channels use the name, but unofficial/private/open private rooms use the channel ID, which can be gotten from ORS.

CKU

This command requires channel op or higher. Kicks a user from a channel.
Syntax

```
>> CKU {"operator":string,"channel":string,"character":string}
```

Raw sample

```
CKU {"operator":"Pas un Caractere","channel":"ADH-
c7fc4c15c858dd76d860","character":"Teal Deer"}
```

Parameters
operator: The chanop doing the kicking.

channel: The channel from which the character is being kicked.

character: The character being kicked.
Notes/Warnings
As with all commands that refer to a specific channel, official/public channels use the name, but unofficial/private/open private rooms use the channel ID, which can be gotten from ORS.

COA

This command requires channel op or higher. Promotes a user to channel operator.
Syntax

```
>> COA {"character":"character_name", "channel":"channel_name"}
```

Raw sample

```
COA {"character":"Teal Deer", "channel":"ADH-c7fc4c15c858dd76d860"}
```

Parameters
* channel: The channel the op is being added to.

- character: The character being made an op.

Notes/Warnings
As with all commands that refer to a specific channel, official/public channels use the name, but unofficial/private/open private rooms use the channel ID, which can be gotten from ORS.

COL

Gives a list of channel ops. Sent in response to JCH.
Syntax

```
>> COL { "channel": string, "oplist": [string] }
```

Raw sample

```
COL { "oplist": ["","Robert Grayson","Natsudra"], "channel": "Frontpage"}
```

Parameters
channel = The channel where these characters are ops.

Notes/Warnings
As with all commands that refer to a specific channel, official/public channels use the name, but unofficial/private/open private rooms use the channel ID, which can be gotten from ORS. The first name in the list will be the channel owner. Keep in mind that not all official channels have owners, so you may receive "" in the first position.

CON

After connecting and identifying you will receive a CON command, giving the number of connected users to the network.
Syntax

```
>> CON { "count": int }
```

Raw sample

```
CON {"count": 254}
```

COR

This command requires channel op or higher. Removes a channel operator.
Syntax

```
>> COR {"character":"character_name", "channel":"channel_name"}
```

Raw sample

```
COR {"character":"Teal Deer", "channel":"ADH-c7fc4c15c858dd76d860"}
```

Parameters
* channel: The channel the op is being removed from.

- character: The character being demoted.

Notes/Warnings
As with all commands that refer to a specific channel, official/public channels use the name, but unofficial/private/open private rooms use the channel ID, which can be gotten from ORS.

CSO

Sets the owner of the current channel to the character provided.
Syntax

```
>> CSO {"character":"string","channel":"string"}
```

Raw sample

```
CSO {"character":"Jinni Wicked","channel":"ADH-3875a3c8c11325b49992"}
```

Parameters
* character: the character to set as owner

- channel: which channel to set the owner in

Notes/Warnings
As with all commands that refer to a specific channel, official/public channels use the name, but unofficial/private/open private rooms use the channel ID, which can be gotten from ORS.

CTU

Temporarily bans a user from the channel for 1-90 minutes. A channel timeout.
Syntax

```
>> CTU {"operator":"string","channel":"string","length":int,"character":"string"}
```

Raw sample

```
CTU {"operator":"Michael Donohue","channel":"ADH-2e7542f374c7ec3a542d","length":10,"character":"TestingStuff"}
```

Parameters
* operator: the moderator enacting the timeout

- channel: which channel to the user was timed out from.
- length: the number of minutes the character is timed out for
- character: which character has been timed out

Notes/Warnings
As with all commands that refer to a specific channel, official/public channels use the name, but unofficial/private/open private rooms use the channel ID, which can be gotten from ORS.

DOP

The given character has been stripped of chatop status.

Syntax

```
>> DOP { "character": character }
```

ERR

Indicates that the given error has occurred.

Syntax

```
>> ERR { "number": int, "message": string }
```

Raw sample

```
ERR {"message": "You have already joined this channel.", "number": 28}
```

FKS

Sent by as a response to the client's FKS command, containing the results of the search.

Syntax

```
>> FKS { "characters": [object], "kinks": [object] }
```

Raw sample

```
FKS {"characters":["Some Guy", "Another Guy", "Some Gal"] "kinks": ["523","66"]}
```

Notes/Warnings

The numbers under kinks are the kinkids sent by the client. All search parameters can be retrieved here (http://www.f-list.net/json/chat-search-getfields.json?ids=true).

FLN

Sent by the server to inform the client a given character went offline.

Syntax

```
>> FLN { "character": string }
```

Raw sample

```
FLN {"character":"Hexxy"}
```

Notes/Warnings

Should be treated as a global LCH for this character.

HLO

Server hello command. Tells which server version is running and who wrote it.

Syntax

```
>> HLO { "message": string }
```

Raw sample

```
HLO {"message":"Welcome. Running F-Chat 0.8.6-Lua by Kira. Enjoy your stay."}
```

ICH

Initial channel data. Received in response to JCH, along with CDS.

Syntax

```
>> ICH { "users": [object], "channel": string, "mode": enum }
```

Raw sample

```
ICH {"users": [{"identity": "Shadlor"}, {"identity": "Bunnie Patcher"}, {"identity":
"DemonNeko"}, {"identity": "Desbreko"}, {"identity": "Robert Bell"}, {"identity": "Jayson"},
{"identity": "Valoriel Talonheart"}, {"identity": "Jordan Costa"}, {"identity": "Skip Weber"},
{"identity": "Niruka"}, {"identity": "Jake Brian Purplecat"}, {"identity": "Hexxy"}], "channel":
"Frontpage", mode: "chat"}
```

Notes/Warnings

"mode" can be "ads", "chat", or "both".

IDN

Used to inform the client their identification is successful, and handily sends their character name along with it.

Syntax

```
>> IDN { "character": string }
```

Notes/Warnings

If you send any commands before identifying, you will be disconnected.

JCH

Indicates the given user has joined the given channel. This may also be the client's character.

Syntax

```
>> JCH { "channel": string, "character": object, "title": string }
```

Raw sample

```
JCH {"character": {"identity": "Hexxy"}, "channel": "Frontpage", "title": "Frontpage"}
```

Notes/Warnings

As with all commands that refer to a specific channel, official/public channels use the name, but unofficial/private/open private rooms use the channel ID, which can be gotten from ORS.

KID

Kinks data in response to a KIN client command.

Syntax

```
>> KID { "type": enum, "message": string, "key": [int], "value": [int] }
```

Parameters

Type: has three valid values; "start", "custom", and "end".

Notes/Warnings

The message field is sent when the type is "start" or "end", as it will be displayed to the user. First, a KID command of type "start" is sent, then a series of KID commands of type "custom", holding "key" "value" properties of the character's custom kinks. Then, finally a KID command of type "end" is sent.

LCH

An indicator that the given character has left the channel. This may also be the client's character.

Syntax

```
>> LCH { "channel": string, "character": character }
```

LIS

Sends an array of all the online characters and their gender, status, and status message.

Syntax

```
>> LIS { characters: [object] }
```

Raw sample

```
LIS {"characters": [["Alexandrea", "Female", "online", ""], ["Fa Mulan", "Female", "busy",
"Away, check out my new alt Aya Kinjou!"], ["Adorkable Lexi", "Female", "online", ""],
["Melfice Cyrum", "Male", "online", ""], ["Jenasys Stryphe", "Female", "online", ""], ["Cassie
Hazel", "Herm", "looking", ""], ["Jun Watarase", "Male", "looking", "cute femmy boi looking for
a dominate partner"],["Motley Ferret", "Male", "online", ""], ["Tashi", "Male", "online", ""],
["Viol", "Cunt-boy", "looking", ""], ["Dorjan Kazyanenko", "Male", "looking", ""], ["Asaki",
"Female", "online", ""]]}
```

Notes/Warnings

Because of the large amount of data, this command is often sent out in batches of several LIS commands. Since you got a CON before LIS, you'll know when it has sent them all.

The characters object has a syntax of ["Name", "Gender", "Status", "Status Message"].

NLN

A user connected.

Syntax

```
>> NLN { "identity": string, "gender": enum, "status": enum }
```

Raw sample

```
NLN {"status": "online", "gender": "Male", "identity": "Hexxy"}
```

Parameters
Identity: character name of the user connecting.
Gender: a valid gender string.
Status: a valid status, though since it is when signing on, the only possibility is online.

IGN

Handles the ignore list.
Syntax

```
>> IGN { "action": string, "characters": [string] | "character":object }
```

Raw sample

```
IGN {"characters":["Teal Deer", "Pas un Caractere", "Testytest"],"action":"init"}
IGN {"character":"Teal Deer","action":"add"}
IGN {"character":"Teal Deer","action":"delete"}
```

Parameters
action

- init: Sends the initial ignore list. Uses characters:[string] to send an array of character names.
- add: Acknowledges the addition of a character to the ignore list. Uses character:"string".
- delete: Acknowledges the deletion of a character from the ignore list. Uses character:"string".
- *Presumably, 'list' and 'notify' also have server responses, but I have yet to get a client that will provide me those in the debug log.*

FRL

Initial friends list.
Syntax

```
>> FRL { "characters": [string] }
```

Notes/Warnings
FRL is a combination of all of this account's bookmarks and friends.

ORS

Gives a list of open private rooms.
Syntax

```
>> ORS { "channels": [object] }
```

Raw sample

```
ORS { channels: [{"name":"ADH-300f8f419e0c4814c6a8","characters":0,"title":"Ariel's Fun Club"},{"name":"ADH-d2afa269718e5ff3fae7","characters":6,"title":"Monster Girl Dungeon RPG"},{"name":"ADH-75027f927bba58dee47b","characters":2,"title":"Naruto Descendants OOC"} ...] }
```

Parameters
* name: The ID of the room. ADH- and twenty characters of hexadecimal.

- characters: The number of characters in the room.
- title: The user-friendly name of the room. This can contain all sorts of weird extended ASCII, so be prepared to handle that.

Notes/Warnings
The channels object is an associative array/dictionary.

PIN

Ping command from the server, requiring a response, to keep the connection alive.
Syntax

```
>> PIN
```

Notes/Warnings
You have to respond to pings or you will be disconnected, as that will be detected as a timeout. The server will try to get a ping response three times, each time waiting 30 seconds. So if you don't respond, you will be disconnected after 90 seconds. Sending multiple pings within ten seconds will get you disconnected also.

PRD

Profile data commands sent in response to a PRO client command.
Syntax

```
>> PRD { "type": enum, "message": string, "key": string, "value": string }
```

Notes/Warnings
The message field is sent when the type is "start" or "end", as it will be displayed to the user. First, a

PRD command of type "start" is sent, then a series of PRD commands of type "info" and "select", holding "key" "value" properties of the character's profile properties. Then, finally a PRD command of type "end" is sent.

PRI

A private message is received from another user.
Syntax

```
>> PRI { "character": string, "message": string }
```

Notes/Warnings
There is flood control; the same as the MSG command. At this time of writing, the maximum length of a private message is 50000 characters.

MSG

A message is received from a user in a channel.
Syntax

```
>> MSG { "character": string, "message": string, "channel": string }
```

Notes/Warnings
There is flood control and a max length. As with all commands that refer to a specific channel, official/public channels use the name, but unofficial/private/open private rooms use the channel ID, which can be gotten from ORS.

LRP

A roleplay ad is received from a user in a channel.
Syntax

```
>> LRP See MSG.
```

Raw sample

```
LRP { "channel": "Sex Driven LFRP", "message": "WHERE IS IT!?!? *uses magnifying glass*",
"character": "Jay Rabbit"}
```

RLL

Rolls dice or spins the bottle.
Syntax

```
>> RLL {"channel": string, "results": [int], "type": enum, "message": string, "rolls": [string],
"character": string, "endresult": int}
OR
>>RLL
{"target":"string","channel":"string","message":"string","type":"bottle","character":"string"}
```

Raw sample

```
RLL {"channel":"ADH-dce8eb7af86213ac4c15","results":[22],"type":"dice","message":"[b]Teal
Deer[/b] rolls 4d10: [b]22[/b]","rolls":["4d10"],"character":"Teal Deer","endresult":22}
RLL {"target":"Teal Deer","channel":"ADH-c7fc4c15c858dd76d860","message":"[b]Michael
Donohue[/b] spins the bottle: [b]Teal Deer[/b]","type":"bottle","character":"Michael Donohue"}
```

Parameters
channel: What channel to display the result in.
type: dice or bottle
character: Who rolled the dice.
message: The message the official clients will display in the channel.
If type:dice

- results: Separate results for each set of dice.
- rolls: An array of dice sets and added numbers.
- endresult: The sum of all results.

If type:bottle

- target: who was selected.

Notes/Warnings
As with all commands that refer to a specific channel, official/public channels use the name, but unofficial/private/open private rooms use the channel ID, which can be gotten from ORS.

RMO

Change room mode to accept chat, ads, or both.
Syntax

```
>> RMO {"mode": enum, "channel": string}
```

Raw sample

```
RMO {"mode":"chat","channel":"ADH-c7fc4c15c858dd76d860"}
```

Parameters

channel: Which channel is being changed.

mode:

- chat: Show only MSG.
- ads: Show only LRP.
- both: Show MSG and LRP.

Notes/Warnings

As with all commands that refer to a specific channel, official/public channels use the name, but unofficial/private/open private rooms use the channel ID, which can be gotten from ORS.

RTB

Real-time bridge. Indicates the user received a note or message, right at the very moment this is received.

Syntax

```
>> RTB { "type": string, "character": string }
```

SFC

Alerts admins and chatops (global moderators) of an issue.

Syntax

```
>> SFC {action:"string", moderator:"string", character:"string", timestamp:"string"}
>> SFC {callid:int, action:"string", report:"string", timestamp:"string", character:"string",
logid:int}
```

Raw sample

```
SFC {"action":"confirm", "moderator":"glitter", "character":"testingstuff", "timestamp":"string"}
SFC {callid:int, action:"report", report:"Current Tab/Channel: Sex Driven LFRP | Reporting User:
```

```
TestingStuff | :3 is not an RP ad.", timestamp:"", character="testingstuff","logid":18924}
```

Parameters
* action: Either 'report' or 'confirm'. Report sends the report out to all globals/admins. Confirm claims the report and removes it from the list of unclaimed alerts.

- report (report): This includes the channel, the reported character name, and the text of the report.
- timestamp: The time the report was filed. Give me a few, I'm still trying to figure out the format.
- character: The character that filed the report. This should probably be turned into a user link or PM link, by the client.
- moderator (confirm): The moderator/admin who claimed the report
- callid: The alert number. This should be processed into the confirm link/button by the client, on report.
- logid (report): The log number. Stick this at the end of a log link, to provide access to the log connected to this report: http://www.f-list.net/fchat/getLog.php?log=<logid>

Notes/Warnings
Also responds with SYS to the reporter: "SYS {message="The moderators have been alerted."}" This is not complete, and parts may be a little off. Let me know if you're getting parameters I haven't included or if I've mislabeled a string/int. I have no example for the timestamp parameter.

STA

A user changed their status
Syntax

```
>> STA { status: "status", character: "channel", statusmsg:"statusmsg" }
```

Raw sample

```
{"status":"looking","character":"Jippen Faddoul","statusmsg":"Just testing something"}
```

SYS

An informative autogenerated message from the server. This is also the way the server responds to some commands, such as RST, CIU, CBL, COL, and CUB. The server will sometimes send this in concert with a response command, such as with SFC, COA, and COR.
Syntax

```
>> SYS { "message": string, "channel": string }
```

Raw sample

```
SYS { "message":"Testytest has been added to the moderator list for derp","channel": "ADH-011aeb5bb591b1f4721a"}
SYS { "message":"Your invitation has been sent" }
```

Parameters
message: The message the server is sending
channel: An optional argument if the message is related to a channel (such as a COL)
Notes/Warnings
There currently is no command send in response to COL, CUB, CBL, and RST. Your client will have to parse SYS messages to look for these commands in order to implement them entirely. As with all commands that refer to a specific channel, official/public channels use the name, but unofficial/private/open private rooms use the channel ID, which can be gotten from ORS.

TPN

A user informs you of his typing status.
Syntax

```
>> TPN { "character": string, "status": enum }
```

Raw sample

```
TPN {"character":"Leon Priest","status":"clear"}
```

Parameters
status: can have a value of "clear", "paused", and "typing".

UPT

Informs the client of the server's self-tracked online time, and a few other bits of information
Syntax

```
>> UPT { "time": int, "starttime": int, "startstring": string, "accepted": int, "channels": int, "users": int, "maxusers": int }
```

Raw sample

```
UPT { "time": 1359398530, "starttime": 1353393109, "startstring": "Tue, 20 Nov 2012 06:31:49 +0000", "accepted": 1665213, "channels": 694, "users": 2105, "maxusers": 2946 }
```

Parameters
time: POSIX timestamp of the current time
starttime: POSIX timestamp of when the server was last started
startstring: Human-readable timestamp of when the server was last started

accepted: How many connections have been accepted since last start
channels: How many channels the server recognizes
users : How many users are currently connected
maxusers: The peak count of online users since last restart

VAR

Variables the server sends to inform the client about server variables.

Syntax

```
>> VAR { "variable": string, "value": int/float }
```

Raw sample

```
VAR {"value":4096,"variable":"chat_max"}
VAR {"value":50000,"variable":"priv_max"}
VAR {"value":50000,"variable":"lfrp_max"}
VAR {"value":600,"variable":"lfrp_flood"}
VAR {"value":0.5,"variable":"msg_flood"}
VAR {"value":["frontpage"],"variable":"icon_blacklist"}
VAR {"value":"35868","variable":"permissions"}
```

Parameters

value: All parameters except for msg_flood should be treated as integers.

variable:

- chat_max: Maximum number of bytes allowed with MSG.
- priv_max: Maximum number of bytes allowed with PRI.
- lfrp_max: Maximum number of bytes allowed with LRP.
- lfrp_flood: Required seconds between LRP messages.
- msg_flood: Required seconds between MSG messages.
- permissions: Permissions mask for this character.
- icon_blacklist: An array of channels that do not allow (e)icons.

Notes/Warnings

Permissions mask is a little complicated.

Admin: 1
chat-chatop: 2
chat-chanop: 4
helpdesk-chat: 8
helpdesk-general: 16
moderation-site: 32
reserved: 64
misc-grouprequests: 128
misc-newsposts: 256
misc-changelog: 512

misc-featurerequests:	1024
dev-bugreports:	2048
dev-tags:	4096
dev-kinks:	8192
developer:	16384
tester:	32768
subscriptions:	65536
former-staff:	131072

F-Chat	
Clients	F-Chat 1.0 | F-Chat 2.0 | F-Chat 3.0 | F-Chat Pidgin/Adium Plugin | slimCat | Official List of Third-Party Clients | Rules for Chat Bots
For Users	F-Chat Command Reference | List of BBCode Tags | Smileys | Chat Help | Chat Troubleshooting | Websockets Troubleshooting | Chat Rules | F-Chat Official Channels
For Developers	Protocol | Error Codes | Client Commands | Server Commands | JSON Endpoints

Retrieved from "https://wiki.f-list.net/index.php?title=F-Chat_Server_Commands&oldid=2581"

Categories: Developer Resources | F-Chat

- This page was last edited on 26 February 2017, at 06:52.