FriendUP Developer's Manual

Volume 3, FriendUP API documentation

"Draft", Rev. 1

January, 2018

Table of Contents

Introduction

This document describes the HTTP/Websocket interface between FriendCore and the Workspace - the Workspace uses websockets for most of the calls to optimise speed.

Most requests are self-explanatory. Some probably are not as Friend comes with a couple of unique features. Please refer to our Developers' manual to get explanatory information on the different parts of the API.

While we will strive hard to keep the API as stable as possible there might be changes in upcoming releases. Watch out for unfinished functionality and use with caution.

The system.library

The system.library in FriendUP is the extensible component that handles most of the logic in Friend Core. It establishes the operating system template in the server core. It upgrades operating system features of the underlying OS (Linux or Windows) to behave like a Friend system.

Using devices

In Friend Core, devices are units connected to Friend Core using DOS drivers and DOS handlers. There are a few ways to manipulate these using the device library calls of the system.library.

System.library misc

system.library/help

Function return all available Friend WebCalls

Parameters

sessionid	(required) - id of user session

---

system.library/login

Function allow user to login and get user session id

Parameters

username	(required) - name of user
password	(required) - user password
deviceid	(required) - id which recognize device which is used to login
encryptedblob	- used by keys which allow to login
appname	- application name which want
sessionid	- session id of logged user

Returns

information about login process "{result:-1,response: success/fail, code:error code }

---

system.library/module

Function allow user to call functions handled in modules

Parameters

sessionid	- (required) session id of logged user
module	- (required) module which will be used (all other params will be taken from request)

Returns

output from modules

System.library user

system.library/user/create

Create user. Function require admin rights.

Parameters

sessionid	- (required) session id of logged user
username	- (required) user name
password	- (required) password
fullname	- full user name
email	- user email
level	- groups to which user will be assigned, separated by comma

Returns

{ create: sucess } when success, otherwise error with code

---

system.library/user/delete

Delete user. Function require admin rights.

Parameters

sessionid	- (required) session id of logged user
id	- (required) id of user which you want to delete

Returns

{ Result: success} when success, otherwise error with code

---

system.library/user/updatepassword

Update password

Parameters

sessionid	- (required) session id of logged user
username	- (required) name of the user which will go through change password process
password	- (required) new password

Returns

{ updatepassword: success!} when success, otherwise error with code

---

system.library/user/update

Update user. Changes on other user accounts require admin rights.

Parameters

sessionid	- (required) session id of logged user
id	- (required) id of user which you want to change
username	- new user name
password	- new password
fullname	- new full user name
email	- new user email
level	- new groups to which user will be assigned. Groups must be separated by comma sign

Returns

{ update: success!} when success, otherwise error with code

---

system.library/user/logout

Logout user

Parameters

sessionid	- (required) session id of logged user

Returns

{ logout: success!} when success, otherwise error with code

---

system.library/user/sessionlist

Get sessions attached to user

Parameters

sessionid	- (required) session id of logged user
username	- (required) name of user which sessions you want to get

Returns

sessions attached to users in JSON format, otherwise error code

---

system.library/user/killsession

Kill user session (remote logout)

Parameters

sessionid	- (required) session id of logged user
sesid	- (required if deviceid and username are not available) session id of user which you want to kill
deviceid	- (required if sesid is not available) device id of user which you want to kill
username	- (required if sesid is not available) user name of user which you want to kill

Returns

{ killsession: success} when success, otherwise error code

---

system.library/user/activelist

Get active user list (avaiable in FC memory)

Parameters

sessionid	- (required) session id of logged user
usersonly	- if set to 'true' get unique user list

Returns

all users in JSON list when success, otherwise error code

---

system.library/user/activelwsist

Get active user list, all users have working websocket connections

Parameters

sessionid	- (required) session id of logged user
usersonly	- if set to 'true' get unique user list

Returns

all users in JSON list when success, otherwise error code

---

system.library/user/updatekey

Update key. Function reload key assigned to user from database

Todo:

this function should not be here probably

Parameters

sessionid	- (required) session id of logged user
keyid	- (required) key id which will be reloaded

Returns

{ result: sucess } when success, otherwise error code

System.library services

system.library/services/list

List all avaiable services. Function send commands to all connected servers and get information about services avaiable.

Parameters

sessionid - (required) session id of logged user

Returns

servers and services avaiable on them in JSON format when success, otherwise error code

---

system.library/services/<service_name>/start

Start service

Parameters

sessionid - (required) session id of logged user

Returns

{ Status:ok } when success, otherwise error code

---

system.library/services/<service_name>/stop

Stop service

Parameters

sessionid - (required) session id of logged user

Returns

{ Status:ok } when success, otherwise error code

---

system.library/services/<service_name>/pause

Pause service

Parameters

sessionid - (required) session id of logged user

Returns

{ Status:ok } when success, otherwise error code

---

system.library/services/<service_name>/install

Install service

Parameters

sessionid	- (required) session id of logged user

Returns

{ Status:ok } when success, otherwise error code

---

system.library/services/<service_name>/uninstall

Uninstall service

Parameters

sessionid	- (required) session id of logged user

Returns

{ Status:ok } when success, otherwise error code

---

system.library/services/<service_name>/status

Get status of service

Parameters

sessionid	- (required) session id of logged user

Returns

{ Status:ok } when success, otherwise error code

---

system.library/services/<service_name>/command

Send command to service

Parameters

sessionid	- (required) session id of logged user

cmd	- command passed to service
servers	- if not available then command will be send to all FC with current service. If parameter will be available then command will be passed to servers separated by comma

Returns

{ Status:ok } when success, otherwise error code

---

system.library/services/<service_name>/getwebgui

Get service user interface (html form)
Parameters

sessionid	- (required) session id of logged user

Returns

GUI form when success, empty <div><> when gui is not available

System.library device

Our device interface describes the options for managing users' drives and other devices.

system.library/device/refreshlist

Refresh user drives. This function check database Filesystem changes and mount unmounted devices.

Parameters

sessionid	- (required) session id of logged user

Returns

information which devices are mounted

---

system.library/device/knock

Get detailed information about drive.

Parameters

sessionid	- (required) session id of logged user
devname	- (required) device name

Returns

ok + separator + message when call passed without problems, otherwise error code will be returned

---

system.library/device/polldrives

Return list of available drives

Parameters

sessionid	- (required) session id of logged user Hogne give more details

---

system.library/device/mount

Mount device

Parameters

sessionid	- (required) session id of logged user
devname	- (required) device name. Parameter and logged user id is used to get user device parameters from Filesystem table.
path	- used by most filesystems to point directory which will be mounted (like server filesystem)
enc	- set to 'true' to encode data on disk
type	- filesystem type

execute	- point to application which will be launched when mount function will be called
visible	- set to 'true' if you want to make device visible for users

Returns

{ Response: Mounted successfully. } when success, otherwise error number

---

system.library/device/unmount

Unmount device

Parameters

sessionid	- (required) session id of logged user
devname	- (required) device name which system will try unmount.

Returns

{ Response: Successfully unmounted } when success, otherwise error code

---

system.library/device/refresh

Refresh user drive in FriendCore

Parameters

sessionid	- (required) session id of logged user
devname	- (required) device name. Parameter and logged user id is used to get user device parameters from Filesystem table.

Returns

{ Result: Device updated!} when success, otherwise error number

---

system.library/device/share

Refresh user drive in FriendCore

Parameters

sessionid	- (required) session id of logged user
devname	- (required) device name which you want to share with another user
username	- (required) name of the user to which you want to share your drive

Returns

{ Result: Device shared successfully} when success, otherwise error code

---

system.library/device/list

List mounted devices

Parameters

sessionid	- (required) session id of logged user

Returns

All devices and their attributes in JSON when success, otherwise error

---

system.library/device/listsys

List all avaiable filesystems

Parameters

sessionid	- (required) session id of logged user

Returns

All filesystems and their attributes in JSON when success, otherwise error

---

system.library/device/update

Store device attributes in database

Parameters

sessionid	- (required) session id of logged user

Returns

{ Result: Database updated} when success, otherwise error code

System.library file

system.library/file/info

Get information about file

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to file with device name before ':' sign

Returns

return information about file in JSON format when success, otherwise error code

---

system.library/file/call

Call file

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to file with device name before ':' sign
args	- (required) additional parameters to call function

Returns

return call response in JSON format when success, otherwise error code

---

system.library/file/dir

Get directory content

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to directory

Returns

return directory entries in JSON format when success, otherwise error code

---

system.library/file/rename

Rename file or directory

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to file which name you want to change
newname	- (required) new file name

Returns

{ response:0 } when success, otherwise error number

---

system.library/file/delete

Delete file or directory

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to file which you want to delete

Returns

{ response:0 } when success, otherwise error number

---

system.library/file/makedir

Make directory in specified path

Parameters

sessionid	- (required) session id of logged user
path	- (required) path which will be used to generate directory

Returns

{ response:0 } when success, otherwise error number

---

system.library/file/exec

Execute file

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to file which you want to delete

Returns

response generated by file when success, otherwise error code

---

system.library/file/read

Read or download file

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to file which you want to read
mode	- (required) "rb" - read bytes, "rs" - read as stream
offset	- offset from which file will be readed
bytes	- number of bytes which you want to read
download	- if set to 1 then whole file will be readed and no friend special header will be added

Returns

file content when success, otherwise error number

---

system.library/file/write

Write data to file

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to file which you want to delete
mode	- (required) "wb" - write binary
data	- (required) data which will be stored in file
size	- (required) number of bytes which will be stored in file
encoding	- type of encoding, currently only "url" type is supported

Returns

{ FileDataStored : <number of="" bytes="" stored>=""> } when success, otherwise error number

---

system.library/file/copy

Copy file from one place to another

Parameters

sessionid	- (required) session id of logged user
from	- (required) path to source file
to	- (required) path to destination path

Returns

{ response: 0, Written: <number of="" bytes>="">} when success, otherwise error number

---

system.library/file/upload

Upload file to Friend drive

Parameters

sessionid	- (required) session id of logged user
path	- (required) path where file will be uploaded

Returns

{ Uploaded files: <number>} when success, otherwise error number

---

system.library/file/diskinfo

Get information aboutdisk

Parameters

sessionid	- (required) session id of logged user
path	- (required) path with device name

Returns

{ Disksize: <size of disk in bytes>, StoredBytes: <user disk space> } when success, otherwise error number

---

system.library/file/expose

Share file (make this file avaiable from outside)

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to file which you want to share

Returns

{hash:<generated hash>, name:<name of shared file> } when success, otherwise error number

---

system.library/file/conceal

Unshare file (make this file not avaiable from outside)

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to file which you want to share

Returns

{ response: } when success, otherwise error number

---

system.library/file/checkaccess

Check if actual user have access to file

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to file which access rights you want to check

Returns

{ Result: access } when success, otherwise error number

---

system.library/file/access

Get file access rights

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to file which access rights you want to check

Returns

file access rights when success, otherwise error number

---

system.library/file/protect

Protect file, directory or drive

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to file on which access rights will be set
user	- access rights for file owner (ARWED string)
group	- access rights for user groups
other	- access rights for others

Returns

{ Result: access } when success, otherwise error number

---

system.library/file/notificationstart

Create new notification. When notification is set on path and event will appear on it then user will get notification.

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to file or directory on which notification will be set
id	- notification id, if its set then FC is trying to update notification instead of create new one

Returns

{ Result: 0} when success, otherwise error number

---

system.library/file/notificationremove

Delete notification.

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to directory or file where you are
id	- notification id which you want to remove

Returns

{ Result: 0 } when success, otherwise error number

---

system.library/file/notifychanges

Send notification to all UserSessions which set notification on path

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to file

Returns

{ Result: 0 } when success, otherwise error number

---

system.library/file/compress

Compress files and folders

Parameters

sessionid	- (required) session id of logged user
files	- (required) path to files or directories which you want to archive. Entries must be separated by semicolon
archiver	- (required) type or archivizer. Currently only zip is supported
destination	- (required) path to place where archive will be stored

Returns

{ Result: 0 } when success, otherwise error number

system.library/file/decompress

Decompress file

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to place where archive will be decompressed
archiver	- (required) type or archivizer. Currently only zip is supported

Returns

{ Result: 0 } when success, otherwise error number

system.library/file/infoget

Get metadata from file

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to file or directory from which you want to get metadata
key	- (required) key name

Returns

metadata in JSON format when success, otherwise error number

system.library/file/infoset

Set metadata on file

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to file or directory on which you want to set metadata

key	- (required) key name
value	- (required) data which will be stored inside the key

Returns

{ Result: Info Set for file"} when success, otherwise error number

---

system.library/file/getmodifydate

Get file or directory modification date

Parameters

sessionid	- (required) session id of logged user
path	- (required) path to file or directory from which modification date will be taken

Returns

{ modifytime: <date>} when success, otherwise error number

System.library ufile

system.library/ufile/open

Open file

Parameters

sessionid	- (required) session id of logged user
path	- (required) path with device name to file
mode	- (required) mode in format "rb" - read binary, "wb" - write binary, "rs" - read stream

Returns

{fileptr:<number>} when success, otherwise error code

---

system.library/ufile/close

Close file

Parameters

sessionid	- (required) session id of logged user
fptr	- (required) pointer to opened file

Returns

{result:success} when success, otherwise error code

---

system.library/ufile/read

Read file

Parameters

sessionid	- (required) session id of logged user
fptr	- (required) pointer to opened file
size	- (required) number of maximum bytes which you want to receive

Returns

received bytes when success, otherwise error code

---

system.library/ufile/write

Write file

Parameters

sessionid	- (required) session id of logged user
fptr	- (required) pointer to opened file
size	- (required) number of bytes which you want to send
data	- (required) data which you want to send

Returns

{filestored:<stored bytes>} when success, otherwise error code

System.library connection

system.library/connection/list

Get information about FC connections

Parameters

sessionid	- (required) session id of logged user
id	- provide if you want to get information about specific connection

Returns

function return information about FriendCore connection

---

system.library/connection/listcluster

Get information about FCCluster connections

Parameters

sessionid	- (required) session id of logged user
address	- provide internet address you want to get information about specific connection

Returns

function return information about FriendCore connection

---

system.library/connection/add

add new FCConnection

Parameters

sessionid	- (required) session id of logged user
address	- (required) internet address (ip or DNS name)
name	- (required) name which describe new connection
type	- type of connection, avaiable values are: 0 - normal server, 1 - cluster node, 2 - cluster master, 3 - master of Friend servers

Returns

return code 59 if everything was created without problems, otherwise error number

---

system.library/connection/edit

edit FCConnection

Parameters

sessionid	- (required) session id of logged user
id	- (required) id of entry which you want to update
address	- internet address
name	- name of connection
destinationfcid	- Friend Core ID of destination server
pem	- ssl key in "pem" format

clusterid	- id in cluster
approved	- 1 if server was approved by admin, 0 - if not
servertype	- type of connection, avaiable values are: 0 - normal server, 1 - cluster node, 2 - cluster master, 3 - master of Friend servers

Returns

return code 0 if everything was updated without problems, otherwise error number

---

system.library/connection/add

add new FCConnection

Parameters

sessionid	- (required) session id of logged user
id	- (required if name is not provided) id of entry which you want to remove (disconnect)
name	- (required if id is not provided) name of entry which you want to remove (disconnect)

Returns

return code 63 if everything was created without problems, otherwise error number

System.library app

system.library/app/help

add new FCConnection

Parameters

sessionid	- (required) session id of logged user

Returns

return information about avaiable functions (app section)

---

system.library/app/list

return all application names avaiable on the server

Parameters

sessionid	- (required) session id of logged user

Returns

return application avaiable on the server in JSON format

---

system.library/app/userlist

List of all users connected to Application Shared Session

Parameters

sessionid	- (required) session id of logged user
sasid	- (required) shared session id
usersonly	- set true if you want to get only unique user names

Returns

string with information which users are connected to shared session

---

system.library/app/register

Register new Application Shared Session

Parameters

sessionid	- (required) session id of logged user

authid	- (required) authentication id (provided by application)

Returns

{ SASID: <number> } when success, otherwise response with error code

---

system.library/app/unregister

Unregister Application Shared Session

Parameters

sessionid	- (required) session id of logged user
sasid	- (required) shared session id which will be removed

Returns

{SASID:<number>} when success, otherwise error code

---

system.library/app/accept

Accept invitation from assid owner

Parameters

sessionid	- (required) session id of logged user
sasid	- (required if authid is not provided) shared session id
authid	- (required if sasid is not provided) application authentication id

Returns

{response:success,identity:<user name>="">}, when success, otherwise error code

---

system.library/app/decline

Decline invitation from assid owner

Parameters

sessionid	- (required) session id of logged user
sasid	- (required) shared session id

Returns

{response:success,identity:<user name>="">}, when success, otherwise error code

---

system.library/app/share

Share your Application Shared Session with other users

Parameters

sessionid	- (required) session id of logged user
sasid	- (required) shared session id
users	- (required) users which we want to invite to Shared Application Session. Function expect user names separated by comma
message	- information which we want to send invited people

Returns

{response:success,identity:<user name>="">}, when success, otherwise error code

---

system.library/app/unshare

Unshare your Application Shared Session. Terminate

Parameters

sessionid	- (required) session id of logged user
sasid	- (required) shared session id
users	- (required) users which we want to remove from Shared Application Session. Function expect user names separated by comma

Returns

list of users removed from SAS

system.library/app/send

Send message to other users (not owner of sas)

Parameters

sessionid	- (required) session id of logged user
sasid	- (required) shared session id
usernames	- users to which we want to send message. Function expect user names separated by comma
msg	- (required) message which we want to send to users

Returns

{response:success}, when success, otherwise error code

system.library/app/sendowner

Send message to SAS owner

Parameters

sessionid	- (required) session id of logged user
sasid	- (required) shared session id
msg	- (required) message which we will be send

Returns

{response:success}, when success, otherwise error code

system.library/app/takeover

Take over other user SAS session

Parameters

sessionid	- (required) session id of logged user

sasid	- (required) shared session id
username	- (required) user name which will take over of SAS
deviceid	- (required) deviceid of user device which will take over SAS

Returns

{response:success}, when success, otherwise error code

---

system.library/app/switchsession

Switch user SAS session

Parameters

sessionid	- (required) session id of logged user
sasid	- (required) shared session id
deviceid	- (required) deviceid of user device to which user want to switch in SAS

Returns

{response:success}, when success, otherwise error code

---

system.library/app/putvar

Put variable into Application Session

Parameters

sessionid	- (required) session id of logged user
sasid	- (required) shared session id
var	- (required) variable which will be stored in SAS
varid	- variable ID, if not provided new will be created. Otherwise updated
mode	- set to "private" if you want to have private variable. Otherwise it will be public

Returns

{VariableNumber:<number>}, when number > 0 then variable was created/updated. Otherwise error number will be returned

---

system.library/app/getvar

Get variable from Application Session

Parameters

sessionid	- (required) session id of logged user
sasid	- (required) shared session id
varid	- variable ID from which data will be taken

Returns

{VariableData:<data>} when success, otherwise error with code

---

system.library/app/install

Install application

Parameters

sessionid	- (required) session id of logged user
url	- (required) url to application which will be installed

Returns

Function not finished

System.library admin

system.library/admin/info

Function return information about FriendCore

Parameters

sessionid	- (required) session id of logged user

Returns

function return information about FriendCore

---

system.library/admin/listcores

Function return information about FriendCore and connected FCores

Parameters

sessionid	- (required) session id of logged user
module	- (required) module which will be used (all other params will be taken from request)

Returns

function return information about FriendCore and connected FCores

---

system.library/admin/connectionsinfo

Get information about FC connections

Parameters

sessionid	- (required) session id of logged user

Returns

function return information about FriendCore connections

---

system.library/admin/remotecommand

Send command to remote server

Parameters

sessionid	- (required) session id of logged user

Returns

remote functions response

system.library/admin/servermessage

Send message to all sessions

Parameters

sessionid	- (required) session id of logged user

Returns

remote functions response

system.library/admin/restartws

Restart Websockets function

Parameters

sessionid	- (required) session id of logged user

System.library INVAR

system.library/invar/listgroups

Return all INVAR groups

Parameters

sessionid	- (required) session id of logged user

Returns

return available INVAR groups

system.library/invar/listgroup

List all avaiable entries in INVAR group

Parameters

sessionid	- (required) session id of logged user
gid	- (required) invar group id

Returns

return available INVAR entries in group

---

system.library/invar/centry

Create INVAR entry and add it to group

Parameters

sessionid	- (required) session id of logged user
gid	- (required) invar group id
name	- (required) entry name
data	- (required) user data

Returns

string status:ok, response:<invar entry>=""> when success, otherwise error code

---

system.library/invar/dentry

Delete INVAR entry

Parameters

sessionid	- (required) session id of logged user
gid	- (required) invar group id
eid	- (required) entry id

Returns

string status:ok when success, otherwise error code

---

system.library/invar/uentry

Update INVAR entry

Parameters

sessionid	- (required) session id of logged user
eid	- (required) entry id
data	- (required) user data

Returns

string status:ok, response:<invar entry>=""> when success, otherwise error code

---

system.library/invar/gentry

Get INVAR entry

Parameters

sessionid	- (required) session id of logged user
eid	- (required) entry id

Returns

string status:ok, response:<invar entry>=""> when success, otherwise error code

---

system.library/invar/cgroup

Create INVAR group

Parameters

sessionid	- (required) session id of logged user
name	- (required) group name

Returns

string status:ok, response:<invar entry>=""> when success, otherwise error code

---

system.library/invar/dgroup

Delete INVAR group

Parameters

sessionid	- (required) session id of logged user no finished