The Protocol

Basics
The connection between the client and server is continuous whilst the client is using the software in ‘on-line mode’. Once the connection is created (using standard TCP/IP SYN/ACK connection) client and server talk using some standard conventions.

For each command and response all information is sent in one chunk. A colon is used as the delimiter between sections of the chunk and a semicolon is used to indicate the end of the chunk. When these symbols are in the data for the chunk they can be escaped using standard methods.

The client sends requests and the server responds indicating if the command was successful and giving any other information required. Server response always starts with a number and will be in one of two forms. Success is indicated simply by a 0 being sent. An error is in the form of number:errormessage where the number specifies the type of error (indication how the client should proceed) and the message is what the client is to display. Some standard error messages can be given by keywords but this allows easier addition of error situations that we may not foresee.

Error numbers
1 Connection will be terminated after the error has been sent – shut down connection (e.g Incorrect username and password)
2 Command has failed but connection will continue (e.g Invalid command or bad arguments) *later we may want to split this into two errors – one for internal errors, eg invalid command and one with messages to the user*
3 Command failed because you don’t have the correct permissions to do something – but continue the connection

An exception to this situation is in the case of update commands where a list of information (eg calendars) is given. These commands end with a number followed by colon – this number is the system time of the server (thus it will never be 1 – 4)

Initialising the connection
Once the connection has been established the server will send a success response to indicate that it is ready for communication. The client the sends the first chunk starting with **user** followed the username. The server will respond with a successful response to indicate the client is to send the password – it doesn’t let the user know yet if the username is invalid as this could be a security risk. Next the client sends **pass** followed the password, the server sends a response indicating success or error. Finally the client sends **version** followed by a string indicating the version of the client and any information about encryption (this will be elaborated upon later). Once the server has sent the appropriate response if this process has been successful the server will now simply wait for commands from the client and process them.

Commands
Commands have been kept to a minimum, these should be used as building blocks so that changes in the later stages are easy to make.

These are the commands and syntax. Commands are all lower case.

<u>**Calendar commands**</u>

list:edittimefrom This lists all the calendars that the user is a member of. Only calendars that have been changed from edittimefrom onwards are returned (the full list – which’ll have to be accessed a lot is accessed if the argument is 0) These are returned in the form *0:calendarname:calendarID:groupID* where calendar ID is the unique 8 digit number used to reference the calendar, and groupID the unique 8 digit group identifier. The end of the list is indicated when the server time (seconds past 1970) followed by a ; is sent

listmine This lists all calendars the user is an owner of (not listed in list). These are returned in the same form as the list. End is indicated by empty success response

calinfo:calID:gid Returns a text description of a calendar in the form *0:information*

getcal:calendarid:groupid:edittimefrom Get information from a calendar. Calendar ID is the identifier for the chosen calendar and datefrom is date from which any events which have been added or edited should be returned. The date is given in the form of the number of seconds after 1970 the date is. To get an entire calendar datefrom would be 0.
Each calendar entry is returned as one chunk
*0:eventID:starttime:endtime:repeatperiod:repeattimes:title:data*starttime and endtime are in the same format as described before, title and data are strings.
Repeatperiod is a string, repeattimes will be an integer. These can be in any format for now because they're delt with exclusively by the client
The end of the list is indicated by a timestamp response containing the server time (seconds past 1970) followed by a ; is sent, *eg 0:123889123;*

createcal:name:groupID Create a new calendar. Name is the name of the calendar. Group is the group this calendar is for. If this is successful the response will be *0:calendarID*

remcal:calendarID:groupID Remove a calendar you own. Response is standard response. Currently the calendar owner and group owners can remove calendars.

getcalendars:groupID:edittimefrom Send all calendars in a group that have been changed since edittimefrom. The end of the list is indicated when the server time (seconds past 1970) followed by a ; is sent

addevent:calendarID:groupID:starttime:endtime:repeatperiod:repeattimes:title:data Add event to calendar. Starttime, endtime, title, repeatperiod, repeattimes, data are in the forms detailed before. Successful response is *0:eventID*

editevent:calendarID:groupID:eventID:starttime:endtime:repeatperiod:repeattime:title:data Edit event with eventID. If you don't want to edit a field leave it blank e.g *editevent:1:1:1::::::Cancelled* will change the data of event 1 in cal 1 in group 1 to cancelled. Successful response is standard

remevent:calID:groupID:eventID Remove an event from a calendar. Response is standard (we could have an editevent command but it isn't necessary)

getanyonewrite:calID:groupID Returns whether anyone can edit a calendar, in form 0:info where info is 1 if they can 0 if they can't

setanyonewrite:calID:groupID:set Set if anyone can add events to a calendar, set is 1 if they can 0 if they can't. **Can only be done by cal owner**

setcalinfo:calID:groupID:info Set the info of a cal to info. Response is success on success or permissions error. **Can only be done by cal owner**

getdeletedevents:calID:groupID:timetamp Returns deleted events in CalID of groupID from timestamp onwards. Response in form
0:eventID:calID; End of list is denoted by *0:timestamp;*

<u>Group commands</u>

listallgroups Returns a list of all groups in the following form *groupid:groupname:canjoin groupid is the groupid, group name is name, canjoin is 1 if you can join, 0 if can't*

mygroups Lists all groups you are a member of. Groups come in the form *groupname:groupid* The end of the list is indicated when the server time (seconds past 1970) followed by a ; is sent

groupinfo:groupID Returns a text description of a group in the form *0:information*

join:groupID Join the group of groupID and thus get all those group calendars when you send list. Response is just a standard server response

leave:groupID Leave the group of groupID. This group will now be listed when you send listnewgroups in case of mistake**.**

creategroup:name:decription Creates a group of group name, with a description of description. Response is *0:groupID*

getanyonejoin:groupid Returns whether anyone can join the group, in the form 0:info info is 1 if anyone can join, 0 if they can't

getdeletedcalendars:groupid:timestamp Returns the calendar ids of all calendars deleted from a group after timestamp. Responses in form *0:calendarID* and the end of the list is indicated by a timestamp response indicating the current server time.

getdeletedgroups:timestamp Returns the group ids of all groups deleted from the database after timestamp. Responses in form *0:groupID* and the end of the list is indicated by a timestamp response indicating the current server time.

Commands for group owners (and 'admins' if they have those permissions)

ownedgroups List all groups you own. Groups come in the form *groupname:groupid*
The end of the list is indicated when the server time (seconds past 1970) followed by a ; is sent

remgroup:groupID Delete the group group ID. Response is standard server response. **Can only be done by group owner**

addtogroup:groupID:username Add user of username to group of group ID. Response is standard. Member has to be one waiting to join

remfromgroup:groupID:username Remove user from group.

groupmembers:groupID Returns all members in group. These are send in the form *0:username1:username2:….:usernamen;*

setanyonejoin:groupID:cando Set if anyone can join or group of if they have to wait for the admin to allow it. Can do is 0 for false, 1 for true. Response is a empty success on success or permissions error. **Can only be done by group owner**

setgrouplinfo:groupD:info Set the info of a group to info. Response is success on success or permissions error. **Can only be done by cal owner**

Other commands

changepass:oldpass:newpass Change password to newpass. Response is standard

listusers List all users. In the response *0:username1:username2:….:usernamen;*

logout Closes the connection. A standard success response is sent if this has ended successfully

Root Commands

These are extra commands that administrator accounts can use.

Adminadduser:username:password Add a user of username and password password

adminremuser:username Remove user username

admin listusers List all users. In the response *0:username1:username2:….:usernamen;*

adminmakeadmin:username Give username admin privileges.

adminremadmin:username Remove username's admin privileges

adminlistadmins List all admins In the response *0:username1:username2:….:usernamen;*

admin listonline List all logged in users. In the response *0:username1:username2:….: usernamen;*

adminchangepass:username:newpass Set username's password to newpass

adminlistgroups:username List all groups username is in. Each group is sent *0:groupname:groupID* end of list is sent by empty success response

Admins can change use all other commands and they can change any calendar/group.