PROJECT ATHENA RELEASE NOTES

OLC Version 2.1

Tom Coppeto

1. Introduction

This document describes the software changes in OLC version 2.1. The first part contains an overview of the user visible changes, and detail on the release as a whole. The release notes are written as a set of changes to the current version. The second section describes the OLC system in general, and is intended to someday evolve into a software user guide. Detail on new features can be found there.

Doesn’t this document look familiar? (But it’s still in scribe.)

2. Changes

2.1. All Clients

2.1.1. Command Parsing

Any argument that does not have a preceding flag or argument flag will be taken as a username, ie: *send edit* will try to send a message to user *edit*. *send joe -edit vi* will attempt to send a message to joe and use *vi* as the editor. Commands which accept additonal optional arguments after the argument flag will not accept a username in that position, ie., *send -edit joe* will send a message to the connected person and use *joe* as the editor (flexibility has its price).

2.1.2. Sending Messages

The *print* option to the *what_now?* prompt has been changed to *list* for compatibility with mh mail sending programs. Any incorrect choice prints a list of acceptable commands. A bug still exists that causes *show* to behave like *send*.

2.1.3. Displaying Messages

2.1.4. Startup Screen

The user’s status is displayed followed by a *press any key to continue* at which point the OLC message of the day is displayed. Somehow this sounds backwards and may change by the time you read this.

2.1.5. Help Files

These have been formatted a bit to reflect command usage and file modification times (they look like man pages).

2.2. *olc*

- A brief command listing is displayed upon startup as a reminder of what to do once inside.
- If a user is not connected upon entering his question, a pitch for the *answers* command is displayed.
- The *what_now?* prompt is displayed after the question has been entered.
- Empty questions (zero length files) will not be accepted.
- The *status* command has changed. *Status* now prints a user's status, see later explanation, as opposed to the queue status (which was usually incorrect). A substitute for this mechanism is in the works.

2.3. *olcr*

- Changing the topic requires the *-topic* argument flag, ie: *topic -t emacs*, for reasons described above.
- Signing on has changed. Please refer to the help file for more info.
- Seemingly strange things will happen if you are asking a question and attempting to consult. See the explanation on clients in the next section.
- *Forward -trash* has been replaced by *cancel*.
- The display returned by list has been altered. Both user login status and question status is displayed for each question. Three characters have been donated to the consultant status. Machine names for users asking questions have been omitted from the listing.
- The *referred* and *pit* question types are no longer presorted. See explanation on lists in the next section for more info.
- Describe has been changed. Please refer to the help file for more info.
- The command *answers* behaves like *stock*. *stock* may disappear.
- The user's machine name is no longer displayed in the list. Three other commands which reveal this information. Consultants' locations are revealed in the list because it is occasionally useful.
- New commands: instance, status. Refer to the help files for more info.
- *olcr* is a link to *olc*- see the section on **Clients**.

3. Manual (supero drafto)

This tends to document the behavior as seen from the client. What the server actually does is another banana altogether. More attention is paid to the new stuff, because it is new.

3.1. Command Parsing

Any incorrect argument usage results in a usage message in additon to an error message. All commands accept a *-help* argument to solely print the usage message without the error.

Name completion works for all argument flags (*-foo*) as well as commands. Most argument flags are intentionally unique to one character, with the exception of flags accepted by *cancel* and *done*. The clients do not print all possible completions of an argument flag upon error. Additonal arguments such as topics or random pieces of text do not complete.

The current error behavior is as follows:

```
olcr> s
Could be one of:
        send
        show
        stock
        status
olcr> done -t
The argument, "-t", is invalid.
Usage is:       done  [<username> <instance id>] [-off] [-title <title>]
                [-topic <topic>]
olcr>
```

Any argument that does not have a preceding flag or argument flag will be taken as a username, ie: *send edit* will try to send a message to user *edit*. *send joe -edit vi* will attempt to send a message to joe and use *vi* as the editor. Commands which accept additonal optional arguments after the argument flag will not accept a username in that position, ie., *send -edit joe* will send a message to the connected person and use *joe* as the editor.

3.1.1. Getting help

Every command has a help file which explains what it does. Each command will also accept a *-help* argument which simply prints a usage message.

3.1.2. Shell Commands

If the first character of the command line is '!', the rest of the line is passed to the shell for evaluation. A silly thing to do is:

```
olcr> !olcr
```

3.1.3. Identifying users

The general format of all commands is:

```
command [user <instance>] [-options]
```

This format works (should) most of the time. Whenever possible, the parser will attempt to accept the command line if the options and specified user are given out of order, but this isn't guaranteed (see above).

Since a user may be asking or answering multiple questions, an id for that particular instance may be required. If the instance id is ommitted, the default of 0 will be used. The instance id is the number that appears in brackets after the username in the listing.

example:

```
olcr> list
User          Status           Consultant                 Status     Topic

joe      [0]  (active/unseen)                                   (0)  emacs
joe      [1]  (active/unseen)                                   (0)  fortran

olcr> topic joe
Current topic is "emacs".
olcr> topic joe 1
Current topic is "fortran".
```

3.2. Sending Text

Commands which involve the sending/editing of text uniformly accept a *-editor* flag. The flag by itself will cause the editor as indicated by the EDITOR environment variable to be invoked. Alternate editors may be specified after the flag (ie: *send -editor ed*).

The *what_now?* prompt is displayed after each editing session. Typing '?' or simply <return> will display a list of options. The available options are compatible to the *mh what now?* prompt.

The clients verify the potential success of these requests before allowing the user to compose the message.

3.3. Receiving Text

Commands that show text (more than a few lines) all accept a *-file filename* argument set to specify an output file other than the screen. *> filename* also seems to work. Messages displayed on screen are done so using */usr/ucb/more*. The text is stored in a file thus it is possible to view previous screens while *more*ing.

3.4. Asking Questions

not implemented.

3.5. Answering Questions

3.5.1. Signing on

There are four different levels at which one may sign on. Signing on tells the server that it may automatically connect you to existing or incoming questions. Various levels exist to control this behavior. Once the olc administrator has given permission for one to sign on, h/se may freely choose any level.

1. *on -first*: This is the front line. Questions are first matched to consultants signed on at this level iff the topic of the question matches a registered speciality of the consultant.

2. *on -duty*: Failing the above, the question is then passed to consultants signed on at this level. These consultants may be connected to any question, but priority is given to questions in the consultant's specialty.

3. *on -second*: This consultant is only connected to questions in his specialty after

failing to match a consultant in the above two levels.

4. *on -urgent*: This is more like the trash disposal- all questions making it down to this stage are connected to consultants signed on at this level.

The on levels are abbreviated in the list as *sp1*, *dut*, *sp2*, *urg*, respectively.

If no argument to *on* is given, the default *second* is used. The level of *on*-ness may be changed at any time, even during a question.

The command *off* may be used to sign off at any time. Signing off does not break any current connection, if one exists.

3.5.2. Grabbing Questions

See the help file on *grab*.

3.5.3. Forwarding Questions

See the help file on *forward*.

3.5.4. Resolving Questions

See the help file on *done*.

3.5.5. Cancelling Questions

A cancelled question implies that no question was asked or the log was garbage and it shouldn't be logged with legitimate questions. These questions may wind up in obscure devices.

3.6. Stock Answers

Stock (also referred to as *answers*) allows one to access the stock answer browser from inside the OLC client by calling the program */usr/athena/lib/olc/stock/browser*. The browser and the text of the stock answers actually reside on the *olc-stock* file system. The OLC clients will attempt to attach this file system if not attached already. Upon failure it will attempt to attach *olc-stock_backup*. (effectively the mechanism thing */usr/athena/olc_answers*). If you have a question or are connected to a question, *olc* will pass the topic information to the browser to display the menu of that topic, else the top level menu is displayed. This behavior can be overidden with the *-topic* option. The browser is not at this time part of this OLC release and.. (in other words, I know little about it except what arguments it takes).

If a stock answer was saved into the default file from inside the stock answer browser, the good ol' *what_now?* prompt will appear and the *olcr* user will have the option of sending the stock answer directly to the connected user or into his own log.

3.7. List

The default display (and currently there is only a default display) returned by the *list* command displays generic information about a question, and people answering or asking questions.

`olcr>` *list*

```
User           Status           Consultant                 Status     Topic

olcdev   [0]  (active/unseen)                                    (0)  ls
testuser [0]  (logout/pending)                                   (1)  hesiod
                                tjcoppet@PICASSO.MIT [0]  (sp2)
```

The column on the left displays the username, the instance id, and two status strings. The left status string referes to the status of the user and the right status string refers to the status of the question. The user status may be one of active, logout, or unknown and the question status may be one of *active*, *unseen*, *pending*, *done*, *cancel*, *refer*, or *pickup*. Those signed on will not be automatically connected to questions in the refer or pickup state.

The middle column displays the list of people answering questions or signed on and ready to answer a question as shown above. The machine name and instance id are also displayed. The consultant's status string is a three letter abbreviation for level at which the consultant is signed on *off*, *sp1*, *dut*, *sp2*, *urg* (please see the section on signing on for more explanation here).

By default, *list* displays all questions for your given queue. Options may be given to *list* to filter out certain types of questions.

- *list -status* `<status>`: You may specify one out of most of the available status strings. Simply using *list -status* displays a list of these strings.
- *list -topic* `<topic>`: Like the above, you can filter out questions of a certain topic. However to see a list of available topics, *topic -list* mus be used. (only I could manage to write code that behaves like this).
- *list -user* `<username pattern>`: You can list a specific user or patterns of usernames using this option. If you want a list of all users to choose from, don't specify any options. Please don't leave out the *-user* argument flag and expect me to explain what is going on.

3.8. Question descriptions

A question description is a summary of the question. The *describe* command can be used to summarize a question. The output is divided into two parts: the stuff you normally see with *list* and an optional comment. Best to explain with an example:

```
olcr> list
User           Status           Consultant                 Status     Topic

tjcoppet [0]  (active/active)  tjcoppet@PICASSO.MIT [1]  (off) (1)  ls
carla    [0]  (active/active)  beth@KLEE.MIT.EDU    [0]  (off) (1)  ls

olcr> describe
User:   user Tom Coppeto (tjcoppet@PICASSO.MIT.EDU)
Status: active/active
Connected to:  consultant Tom Coppeto (tjcoppet@PICASSO.MIT.EDU)
Topic:  ls              #connected: 1
Note:

olcr> describe -note
Enter your note (64 max chars):
> I don't know, I just work here.
Question description change successful.
```

```
olcr> describe

User:   user Tom Coppeto (tjcoppet@PICASSO.MIT.EDU)
Status: active/active
Connected to:  consultant Tom Coppeto (tjcoppet@PICASSO.MIT.EDU)
Topic:  ls              #connected: 1
Note:   I don’t know, I just work here.

olcr> list -comments
User          Status          Consultant                Status     Topic

tjcoppet [0]  (active/active)  tjcoppet@PICASSO.MIT [1]  (off) (1)  ls
        [I don’t know, I just work here.                                  ]
carla    [0]  (active/active)  beth@KLEE.MIT.EDU    [0]  (off) (1)  ls

olcr> describe -comment
Please enter your message. End with a ^D or ’.’ on a line by itself.

I’m a pepper
You’re a pepper.
Wouldn’t you like to be
a pepper too?


.... over my dead body.
.

What now? (type ’?’ for options): send
Question description change successful.

olcr> describe

User:   user Tom Coppeto (tjcoppet@PICASSO.MIT.EDU)
Status: active/off
Connected to:  consultant Tom Coppeto (tjcoppet@PICASSO.MIT.EDU)
Topic:  ls              #connected: 1
Note:   hello world!

Additional comments:

I’m a pepper
You’re a pepper.
Wouldn’t you like to be
a pepper too?


.... over my dead body.

olcr>
```

The additional comments may be used for extended notes, but are limited to 512 characters and are only viewable from *describe*. The notes are limited to 64 characters in length but are also viewable using *list -comments* as well. Currently the only way to clear these two pieces of text is to enter nothing.

3.9. Notifications

The server currently sends out two types of notifications; personal messages and broadcasts of general interest. Personal messages are sent whenever there is any major change of state for a particular user:

- new messages
- connection to another user
- break in connection (*forward, done, cancel*)
- signal to end conversation (*done, cancel*)
- connected person has logged out

The personal messages will attempt to be transported via *Zephyr* but wil fall back on *writed* and company upon failure. Of course failure, in the *Zephyr* sense may mean logged out or not subscribing to messages. The acknowledgement from these notices determines a user's login status. Because of the fragility in contacting the host's *writed*, a time out of ten seconds exists. If this timeout occurs, the user's status will change to *unknown*.

The broadcast messages are only sent via *Zephyr* (class *OLC*). The various instances are:

- *new_question*: when a new question arrives that is not immediately connected to someone
- *forward*: questions that are forwarded and not immediately connected to somebody else
- *nol*: signing on triggers these notifications
- *resurrection*: a user who has a question somewhere in the queue and logs back in after the server knows he has been logged out
- *lonley_hearts*: users who send messages and are not connected [1] *syserror*: When the server is not using syslog it may shove some status messages through this instance. It is for entertainment value only.
- *syslog*: I don't know why this exists or what it is used for. This appeared when I grepped the sources for the broadcast function.

Wherever it makes sense the server inserts the username and topic of the question for clarification. The topic of the question is also shoved into the opcode field so people can filter out their favorite topics.

I am told that OLC has complete disregard for windowgram fields (or atleast is not taking advantage of them). This shoudl be looked into.

3.10. When there are more than one of you

When you attempt to grab or ask a question, and you are already *doing something* in OLC, olc will prompt you if you *wish to create another instance* to perform the operation. If

[1] The typo intentionally exists because no one realized it before lots of people configured their zwgc.desc files.

you say yes, another *instance* of yourself will be created.

An instance is like a window. However you can only have them created on a need basis and you may be restricted as to how many instances you may have. In a window environment, you can only type into one window at a time. In OLC, you can transact in one instance at a time. For example (I resisted using the word instance), if I am currently asking a question and I wish to grab a question, I can have another instance created to answer the question I grabbed. All commands I issue will be intended for the grabbed question. I can use the *instance* command to change my instance id to that of my original question, and from then until I change my instance again, all commands I issue will be intended for the question I am asking.

An instance is specific to either asking or answering a question and the instance id is used to identify which instance you are referring to. When you are through with your instance, it is deleted. A new instance will be created the next time you need it.

The instance id's are simple numbers; running 0,1,2,3, etc. If you have instances 0, 1 & 2, and 1 is deleted, the next time you need an instance created, its id will be 1 again. [2]

The commands *status*, *who*, *instance* will be usfeful in managing what you are doing. *Status* prints out an entire summary of what you are doing in OLC. If you have more than one instance of yourself, it will also indicate what instance you are currently *in*. *Who* prints out a short message of what you are doing in your current instance. And *instance* may be used to display or change your instance.

Here is an example:

```
olcr> list
User          Status           Consultant                Status     Topic

tjcoppet [0]  (active/unseen)                                   (0)  accounts
carla    [0]  (active/unseen)                                   (0)  ls

olcr> grab carla
Would you like to create another instance to grab this question? yes
User grabbed.
You are now tjcoppet (1).

olcr> list
User          Status           Consultant                Status     Topic

tjcoppet [0]  (active/unseen)                                   (0)  accounts
carla    [0]  (active/active)  tjcoppet@PICASSO.MIT [1]  (off) (1)  ls

olcr>  i(who)
You are currently connected to Carla Fermann (carla@MICKEY-MOUSE.MIT.EDU)

olcr> status
Status for: Tom Coppeto (tjcoppet@PICASSO.MIT.EDU)

   instance     status     connected party    topic
```

[2]The instance id's are generic tags. We originally had tags of red, orange, yellow, green, blue, indigo, and violet, but didn't want to restrict ourselves to seven possible instances.

```
   [0]          unseen     not connected      accounts
-> [1]          off        carla     (0)      ls
```

olcr> *instance*
You are tjcoppet (1). Have a nice day..

olcr> *instance -i*
enter new instance (<return> to exit): 0
You are now tjcoppet (0)

olcr> *status*
Status for: Tom Coppeto (tjcoppet@PICASSO.MIT.EDU)

```
   instance     status     connected party    topic
-> [0]          unseen     not connected      accounts
   [1]          off        carla     (0)      ls
```

olcr> *instance -i 1*
You are now tjcoppet (1)

olcr> *forward*
Question forwarded.
You are tjcoppet (0), again.

olcr> *status*
You currently have an "accounts" question in the queue.

olcr> *list*

```
User          Status           Consultant                Status     Topic

tjcoppet [0]  (active/unseen)                                  (0)  accounts
carla    [0]  (active/pending)                                 (1)  ls
```

olcr>

The idea of the *current instance* is client specific. This means you can run multiple copies of the client and have each client run in a different instance. Or, if you like, you can run multiple clients all in the same instance.

3.11. Clients

Essentially, we currently have one (1) OLC client. This program (though currently missing an important item) is intended to handle all aspects of answering or asking questions. However there is an interface problem with making lots of commands available to people who are not allowed to use them, having compilcated help descriptions, or requiring the user to issue another command in order to ask his question. For these reasons we have a watered down version of this client which does some hand holding during some operations, contains simpler help files and command usage lines; it is basically intended for only asking questions. We have called this client *olc* and for backwards compatibility, the generic client is called *olcr*. They are the same program.

Don't fret when you are asking a question and start up *olcr* and find that you can continue your question, or better yet, answering a question and start up *olc* and find that you can continue to provide help (though this procedure is not recommended, can be quite confusing).

4. Gripes

Mail in bugs to bug-olc@athena. If possible please indicate some sort of version number if you can find one and where you got the client since there are usually a few around and some are obsolete. This is where everything is my fault.

Comments, criticisms, or new ideas are welcome in the olc-discssion discuss meetings, or mail to olcd@athena (it won't be routed to the nearest OLC server). This is where evrything will be my fault.

If you are a consultant you are probably not allowed to talk to me directly, so ask carla how you should deal with bugs & comments.

5. OLC help files

Just a hardcopy of help files for reference. I have only included the help files referred to by the *olcr* client since they are of more general interest.