Actim - Manual

Joseph Kuan

October 16, 2001

Contents

1 Introduction

Actim is a prototype application that allows file transfer and non-interactive command line operations via the internet. At the moment, it uses email (IMAP only) and is in alpha stage. This applicaton is not supposed to be treated as a normal connection program, such as telnet or rsh.

The program is multi-threaded and also provides a command line and graphical user interface.

2 What for ?

In my opinion, I wrote this application for use behind the firewall:

1. For example, your office workstation is behind a very secure firewall. The only way that you can communicate **from** your home PC is through email or http. If the company has socks servers setup, then you can use remote port forward with ssh to establish a connection to your home IP address. Suppose you dial up to an ISP that gives you a **dynamic** IP address. The only way is to publish this IP address by email or put it on your own web page. Then set up your office workstation continuously detecting the new IP address and launch ssh to connect to your home PC.

2. If your organisation only allows email and http, and is without socks server to allow outgoing connections, then you can use this tool to gain some access but in a slow and inconvenient way.

3. Imagine your home PC is connected to the internet at all time and you hardly connect to your home PC from work. Instead, you can switch on or off all the listening services (such as ftp server, telnet server, etc) on your home PC with this application. For instance, you can send an actim event with a command ID, This active server recognises the ID and activates all the listening services. Then the user can telnet, ftp, ssh, etc to the home PC. When the user is finished, the user sends another actim event to switch off all the services. This is one way to protect your home PC. Of course, in this scenario, you must connect to your ISP mail server at all times.

3 Architecture

Diagram 1 summarises the design of actim. The server contains multiple threads which all cooperate in a master-workers queue manner. Some threads are generated dynamically where there is an actim event. The actim server initially creates two threads; event task handler and result handler.

To summarise, the event task handler becomes active when there is a request from the client. If the request is a server command, the event task handler will create a thread to execute the command and close the thread. If it is an event, then it will send the event in email and create a thread to poll for actim result email. Any result that is generated from the local server or has arrived from remote server is first forwarded to the raw result queue. The result handler listens on the raw result queue and either puts the result back to the client or into the hold result queue. This hold result queue acts as an asynchronous feature for the client so that the user may decide to disconnect after launching the request. Then the result is fetched later on.

The picture shows the following steps starting from how a request is being formed to results returned from a client:

1. (Local) First the client creates a request event object from the arguments and configuration setup.

2. (Local) Connection is established and the request object is sent to the server.

3. (Local) The server recieves an incoming object and forwards this to the incoming queue.

4. (Local) The event task handler first checks whether the request is event or command type.

   (a) (Local) If this is command type (Point A in the diagram), the event handler creates a new thread, Command executor, to execute the command and returns the result object into the raw result queue.

   (b) (Local) Otherwise, it checks whether the event is generated locally. If so, then it finds out if there is any result inside the event object. If there is no result, this means the event is just submitted from the client. If then, serialises, encrypts and sends the event to the destinaton (Point B in the diagram). Meanwhile it starts a new thread to poll for any return email.

   (c) (Local) If the event has result, this means the event has been executed from the remote site and returns the result back. Forward it to the raw result queue. (Point C in the diagram)

5. (Remote) On the destiny side, actim is also running. The poll media thread picks the email up because of the same subject. Then it unpacks the email content, decrypts and unserialises into an object and forwards to the incoming queue.

6. (Remote) The event task handler thread fetches this object from the incoming queue. This object can originate from two different status. It can be a request generated remotely or it is generated locally and returns

from a remote site with a result. In the former case, go to the next step. Otherwise go to the last step.

7. (Remote) A new thread is created, event executor, which invokes the method request by the event (Point D in the diagram) and sends the result back to the sender. Back to step 4.

8. (Local) The result object is then forwarded to the client and the client reports the result.

4 Quick Start - Command Line

Suppose that you have successfully installed the application on your home PC and office workstation.

Assume the name of your home PC is A, and office workstation is B.
You want to send a command from machine A to machine B.
Suppose the email account name on machine A is 'accountA' and the password is 'pswdA'.
For machine B, the account and password are 'accountB' and 'pswdB' respectively.
Before you start, you need to setup the following options on both machines A and B:

Client ($HOME/.actimrc)	Server ($HOME/.actimdrc)
emailAddress	IMAP4_server
emailAccount	SMTP_server
emailPassword	mailSubject

Note that the mailSubject must have the same value at both ends. The emailAccount or emailPassword can be left with empty string and the user will be prompted for password. The user can also decide whether the encryption key, eKey, is put into actimrc file or entered from the prompt. The key must have the same value on both machines A and B in order to encrypt and decrypt the email content.

Make sure *actimd* is running on both A and B machines.

```
actimd
```

Then request to start a polling for emails from an user's mailbox. Such that on machine B, enter:

```
actim -C POLL EMAIL SETUP_USER
```

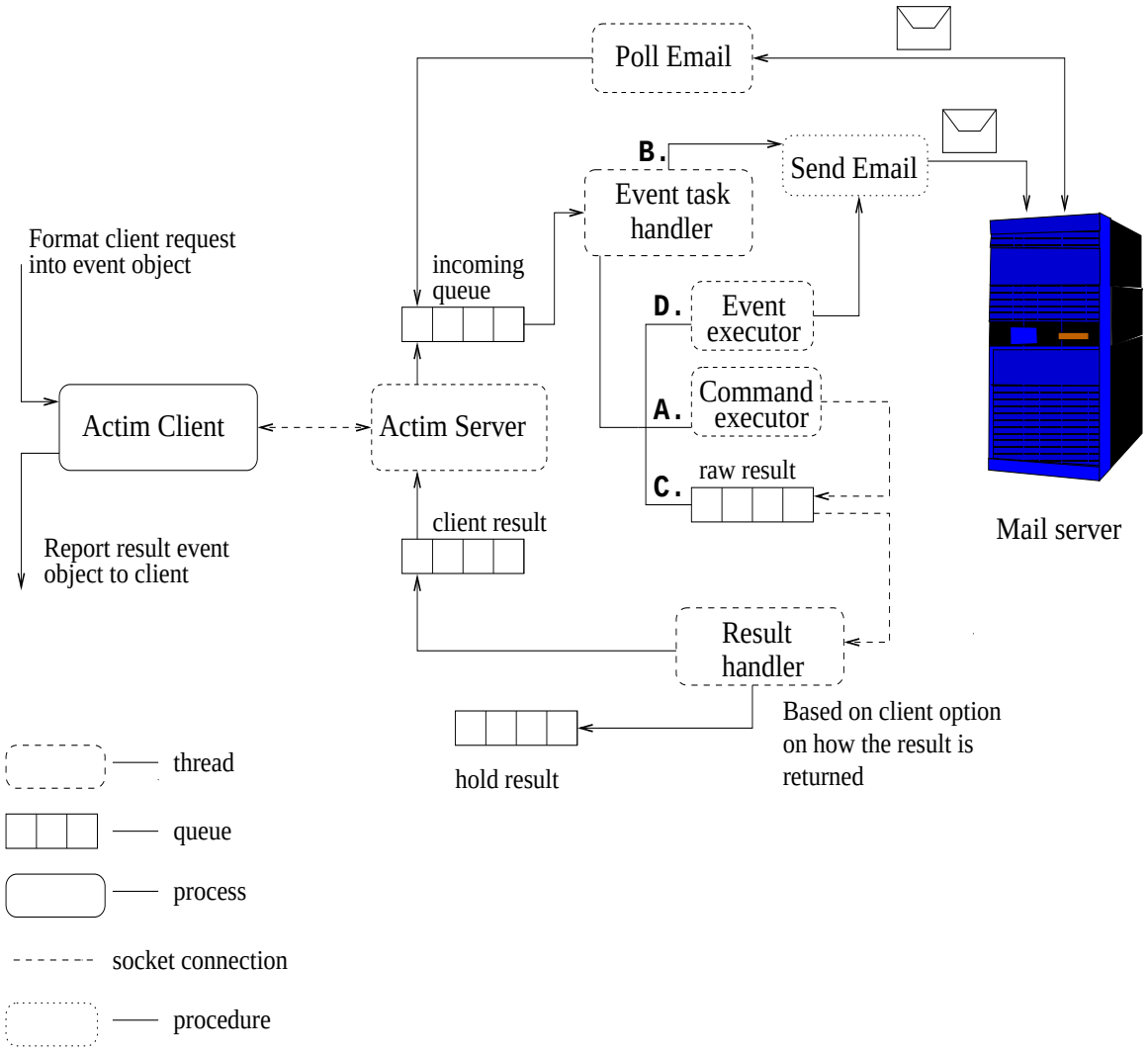

Figure 1: Design of Actim

Note that once polling for emails is started, it will continue to poll forever until the user requests to end the polling on that machine. (END_USER is not implemented yet)

Then on machine A, you can choose to submit a full event request which automatically start up polling for accountA's mailbox then launches a request. In order to do this, you must setup the emailAddress and emailAccount options as the table mentioned above.

```
actim -E -e "ps -ef" -d "B@wabot.com" -R
```

Alternatively, if you don't want to expose your emailAccount and emailPassword in your actimrc file, you need to start polling explicitly. Then submit your request.

```
actim -C POLL EMAIL SETUP_USER accountA pswdA
```

If actim displays the return message of a successful login to IMAP4_server, then on machine A enter:

```
actim -E -e "ps -ef" -d "B@wabot.com" -R
```

Then sit back and do something else while waiting for the result.

However instead of starting to poll explicitly first, if you have your emailAccount and emailAddress setup in the actimrc file, then you can invoke an event request anyway. Once the actimd server receives an event, it checks whether a polling has been started. If not, start one and send the email request.

If you want to request XML output, then you can do the following:

```
actim -C QUEUE_INFO ALL --xmlOut
```

The XML output should look similar to the following:

```
<Result>
    <ReturnCode>0<\ReturnCode>
    <Type value="0">Command<\Type>
    <Output>
Hello message from 'pebbles'
    <\Output>
    <Error>
    <\Error>
<\Result>
<Result>
    <ReturnCode>0<\ReturnCode>
    <Type value="0">Command<\Type>
    <Output>
<Queue length="0">CLIENTS_RESULT<\Queue>
```

```
<Queue length="1">SUBMITTED<\Queue>
<Queue length="0">INCOMING<\Queue>
<Queue length="0">RAW_RESULT<\Queue>
<Queue length="0">HOLD_RESULT<\Queue>
    <\Output>
    <Error>
    <\Error>
<\Result>
```

5 Graphical Interface

The actim graphical user interface is invoked by the program, *gactim*. The GUI provides an easy interface for setting up or modifiying configurations on client, server and security, a simple server status control and also launches events on file transfers and command line calls.

Diagram 2 shows the main window screenshot when the *gactim* is called. The main window is generally categorised into 5 functional areas which are indicated in the diagram. The program will search for $HOME/.actimrc and $HOME/.actimdrc files for configuration. If the files do not exist the program will use the default configuration. Then the user can use the gui program to create configuration files.

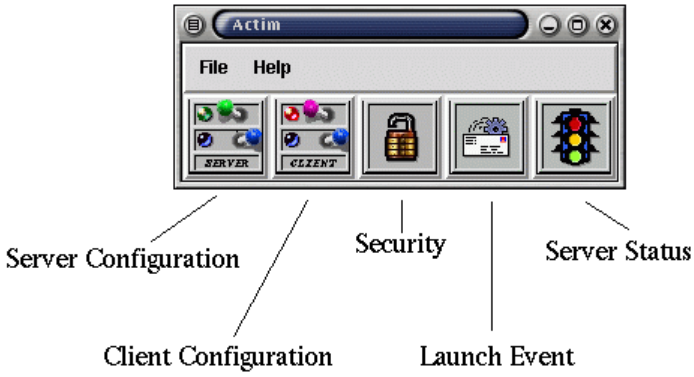

Figure 2: main diagram

5.1 Server & Client configurations

Diagrams 3 and 4 demonstrate the server and client configurations interface respectively. For detail of setting up configurations, see sections 4, 8.1 and 8.2.

The reset button will reset any modification back to default settings. Once the modified configurations are saved, new default settings are defined.

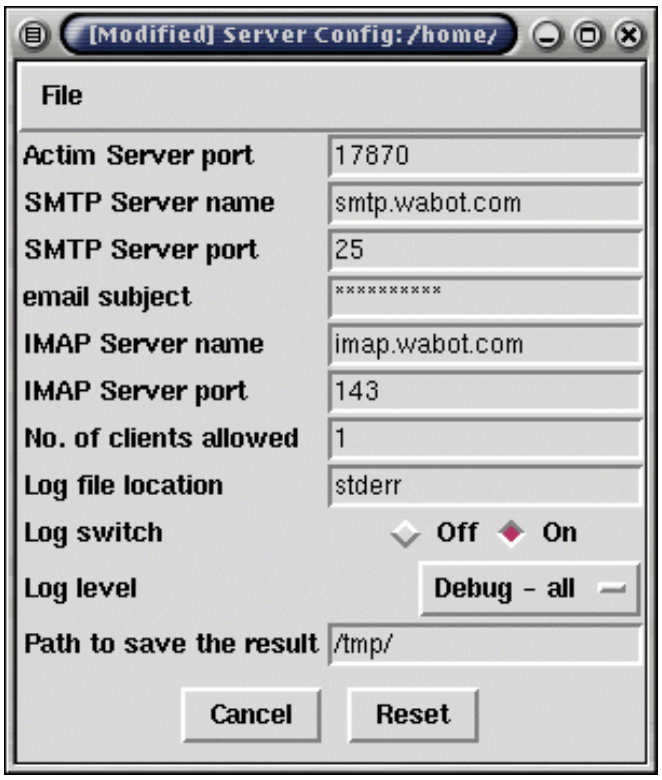

Figure 3: Server Configuration

5.2 Security configuration

Although the security options are located in the same file as server configurations, it is better to separate them into two entities for different purposes. Figure 5 shows the security configurations. When the service time is switched on, the user is able to specify the time for accepting incoming events. The time is in the format of HH:MM:SS. The command line editor button launches a command line editing dialog which lets the users add, delete, modify and test the command lines.

Diagram 7 demonstrates the command line editor dialog and a command line is being entered to test whether it passes any of the command lines.

For instance, the diagram shows there are three restricted command lines defined which are displayed as: *commandID: commandPatterns*. A command

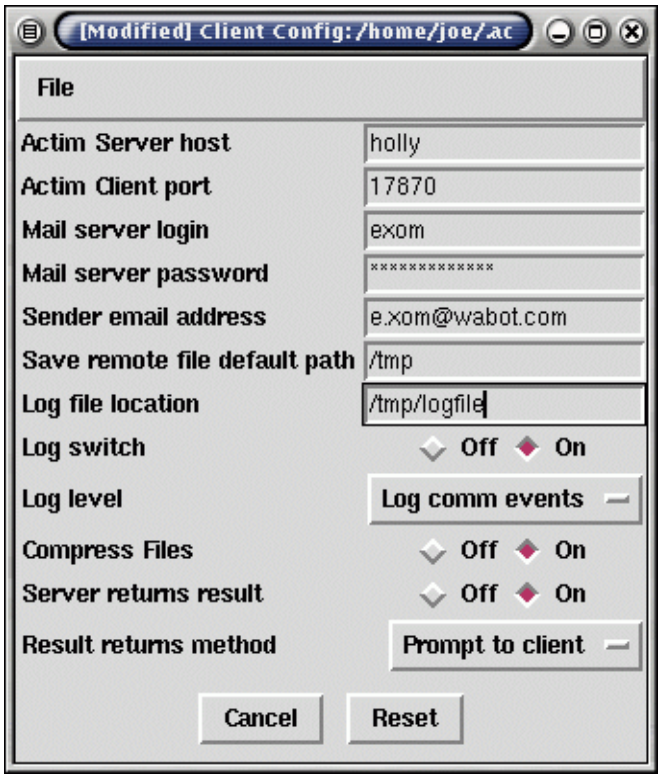

Figure 4: Client Configuration

line is entered and the user clicks on the "Test" button. The matching result indicates which command ID is passed. Screenshot 6 illustrates a command line test failure.

To add a command line, the user must enter a unique command ID and a command line pattern, then press the "Add" button. The pattern can be a normal command line or a regular expression. To delete a command line, select a command line and click on the "Delete" button. For modifying a command line, select a command line by double-clicking which fills the command ID and command line entries. Then hit the "Modify" button once the change is done. The user must remember to save the new security options including the command lines on the "File" menu from security configuration window (diagram 5). The restricted command line option is saved with *rcmdl_* prefix in front of the command ID.

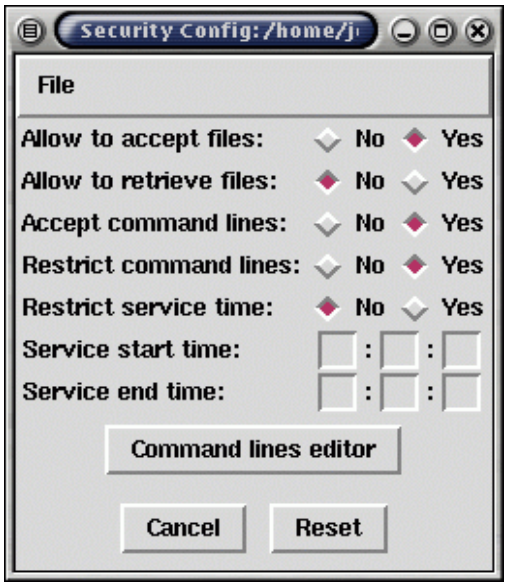

Figure 5: Security Configuration

5.3 Launch events

Figure 8 illustrates the user interface to activate a file transfer event. The user can choose either to send or receive a file via email. The entry labels will change and the browse buttons are (de)activated. See screenshots 8 and 9.

When the user selects command line mode, the window will be modified to command line user interface as shown in figure 10. In the current release, the GUI does not support command line event by specifying command ID (see 9.3.2).

Once all the correct values for the event are entered, the user can hit the "Send" button which connects to the actim server and transmit the event via email. The program, *gactim* checks whether any polling has already started. If not, the application will prompt to the user to enter a symmetric encryption key, shown in figure 11.

The key is used to encrypt a pickle event before sending via email. The key is not sent as part of the email. The remote server is expected to know the key when receiving the email.

The result sent back from the remote server is expressed in XML. The user interface then parses the result and present into output, error and log columns which are displayed when the "Show results" button is pressed. Figure 12 demonstrates that the event result is shown on the user interface.

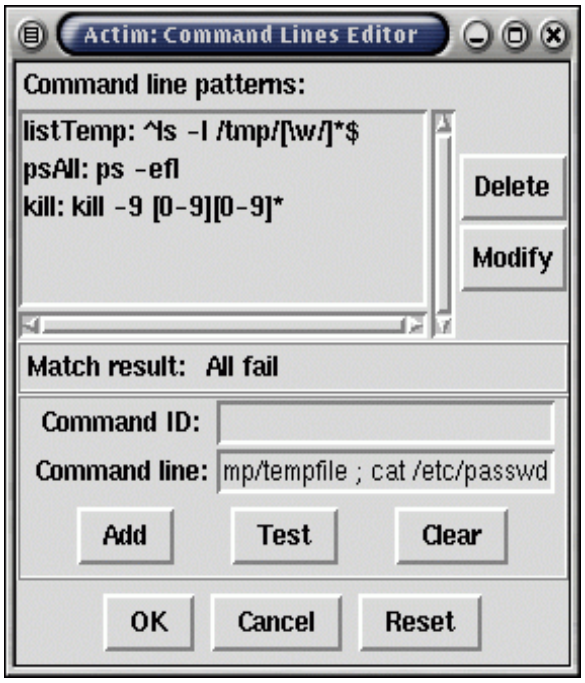

Figure 6: Command line test fails

5.4 Server status

The user can start the actim server by simply clicking on the "Start" button. If the server is started successfully, the bulb next to the server lights up. This also applies to the polling button. As soon as the server is on, the user can switch the polling service on and off. The polling service does not need to be started manually. As soon as the event is launched, the server will automatically detect whether the polling service has been started. If not, the server will initiate the polling by itself.

In the current release, it does not support to switch off polling. The only way is to restart the server. If any configuration is modified and saved during the server is running, the user must restart the server to take effect of the new configuration.

6 Client & Server

Actim comes with client and multi-threaded server programs, *actim* and *actimd* respectively. Once the server is started, the user invokes *actim* to launch a request to the server. This request can be a command line or files transfer. As

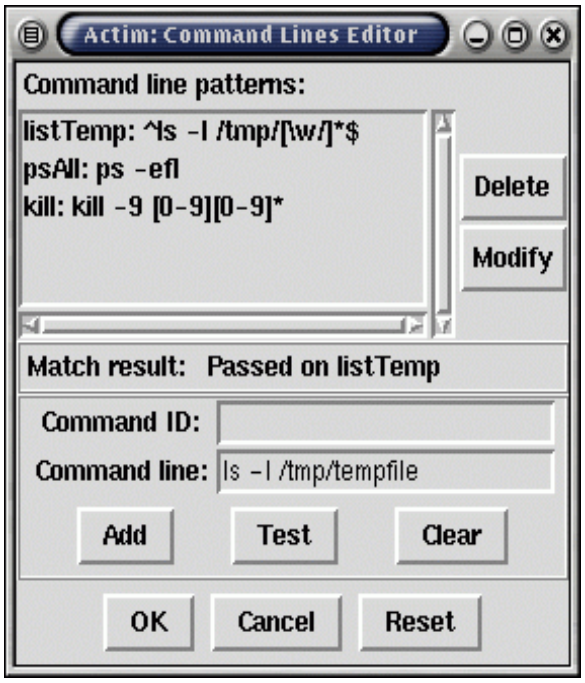

Figure 7: Command line test passes

well as requests, *actim* can also query the server status.

Currently, the main role of the server program is to send and poll for Actim emails requests and execute them, then send the results back. Once an Actim email has been executed, the Actim email will be deleted from the user's mailbox. The server is capable of holding the results in a queue or saving the results into files. After a while, the client can reconnect to the server and fetch the held results back. Alternatively, the client can submit events and wait for the results to return, then disconnect.

7 Security

7.1 Policy

Unfortunately, I only have experience with Unix systems. Hence these security issues are applicable to Unix systems only, not Windows.

- The application does not support public key cryptography, because both ends are supposed to be used by the same user.

Figure 8: Send file event

- No password nor encryption key are transmitted. There is no need because of the same reason above. Both ends are supposed to know the same key.
- The email content is encoded with the symmetric key encryption.
- No version info is used throughout the product.
- The actim server is not allowed to be started by root user.
- The program, *actim* does not start if .actimrc file is world readable. Only owner readable is allowed.
- The program, *actimd* starts if .actimdrc file is owner readable (0400 or 0600). If .actimdrc file is not owned by the caller, then the file must only have read permission granted (0444, 0644, 0604, etc).
- All the file transfers and command line executions are logged through syslog under UNIX platforms.
- Adding new tweaks, startTime and endTime, allows the user to specify what time of a day that the server is allowed to look for any new actim email.
- Adding new tweak, allowExecution, allows the user to config the server to accept command-line events from remote sites.

Figure 9: Receive file event

- Adding new tweak, allowFileReceive, allows the user to config the server to allow files received from a remote site.
- Adding new tweak, allowFileSend, allow the user to config the server to allow files send to a remote site.
- Adding new tweak, restrictedExecution, allows the server to execute certain command-lines. This is only if allowExecution is on.

7.2 Risks

Users should be aware that they are exposed to risks outside the firewall. The log messages from syslog must be monitored frequently. Restricted executions should always be applied and only enable file send and retrieval if it is absolutely necessary. Actimd should be started as another used id and preferably without home directory. Otherwise outsiders can search through users' home directories, especially the .actimrc file. For example, the actim rpm package creates a new user, **actim**, and should run *actimd* as:

```
[joe@pebbles]$ su -c 'actimd -c /tmp/.actimdrc.joe' actim.
```

Then users can request the server to poll for actim emails:

```
[joe@pebbles]$ actim -C POLL EMAIL SETUP_USER
```

Figure 10: Command line event

Figure 11: Prompt for encryption key

Suppose a restricted execution is setup as 'ls -l [\w/]*$'. If an outsider tries to do 'ls -l /home/joe', then it will fail even though it satisfies the regular expression. The risk of restricted commands is depends purely on how strict the regular expression is. Alternatively, just make them as complete commands.

8 Configuration

All the possible default configurations for client and server are located in ClientDft.py and DaemonDft.py respectively. Users are not supposed to modify configurations in these files. Under UNIX platforms, personal configuration options are in $HOME/.actimrc and $HOME/.actimdrc for client and server. Any option specified in .actimrc and .actimdrc will override the configuration values in the ClientDft.py and DaemonDft.py files. For Window platforms, these go to $ACTIM/actim.ini and $ACTIM/actimd.ini.

Actim uses ConfigParser module in python to parse configuration files. All the configuration options must be presented under the, [DEFAULT] section and

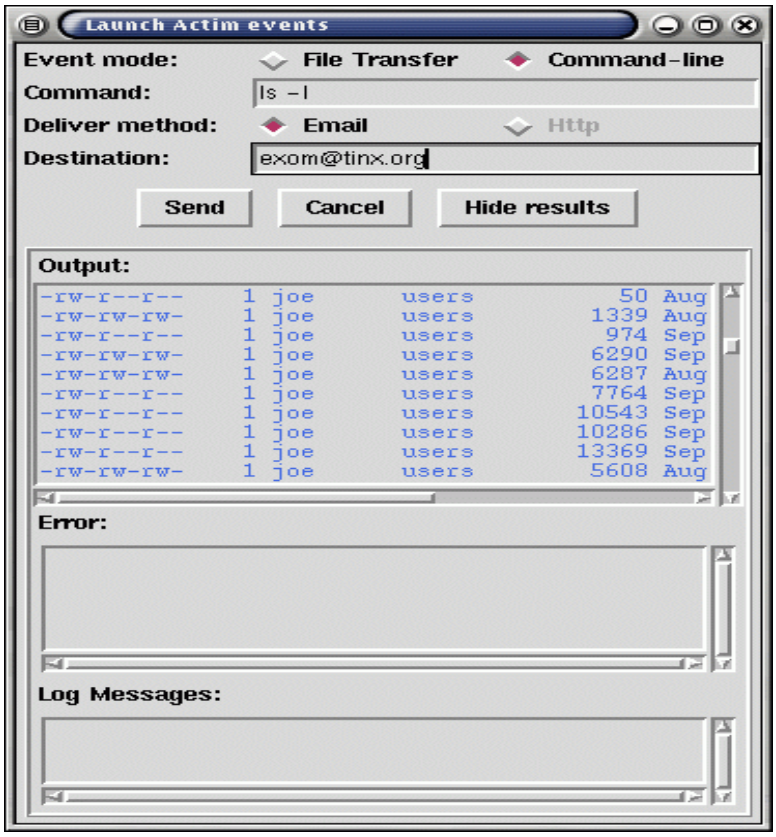

Figure 12: Command line event with result

options are in name and value pairs. ie

```
[DEFAULT]
requestResult=1
emailAccount=eXom
emailPassword=fooBar
emailAddress=e.xom@ntlworld.com
```

8.1 Client configurations

All the configurations can be found in ClientDft.py. However, I strongly recommend you to have your own .actimrc with read only permission to yourself (not world readable). Your .actimrc file should consist of the following configurations:

```
[DEFAULT]
```

Figure 13: Server status started and stopped

```
emailAccount
emailPassword
emailAddress
```

The following is a list of all the configurations for clients:

server	where the server is going to run
port	the port number talking to the server
requestResult	requests all the results of events that are submitted. Possible values are 1 - on, 0 - off.
resultAction	this is valid if the requestResult is 1 Possible values are: - ResultCode.SAVE_RESULT - ResultCode.HOLD_RESULT - ResultCode.RESULT_BACK
transmitMedia	default way to delivery the event. only allows Delivery.SMTP at the moment.
logStatus	1 - switch on the logging, 0 - switch off.
logfile	defines where to write the log file, actim.log Possible values are: - stdout (Log messages write to standard output) - stderr (Log messages write to standard error) Otherwise try to open as a log file
logLevel	grandularity of loggings. Please refer to LogLevel.py for all the possible settings.
targetPath	indicates where the files are going to be stored in file mode. This depends on -p or -g option. -p means target path on the remote machine, and local path for -g.
compressFiles	in order to speed up the delivery, the file(s) to be received or sent is(are) compressed before the delivery.
eKey	encryption key to encrypt the serialised content before sending via email
xmlOutput	all the results are printed out in XML format. The purpose of this option is to make integration to other applications easier. Possible values are 1 - on, 0 - off

8.2 Server configurations

8.2.1 Default options

All the options mentioned in this section are under the '[DEFAULT]' heading.

For server configurations, some options are the same as the clients config, such as:

server port connection logStatus logfile logLevel

Here are the specific server configurations:

pollEmailInterval	the number of seconds polling for new Actim emails
mailSubject	the mail subject of Actim emails
SMTP_port	the port number for smtp
SMTP_server	the server name that accepts SMTP connections
IMAP4_server	the server name that accepts IMAP4 connections
IMAP4_port	the port number talking to the imap server
saveRsltDir	when the client choose the Result.SAVE_RESULT option, this option defines where the result is saved to
numConnection	the number of client connections that the server accepts

Meanwhile I strongly recommend having your own read only $HOME/.actimdrc that contains the options **SMTP_server**, **IMAP4_server** and **mailSubject**.

8.2.2 Security options

All the options mentioned in this section are under the '[SECURITY]' heading.

Here are the specific server's security configurations:

allowExecution	1 - allow accepting command line execution requests from outside 0 - disallow any command line execution activity.
restrictedExecution	This option is only effective if allowExecution is on. 1 - applies restricted execution on command line requests 0 - No restriction on any command line If this is on, at least one valid rcmd_xxxx option must be defined.
[rcmdl_xxxx]	Must define this option when restrictedExecution is on. Define restricted command line. This option can accept either a complete command or a command in regular expression. xxxx can be any alphanumeric characters.
allowFileSend	1 - allows accepting incoming requests to put files onto the server 0 - disallows any sending file request
allowFileReceive	1 - allows accepting remote requests to retrieve files 0 - disallows any retrieve file request
serviceTimeLimit	1 - Sets the actimd server to accept requests within a certain time 0 - Accepts requests at any time. when this option is on, both startTime and endTime options are needed to be defined
[startTime]	Must define this option when serviceTimeLimit is on. Time that starts accepting any requests Must be in HH:MM:SS format
[endTime]	Must define this option when serviceTimeLimit is on. Time that stops accepting any requests Must be in HH:MM:SS format

8.2.3 Restricted command lines

Restricted commands are defined with rcmdl_xxxx options. These commands can be expressed in complete command line or in regular expression. When actim server receives a command line request, first it will check all the current configurations and see whether it has any restrictions in running command lines. If so, then it will perform a regular expression match on each rcmdl_xxxx value to see whether the incoming command is allowed.

However, care must be taken when defining the rcmdl_option. Here are some brief guidelines:

- Instead of defining a complete command line, eg. 'ps -efl', users should

define as rcmdl_psAll=^ps -efl$ instead.

- When expressing command lines in regular expression syntax, users should disallow any multiple command characters, such as ';' and '|', unless it is necessary. For example, instead of putting rcmdl_listCVS=ls -l /home/cvs/*, users should do rcmd_listCVS=^ls -l /home/cvs/[\w/]*$ instead.

  The rcmdl_xxxx option also has another feature. The name after rcmd_ prefix can act as a command id. With the above example, users can call the command id remotely with **-i** option. For example,

  ```
  actim -E -i listCVS -d e.xom@ntlworld.com.
  ```

9 Usage for *actim*

9.1 Commands categories

- **-C** – Server commands. Use for finding server status.
- **-E** – Event commands. An event can be a command line or files transfer request.

9.1.1 Help commands

- **-u** – usage help
- **-h** – a more detailed help page
- **-s** – displays all the current default setup

9.1.2 Misc commands

The misc command can be specified either before or after a well-formed server or event command.

- **-R** – Wait for results to be sent back. This only applies to event commands, not server commands. For event commands, if this is not specified, the option is referred to requestAction value in .actimrc, then ClientDft.py
- **-c** file – specifies the configuration files instead of /etc/actimrc or $HOME/.actimrc
- **-l** file – specifies the log file, override /etc/actimrc or $HOME/.actimrc
- –**xmlOut** – output results in XML format

9.2 Server commands

All the command names and arguments are case-sensitive.

- QUEUE_INFO <queue name | ALL>
- SHOW_CONFIG
- FETCH_RESULT <account name>
- POLL <MEDIA> <OPERATION> [<account name> <password>]

Actimd has a number of queues with threads managing tasks. The QUEUE_INFO command is to request the server to report any task in a particular queue. The queue names can be:

- HOLD_RESULT
- RAW_RESULT
- INCOMING
- CLIENT_RESULT
- SUBMITTD

The keyword **ALL** represents all the queues. The only queue that normal users need to be aware of are HOLD_RESULT.

SHOW_CONFIG command requests the server to report all the current configuration values.

If there are results configured to put into HOLD_RESULT queue, then FETCH_RESULT retrieves all the results in the queue that matches the account name.

POLL is to instruct the server to start/end a polling action for a user's email account. MEDIA can only have the value EMAIL at the moment. As for OPERATION, this can be SETUP_USER, END_USER or CHECK. SETUP_USER starts the polling and the END_USER (should be supported soon) ends the continuous polling. CHECK command is to detect whether a polling has been setup.

Examples of server commands

```
actim -C QUEUE_INFO ALL
actim -C QUEUE_INFO HOLD_RESULT
actim -C FETCH_RESULT "e.xom@ntlworld.com"
```

9.3 Event commands

As mentioned before, an event request can be either command-line or files transfer. Each different operation requires different subsequent arguments, such that:

actim -E <-f file [-f file]* | -e command_line | -i commandID> ....

9.3.1 File mode

The following shows the format of a file transfer event command:

actim -E -f file [-f file ...] <-p | -g> [-m method] [-t target_path] -d dest

- **-f** file – file mode, depends on option -p or -g which respectively indicates the file is either local or remote
- **-p** – put mode, meaning the file specified is local and transfers to a remote location
- **-g** – get mode, meaning the file specified is remote and transfers to a local machine
- **-m** method – media, indicates the request is delivered via the specified method.

  The current support option is:

  – **email** or **smtp** If not specified, it will refer to transmitMedia option.
- **-t** targetPath – target location where the file(s) is(are) going to be stored. This is based on the value of **-p** or **-g** option. If this is not specified, then the value of ClientDft.targetPath is used.
- **-d** dest – destination, this can be an email address or http address depending on the **-m** option.

9.3.2 Command-line mode

The following shows the format of a command line event command:

actim -E -e "command line" [-m method] [-t target_path] -d dest
OR
actim -E -i commandID [-m method] [-t target_path] -d dest

A Return codes

0	Everything runs successfully.
1	Invalid arguments.
2	An option supposed to have integer value but an invalid type is found.
3	A required option with empty value
4	Python version does not meet requirements.
5	Server not started
6	actimrc or actimdrc files are not owner readable only
7	Run actimd as root
8	Command-line execution is not allowed
9	Non existing file in send event
10	Try to send a file without any permission from allowFileSend
11	Try to retrieve a file without any permission from allowFileReceive
12	Error in retrieving file
13	Restricted execution is on but no rcmdl_.... option found
14	Invalid re syntax in rcmdl_.... option
15	Command line expression does not qualify
16	Unrecognised server status command
17	Usage error on server status command
19	Server command error