A simple common message-standard.

The MANUS Team,
R.Akkersdijk
akkersdi@hio.hen.nl
HIO Enschede
the Netherlands

1 Introduction.

In a network of cooperative hosts, it is possible that messages are accidentally delivered to the wrong process because of some user inspired mistake. The purpose of this proposal is to defined a common message-standard that allows for the gracefull handling of such “lost” messages.

1.1 General.

Messages are ascii strings, terminated by a <CR><LF>. The use of an explicit terminator makes it easier to pass messages through pipes, to store them in files for replay or to duplicate them to the console for debugging.

The first part of a message is either a 4 character word, implying a request, or a 3 digit code (plus a flag), indicating the status of a reply.

The minimal message size therefore is 6 bytes (4 characters plus a <CR><LF>). No extra terminating ‘\0’ is included nor expected.

1.2 Requests.

Requests begin with a 4 letter word indicating the nature of the request, followed by optional data.

```
"<type>[data]\r\n"
```

1.3 Replies.

Replies begin with a 3 digit code, a one character flag, followed by optional data.

```
"<xyz><flag>[data]\r\n"
```

The 3 digit `xyz` code specifies the status of the reply (see below). Normally `flag` is a space but when more than one reply message is needed, all, but the last one, will have a hyphen (’`-`’) as the flag.

The status code is encoded as follows.

Code	Meaning
1yz	Positive Preliminary Reply; Action being initiated; Another, final, reply will come later.
2yz	Positive Completion Reply; Command completed.
3yz	Positive Intermediate Reply; Command accepted but delayed, please supply more information.
4yz	Transient Negative Completion Reply; Command is not accepted, no further action will be undertaken, but things may change so try again later.

Code	Meaning
5yz	Permanent Negative Completion Reply; Command not accepted.

The second digit gives a further clarification of the first.

Code	Meaning
x0z	About syntax-errors, e.g. unimplemented commands.
x1z	About information.
x2z	About connections.
x3z	About authentication & accounting.
x4z	About resources.
x5z	Application related things.

The last digit gives a further specification of the previous part of the reply code.

Reply codes should be chosen such that they alone already provide enough information for the application program to decide what action to undertake at a particular moment. The rest of the message will then be available for data or can be used as human-readable text for debugging purposes.

1.4 Examples.

Replies that may be common to several servers:

```
"150 Ok; Job has been spooled.\r\n"
"220 Command completed.\r\n"
"310 Ok; Give First Item.\r\n"
"311 Ok; Give Second Item.\r\n"
"440 Sorry, too many clients.\r\n"
"500 Command unrecognized.\r\n"
"530 Permission denied.\r\n"
"550 Illegal message size.\r\n"
```

As a specific example consider a name server process which binds names to (network)addresses:

```
request "BIND <name> to <addr>\r\n"
reply "210 Bound <name> to <addr>\r\n"
reply "440 No free slots in name-binding table.\r\n"
reply "500 Syntax is: BIND <name> to <addr><CR><LF>\r\n"
reply "510 Name <name> already bound.\r\n"
reply "530 Only allowed to BIND from same host.\r\n"
```

of course it must be possible to remove names (unbind):

```
request "UBND <name>\r\n"
reply "210 Unbound <name> from <addr>\r\n"
reply "500 Syntax is: UBND <name><CR><LF>\r\n"
reply "510 Name <name> is not bound.\r\n"
reply "530 Only allowed to UBND from same host.\r\n"
```

and to obtain a list of name to address mappings:

```
request "NLST\r\n"
reply "210 <name> -> <addr>\r\n"
reply "550 No names bound.\r\n"
```

Note that the 210 reply is a typical case where a continuation flag would be used to send multiple replies.

Another server could write messages to a terminal line:

```
request "WTTY <line> <text>\r\n".
reply "220 Ok.\r\n"
reply "430 write <line>; Operation would block\r\n"
reply "500 Syntax is: WTTY <line><SPACE><text><CR><LF>\r\n"
reply "510 <line> is not a valid terminal line.\r\n"
reply "530 Open /etc/utmp failed; <mesg>\r\n"
reply "531 <line> is not being used.\r\n"
reply "532 fork process failure; <mesg>\r\n"
reply "533 open <line> failure; <mesg>\r\n"
reply "534 fcntl <line> failure; <mesg>\r\n"
reply "535 write <line> failure; <mesg>\r\n"
```

1.5 Finally

The proposal does not prohibited the occurence of more than one <CR><LF> in a message. However messages must be such that the text upto the **first** <CR><LF> enables the receiver to decide about appropriate actions[1].

For instance if a server receives a 'BIND xxxx ....' request while it only handles 'RWHO zzz', it can consume the message and reply with a general "500 Sorry I don't grok this.<CR><LF>" error reply.

Similarly, for the sake of clients receiving unrecognizable replies, it might be prudent to have a common "ABRT<CR><LF>" dummy-request to abort a connection. Obviously they too might use the universal "500 ...<CR><LF>" message code.

1. In fact, after the first <CR><LF> it might be possible to allow non-ascii data to be embedded.