The notificationd protocol

`notificationd` is a protocol and implementation for relaying notification for display on multiple devices.

1 Introduction

The key words *"MUST"*, *"MUST NOT"*, *"REQUIRED"*, *"SHALL"*, *"SHALL NOT"*, *"SHOULD"*, *"SHOULD NOT"*, *"RECOMMENDED"*, *"NOT RECOMMENDED"*, *"MAY"*, and *"OPTIONAL"* in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2 Protocol overview

2.1 Messages

`notificationd` uses a plain-text line-based protocol for client/server communication.

The following pseudo-bnf provides an overview of the protocol.

```
message ::= [ <id> <space> ] [ <sign> ] <command> <arguments> [ [ <space> ] ':'
<trailing> ] <crlf>
id ::= <digit> { <digit> }
sign ::= '+' | '-' | '$'
command ::= <alpha> { <alpha> }
arguments ::= { <space> <argument> }
argument ::= <alpha> { <alpha> }
trailing ::= <any character except CR or LF>
space ::= ' ' { ' ' }
crlf ::= CR LF
```

It is *RECOMMENDED* for servers to accept commands as case-insensitive.

All messages with a sign are server messages. A message with sign plus (`+`) is a *succes reply*, and a message with sign minus (`-`) is a *failure reply*. In response to a command, a server may sent 0 or more *success replies* or a single *failure reply* but no combination of both. The dollar (`$`) sign indicates a command from the server.

A *failure reply MUST* have its first argument set to a reply error.

An empty *trailing text* followed by a colon (`':'`) *SHOULD* be accepted as an empty-string.

3 Notifications

4 Protocol message commands

4.1 Authentication

The following commands relate to the authentication protocol.

4.1.1 LOGIN

```
LOGIN <user> [password]
```

Login as `user`. When no `password` is supplied, access may be granted only if the server is configured to accept passwordless logins.

4.2 Notification details

Notifications use much of the basic design of org,freedesktop.Notifications.

Sending a message is a stateful process where the different *notification details* commands configure the current message being sent. When the *notification details* are finalized, they can be sent with SEND.

To reset the currently configured *message details*, the command RESET may be used.

The following details commands typically do not cause a succes reply.

4.2.1 TITLE

`TITLE : *`

A title or summary of the notification.

Similar to the "Summary" in Freedesktop notifications. `TITLE` was used a opposed to `SUMMARY` because it is shorter.

4.2.2 BODY

`BODY [RST] : *`

The body of the notification. The trailing text is appended to the current body. Using the `RST` argument causes the body to be reset. If trailing text is used in combination with `RST`, then the trailing text is set as the current body (disregarding previous lines).

4.2.3 ICON

`ICON : *`

Add a bitmap for the icon. The specific specs of the image will be documented later.

4.2.4 QUIET

`QUIET <bool>`

If the notification should be displayed or simply stored.

4.2.5 EPHERMAL

`EPHERMAL <bool>`

If the notification should be displayed then forgotten.

4.3 Notification transactions

4.3.1 SEND

`SEND`

Send the configured notification to the server. `SEND` *SHOULD NOT* cause the current message details configuration to be reset. Subsequently, repeated *SHOULD* cause the last message to be resent.

4.3.2 RESET

`RESET`

Reset the current notification configuration.

4.3.3 NOTIFY_START

`NOTIFY_START <user> <notification_id>`

Sent by the server to inform that client that it will now list the details of a notification from with id `notification_id`.

The trailing text may be used to provide a timestamp.

4.3.4 NOTIFY_END

`NOTIFY_END <notification_id>`

Sent by the server to inform the client that it will stop sending details of the notification with id `notification_id`.

4.3.5 CONSUME

`CONSUME [bool]`

Ask the server to relay notifications over this connection (consuming them for this user). If no `bool` is supplied `true` is assumed.

4.4 Database

The following commands may be used if notificationd is configured to be persistent.

4.4.1 HISTORY

`HISTORY [limit]`

Request the last `limit` notifications from the database. If no `limit` is specified, the complete history is returned.

4.4.2 SINCE

`SINCE <offset>`

Request all notifications with an ID higher than `offset`. This may be used by clients to track missed notifications.

4.5 Miscellaneous

4.5.1 VERSION

`VERSION`

4.5.2 WHO

`WHO`

List connected peers.

4.5.3 DELETE

`DELETE <id>`

Delete notification with id `id` from the database.

4.5.4 QUIT

`QUIT`

Sent by the client to signal termination of the socket. No further messages should be sent.

The server replies with the current version of the daemon.

4.5.5 NOTICE

`NOTICE : *`

Sent by the server to relay some out-of-spec information to the client. The client may just ignore these messages, display them to the user or log them.

5 Protocol reply errors

The following error codes may be used as the first argument in *failure replies*.

`PARSE`, `MISSING_TRAILING`, `NO_DB`, `DB_FAIL`, `INVALID_ARG`, `INVALID_MESSAGE`, `MISSING_ARG`