`@TellBot` usage

2018-04-13

`@TellBot` has reached all of its intended functionality, and development is restricted to maintenance. The command set may change upon popular demand or developer decision.

Commands

Aside from the mandatory commands from the "botrulez", `@TellBot` implements the following concrete commands:

- `!inbox` — Check own mailbox.
- `!tell` a.k.a. `!tnotify` — Send messages.
- `!reply` / `!reply-all` — Reply to a message.
- `!tgroup` / `!tungroup` — Manage groups of users.
- `!tgrouplist` — List the members of a group.
- `!tlistgroups` — List groups (filtering by name).
- `!tgroupsof` — List groups a user is in.
- `!alias` — Manage aliases of a user.
- `!seen` — Report when a user was last seen.

!inbox

```
!inbox [--stale]
```

Deliver the pending notifies to yourself. If `--stale` is passed, also post messages again that were delivered to you (this can be tried as a last effort of error recovery).

To immediately comment on the messages, you can also (after a `--` separator) post comments in the same message as the command; this is discouraged and may change in the future.

If a user has been away for a long time (*i.e.* more than two days), `@TellBot` does not deliver notifies immediately upon any message of the recipient (as some other bots do), but instead shows a notification, which advises to use this command.

Examples

```
!inbox
  No mail.

!inbox
  [You, 2d 5m ago] message

!inbox --stale
  [You, 2d 5m 13s ago] message
```

!tell or !tnotify

```
!tell [--ping] [--priority=<level>] <user-list> [--] <message>
!tnotify [--ping] [--priority=<level>] <user-list> [--] <message>
```

Schedule `message` to be delivered to all users in the given `user-list`. A `--` separator may be used to separate the recipients from the message body (such as when the message starts with an @-mention); it is not included in the delivered message. (To send a message starting with a double dash, duplicate the separator.)

`@TellBot` displays a breakdown of the users the message will actually be delivered to as a reply to the command. Unless `--ping` is passed, users on the list are not @-mentioned to allow sending messages of low importance without disturbing the recipients.

Each message has an associated priority that determines whether and with how much effort the recipient will be notified of it (*i.e.*, currently, whether an email notification will be sent if the user has opted in). Matching is case-insensitive. The priority levels are:

- `LOW`: A notification will never be sent.
- `NORMAL`: A notification will be sent if the user had been away for some time (the default is a week) and has not received another notification in some time (the default is a week as well).
- `URGENT`: A notification will be dispatched immediately and unconditionally. To prevent abuse, this level is only available to room hosts (and/or site staff).

If the message is empty (or consists of only whitespace), it is dropped (and the potential recipients are not notified), but the list of recipients is still displayed; this may be useful to observe the effects of different user list operations (see the corresponding section). Leading or trailing whitespace is stripped from the message; whitespace "inside" the message is unaltered.

If you submit a message to a group you are a member of and do not explicitly include yourself as a recipient, you will be dropped from the recipient list; this aids messaging groups without redundantly showing you the text you presumably already read while composing it. Explicit mention of yourself is interpreted as the explicit intent of delivering the message to yourself.

`!tell` and `!tnotify` are exactly equivalent; the latter is provided for closeness to the corresponding `@NotBot` command.

Examples

```
!tell @person1 something

!tnotify @person2 @person3 something else

!tell *group -- @somebot stopped working, can you check?
```

!notify

```
!notify [--ping] [--priority=<level>] <user-list> [--] <message>
```

Depending on the configuration, `@TellBot` may respond to a `!notify` command in the same way as to a `!tell`, either immediately, or after a delay (and unless something else does), or not at all. This (if enabled) renders `@TellBot` a failsafe for a `@NotBot` instance running in the same room, in case the latter goes down.

!reply and !reply-all

```
!reply <message>
!reply-all <message>
```

If (and only if) used as direct replies to delivered messages that are not older than some implementation-defined time (*i.e.* two days), these commands will send a message back to the sender of the received message (`!reply`) or the group the message was sent to (`!reply-all`). If the message was not sent to a group, `!reply-all` behaves like `!reply`.

The message starts immediately after the command (and can in particular start with any character). Implicit self exclusion happens in the case of `!reply-all`, and does not happen for `!reply`.

Examples

```
[You, 5s ago] some message
  !reply some reply

[person to *group, 5s ago] another message
  !reply Can you clarify that a bit?

[person to *group, 5s ago] yet another message
  !reply-all another reply
```

!tgroup and !tungroup

```
!tgroup [--ping] *<group> [<user-list>] [-- <description>]
!tungroup [--ping] *<group> <user-list>
```

`!tgroup` updates the given `group` with the result of building `user-list` basing upon it. If `user-list` is empty, the members of the group are displayed without mutating it.

`!tungroup` removes all members of `user-list` (starting with an empty set) from the group. Note that the `user-list` must be nonempty; otherwise, the command has no effect.

Unless `--ping` is passed, user names are not @-mentioned to avoid unnecessary alerting.

If `description` is passed, subsequent queries of that group will display it until it is replaced.

Examples

```
!tgroup *group @person1 @person2
  Group: *group
  Members before: -none-
  Members after (2): person1, person2

!tgroup --ping *group
  Group: *group
  Members (2): @person1, @person2

!tgroup *group -*group -- A demonstration group.
  Group: *group
  New description: A demonstration group.
  Members before (2): person1, person2
  Members after: -none-

!tgroup *group
  Group: *group
  Description: A demonstration group.
  Members: -none-
```

!tgrouplist

```
!tgrouplist [--ping] *<group>
```

`!tgrouplist` is an alias for `!tgroup` that only allows listing a group. It is provided for closeness to the corresponding `@NotBot` command.

!tlistgroups

```
!tlistgroups [pattern]
```

Enumerate all groups (or those whose name without the `*` sigil match a globbing `pattern`) known to `@TellBot`. The output is alphabetically sorted.

Pattern syntax

`pattern` may not contain whitespace, and can include the following metacharacters:

- `?` matches an arbitrary single character.

  To match a literal question mark, enclose it in a character class (*i.e.* use `[?]`; see below).

- `*` matches an arbitrary amount of arbitrary characters (including none).

  To match a literal asterisk, enclose it — similarly to the question mark — in a character class (*i.e.* use `[*]`).

- `[` initiates a *character class* that matches one character of any such mentioned in it; it is closed by a `]`, and the `[` may be immediately followed by `!` denoting a negative match; character ranges (separated by dashes `-`) are allowed.

  To include a closing bracket (`]`), place it immediately after the opening bracket (`[`) or the negation sign (`!`); to include a dash, place it at the beginning (similarly to `]`) or end (immediately before the closing `]`) of the character class. Nested opening brackets are not treated specially.

  Thus, `[][]` matches a closing or an opening bracket; `[!]]` matches any single character but a closing bracket; `[?-]` matches a question mark or a hyphen; `[a-z]` matches a letter of (see below) any case.

The pattern must match the entire group name (ignoring case); to "de-anchor" it from an end, use leading or trailing asterisks `*`.

Examples

```
!tlistgroups
  *anyquestions?
  *botdevs
  *groupA, *groupB, *groupC, *groupD
  *test, *testing

!tlistgroups group?
  *groupA, *groupB, *groupC, *groupD

!tlistgroups *st*
  *anyquestions?
  *test, *testing

!tlistgroups [gt]*
  *groupA, *groupB, *groupC, *groupD
  *test, *testing

!tlistgroups *[c-g]
```

```
  *groupC, *groupD
  *testing

!tlistgroups *[?]
  *anyquestions?
```

!tgroupsof

```
!tgroupsof [--ping] <user-list>
```

Reply with the groups the users in `user-list` are members of.

Unless `--ping` is specified, users are not pinged.

Examples

```
!tgroupsof @user1
  Groups of user1: -none-

!tgroupsof --ping @user2
  Groups of @user2 (1): *group1

!tgroupsof *group1
  Groups of user2 (1): *group1
  Groups of user3 (3): *group1, *group2, and *group3
```

!alias and !unalias

```
!alias [--ping] @<user> [<user-list>]
!unalias [--ping] @<user> <user-list>
```

Alias together multiple users (along with all their former aliases) and/or remove some aliases of a user.

`!alias` builds the given `user-list` — which may not contain groups — starting from the current set of aliases of `user`, extends the list by all other aliases of users contained in the list (*except* `user`), and installs the result as a new alias list instead of the former one of `user` (and any added members).

`!unalias` builds the `user-list` — which again may contain no groups — starting from the empty list and removes the entries of `user-list` from the alias list of `user`; it cannot add new aliases.

Unless `--ping` is passed, user names are not @-mentioned to avoid unnecessary alerting.

For consistency's sake, a user is always considered to have a one-entry alias set if no other aliases are present (or the observable behavior should be equivalent to that).

Aliases interact with the other features of `@TellBot` primarily at the time of retrieval. A group some members whereof have been aliased together retains all members *per se* (although only one is displayed); when it is modified, however, only one of the aliases is written back. Messages are stored with the recipient originally indicated, and are delivered to which alias ever of that user appears first. Information about when a user has been seen is stored per original name, and the results for all aliases of a user are aggregated when the information is actually requested.

Hence, creating and removing an alias again would not have too many grave consequences if other information concerning that user is not accessed in the meantime.

Examples

```
!alias @user1 @user2 @user3
  Aliases of @user1 before: user1
  Aliases of @user1 after: user1, user2, and user3

!alias --ping @userA @userB @userX
  Aliases of @userA before: @userA
  Aliases of @userA after: @userA, @userB, and @userX

!unalias @userX @userX
  Aliases of @userX before: userA, userB, and userX
  Aliases of @userA after: userA and userB

!alias @user1 @userA
  Aliases of @user1 before: user1, user2, and user3
  Aliases of @user1 after: user1, user2, user3, userA, and userB

!alias --ping @test
  Aliases of @test: @test
```

!seen

```
!seen <user-list>
```

Reply with time intervals since the users in `user-list` were last seen (*i.e.* posted something in a room observed by `@TellBot` whilst it was running), the rooms they were in, and counts of messages pending delivery to them.

Examples

```
!seen @person1
  @person1 last seen here on {some date}, 5m 2s ago.

!seen @person2
```

```
  @person2 not seen (3 pending messages).

!seen *group
  @person3 last seen here on {some date}, 1d 4h 5s ago (1 pending message).
  @person4 last seen in &test on {some date}, 41d 23h 59m ago.
```

User lists

`@TellBot` uses a moderately powerful array of incremental set operations to allow specifying sets of users.

A **user list** (actually an ordered set) is built starting with a *base* (that is empty where not explicitly mentioned) and changing it in accord with certain operations in the order the latters are given. These operations are:

- `+@<nick>`: Add the specified user to the user list (if not already present).
- `+*<group>`: Add all members to the user list (for each, if not already present).
- `-@<nick>`: Remove the specified user from the list (if present).
- `-*<group>`: Remove all members of the group from the user list (for each, if present).
- `@nick`: A "bare" nickname is equivalent to adding the user.
- `*group`: A "bare" group name is equivalent to adding the group.

Hence, `+` is equivalent to the set union operator, and `-` to the set difference operator; applied to the unitary set containing the specified user in one case, or to the set of the members of a group in the other. Beyond the set semantics, the operators attempt to maintain the relative order of users; the addition operators append "new" users to the end of the list. Therefore, removing a user and re-adding them will result in pushing them to the end.

The order of users does not have any effect *per se*, but is preserved upon display; it also affects the group listing shortening in the bot's response to `!tell` (*i.e.* the omitting of users already mentioned elsewhere in the response).

Note that the operations are not commutative: `-@user +@user` will have a different effect from both `+@user -@user` and discarding both operations (respectively, the user will shifted to the end of the list, the user will be removed, the user will not be affected at all).

Examples

For basic examples, see the respective commands.

- `!tell @person1 @person2 @person1 message` — Deliver a message to @person1 and @person2 (*i.e.*, the message will *not* reach @person1 twice).
- `!tell *programmers -*botdevs message` — Deliver a message to all programmers except the bot developers.
- `!tgroup *programmers` — Do not alter *programmers (the special case mentioned in the documentation of `!tgroup` is equivalent to what would happen if it were not there).

- `!tgroup *programmers *botdevs` — Add all bot developers to *programmers.
- `!tgroup *programmers -*programmers` — Remove all members from *programmers (*i.e.* clear the group).