1 Gateway server internal protocol

This describes the internal protocol for the Gateway server, which is written in Common Lisp.

1.1 Sexpable

An object is sexpable if:

- it provides a method for the **SEXP** generic function;
- it should ever be output from/to the server so that its representation is reconstructible in the client.

1.1.1 SEXP *Generic Function*

`object → sexp`

This function takes a sexpable object and produces its representation as an S-expression.

The resulting S-expression may contain only lists, symbols, numbers and strings.

This representation must not refer to any system-specific details of the object, such as its temporary memory location on the machine and/or any IDs internal to the runtime and not the persistent data store.

1.2 Password

A class fulfills the password protocol if:

- it provides a method for the **MAKE-PASSWORD** and **PASSWORD-MATCHES-P** generic functions;
- it is the **PASSWORD** class itself or a subclass of it.

1.2.1 PASSWORD *Class*

The PASSWORD class provides a default implementation of the password protocol. It may be overridden by a subclass's own behaviour that suits the subclass's own implementation.

1.2.2 MAKE-PASSWORD *Generic Function*

`passphrase → password`

This creates a newly allocated instance of the **PASSWORD** class, for which **(password-matches-p instance passphrase)** returns **T**.

1.2.3 PASSWORD-MATCHES-P *Generic Function*

`password passphrase → generalized-boolean`

Returns **T** if **passphrase** matches the **password**; otherwise, returns **NIL**.

1.3 Chatter

A class fulfills the chatter protocol if:

- it provides a method for the **SEND-MESSAGE** and **MSG** generic functions;
- it is a subclass of the **CHATTER** class.

1.3.1 CHATTER *Protocol Class*

1.3.2 SEND-MESSAGE *Generic Function*

`message recipient → nil`

This sends the provided message, which is a generalized instance of the **MESSAGE** class, to the provided recipient, which is a generalized instance of the **CHATTER** class.

For **MSG**, see the **Message** protocol.

1.4 Message

A class fulfills the message protocol if:

- it provides a method for the **SENDER**, **RECIPIENT**, **DATE-OF**, **CONTENTS** and **MSG** generic function;
- it is a subclass of the **CHATTER** class.

1.4.1 MESSAGE *Class*

The MESSAGE class provides a default implementation of the message protocol. It may be overridden by a subclass's own behaviour that suits the subclass's own implementation.

1.4.2 SENDER *Generic Function*

`message → chatter`

1.4.3 SETF SENDER *Generic Function*

Accesses the sender of the **message**. The sender must be a generalized instance of the **CHATTER** class.

1.4.4 RECIPIENT *Generic Function*

`message → chatter`

1.4.5 SETF RECIPIENT *Generic Function*

Accesses the recipient of the **message**. The recipient must be a generalized instance of the **CHATTER** class.

1.4.6 DATE-OF *Generic Function*

`message → chatter`

1.4.7 SETF DATE-OF *Generic Function*

Accesses the date of the **message**. The date must be a generalized instance of the **DATE** class.

1.4.8 CONTENTS *Generic Function*

`message → chatter`

1.4.9 SETF CONTENTS *Generic Function*

Accesses the contents of the **message**. The contents must be a **STRING**.

1.4.10 MSG *Generic Function*

`sender recipient contents → message`

This creates a newly allocated instance of the **MESSAGE** class, using the provided **sender** (a **CHATTER**), **recipient** (a **CHATTER**), **contents** (a **STRING**) and the current system date.

For **SEND-MESSAGE**, see the **Chatter** protocol.

1.5 Persona

A class fulfills the persona protocol if:

- it provides a method for the **NAME**, **PLAYER** and **FIND-PERSONA** generic function;
- it is a subclass of the **PERSONA** class.

1.5.1 PERSONA *Class*

The PERSONA class provides a default implementation of the persona protocol. It may be overridden by a subclass's own behaviour that suits the subclass's own implementation.

Each persona can be uniquely described by its **name**. It is an error to try to instantiate a persona whose name is **STRING=** to another persona already existing in the database.

1.5.2 NAME, (SETF NAME) *Generic Function*

`persona → name`

Accesses the name of the **persona**. The contents must be a **STRING**.

1.5.3 PLAYER, (SETF PLAYER) *Generic Function*

`persona → name`

Accesses the player of the **persona**. The contents must be a **PLAYER**.

1.5.4 FIND-PERSONA *Generic Function*

`name → persona`

Returns the **persona** whose name is **STRING=** to **name**.