

phshim User Manual

A CCSO Nameserver (Ph/Qi) Backed by an SQL Database

Version 1.0

Dave McGuire
Large Scale Systems Museum
mcguire@lssmuseum.org

September 2026

Abstract

`phshim` is a server for the CCSO Nameserver protocol, also known as Ph or Qi, described in RFC 2378. It answers Ph clients from a PostgreSQL or MySQL/MariaDB database, translating each command into parameterized SQL. Directory fields are defined in the configuration file, and `phshim` creates and maintains the database schema from those definitions. This manual describes how to build, install, configure, and operate `phshim`, and how it implements the Ph protocol.

Contents

1	Introduction	4
1.1	The CCSO Nameserver	4
1.2	What phshim does	4
1.3	About this manual	4
2	Architecture	6
2.1	Process model	6
2.2	Database backends	6
2.3	Schema	6
3	Installation	8
3.1	Requirements	8
3.2	Source layout	8
3.3	Building	8
3.4	Installing	9
3.5	Service management	9
3.5.1	systemd	9
3.5.2	SMF (Solaris and illumos)	10
3.5.3	Port 105 without root	10
4	Database Setup	11
4.1	Creating the schema	11
4.2	Privileges	11
4.3	PostgreSQL	12
4.4	MySQL and MariaDB	12
4.5	Creating the first Hero	12
4.6	Adding fields later	13
5	Configuration	14
5.1	File format	14

5.2	[server]	14
5.3	[site]	15
5.4	[database]	15
5.5	[directory]	16
5.6	[local]	16
5.7	[field]	16
6	Directory Fields	17
6.1	Defining fields	17
6.2	Where field definitions come from	17
6.3	Attributes	17
6.4	Visibility	18
6.5	Multi-line values	18
7	Running the Server	19
7.1	Command line	19
7.2	Startup	19
7.3	Logging	20
7.4	Signals	20
7.5	Exit status	20
8	The Ph Protocol	21
8.1	Sessions	21
8.2	Response format	21
8.3	Commands	22
8.3.1	status	22
8.3.2	siteinfo	22
8.3.3	fields [<i>name</i> . . .]	22
8.3.4	id <i>text</i>	22
8.3.5	set [<i>option=value</i> . . .]	22
8.3.6	login, answer, clear, logout	23
8.3.7	email, xlogin	23
8.3.8	query	23
8.3.9	add	23
8.3.10	delete	24
8.3.11	change	24
8.3.12	help	24
8.3.13	quit, exit, stop	24

8.4 Wildcards and matching	24
9 Authentication and Access Control	26
9.1 Logging in	26
9.2 Heroes	26
9.3 Owners	27
9.4 Local and external clients	27
10 Security	28
10.1 Passwords	28
10.2 Transport	28
10.3 SQL	28
10.4 Privileges	28
10.5 Sensitive fields	28
11 Troubleshooting	29
11.1 Startup errors	29
11.2 Client errors	29
11.3 Testing a server	30
A Response Codes	31
B Limits	33
C Sample Configuration Fields	34
D License	35

Chapter 1

Introduction

1.1 The CCSO Nameserver

The CCSO Nameserver was developed at the University of Illinois Computing and Communications Services Office in the late 1980s, and became the standard campus directory service at many universities before LDAP. Its server was called *qi* and its client *ph*, and the protocol is usually called Ph. RFC 2378 documents it.

A Ph directory is a collection of *entries*, each made up of named *fields* such as *name*, *email*, and *phone*. Clients search the directory with simple text commands over TCP port 105, and entry owners and administrators can change entries after logging in.

1.2 What phshim does

phshim implements the server side of the Ph protocol on top of an SQL database instead of qi's own database files:

- Each Ph command is translated into parameterized SQL and run against PostgreSQL or MySQL/MariaDB.
- Directory fields, and their attributes controlling search, visibility, and change rights, are defined in the configuration file.
- The schema is created and extended from those definitions with one command (`--init-db`), so no hand-written SQL is needed.
- Ph wildcard searches are translated to the database's pattern matching.
- Logins, administrator (Hero) privileges, owner-only changes, and local-network-only fields are supported.

1.3 About this manual

Chapter 3 covers building and installing phshim, and Chapter 4 setting up its database. Chapters 5 and 6 describe the configuration file and directory fields. Chapter 7 covers running the server,

Chapter 8 the protocol as `phsim` implements it, and Chapter 9 authentication and access control. Chapter 10 discusses security, and Chapter 11 troubleshooting. The appendices list response codes, limits, and the fields defined in the sample configuration.

The manual page, `phsim(8)`, is a concise reference to the same material.

Chapter 2

Architecture

2.1 Process model

phshim is a forking server. The parent process opens the listening socket and accepts connections; for each connection it forks a child process, which serves that client until it disconnects or sends quit. The parent does no database work of its own after startup.

Each child opens its own database connection, loads the field definitions (from the configuration file or the database), and then reads commands one line at a time. Each command is parsed, checked against the session's authentication state and the field attributes, translated into SQL, and answered in the Ph wire format.

The parent limits the number of simultaneous children to `max_connections`. When the limit is reached, it stops accepting connections until a child exits; new connections wait in the listen queue meanwhile.

2.2 Database backends

phshim supports two database backends:

PostgreSQL through libpq.

MySQL and MariaDB through libmysqlclient or libmariadb.

Either backend, or both, can be compiled in. When both are present, the backend setting in the configuration file chooses one at startup. All database access goes through a small internal interface, so the rest of the server is the same for both.

2.3 Schema

phshim uses two tables, whose names are configurable:

The entry table (`ph_entries`) holds one row per directory entry, with one column per field.

The field table (`ph_fields`) holds one row per field, describing its protocol number, name, length, attributes, and description.

Both are created by `--init-db`; see Chapter 4.

Chapter 3

Installation

3.1 Requirements

- A POSIX system. `phshim` is developed on Linux and supports Solaris and illumos (including SmartOS), macOS, and the BSDs.
- GNU `make` and a C11 compiler, such as GCC or Clang.
- The client development files for the database: `libpq` for PostgreSQL, or `libmysqlclient` or `libmariadb` for MySQL/MariaDB.
- A C library with `getopt_long`: `glibc`, the BSD and macOS libraries, illumos, or Solaris 11. Solaris 10 lacks it.

On Debian and Ubuntu the database libraries are in the `libpq-dev` and `libmysqlclient-dev` (or `libmariadb-dev`) packages; on RHEL and Fedora, `libpq-devel` and `mysql-devel` (or `mariadb-devel`).

3.2 Source layout

Path	Contents
<code>Makefile</code>	Build, install, and distribution rules
<code>src/</code>	C sources and headers
<code>doc/</code>	This manual, the manual page, <code>USAGE.md</code> , and the sample configuration, <code>phshim.conf.sample</code>
<code>Linux/phshim.service</code>	systemd unit
<code>Solaris/phshim.xml</code>	SMF manifest
<code>test/test_phshim.py</code>	Protocol test suite
<code>build/</code>	Objects and the binary (created by the build)

3.3 Building

The `BACKEND` variable selects the database support:

```
make                # PostgreSQL (the default)
make BACKEND=mysql  # MySQL/MariaDB
make BACKEND=both   # both; chosen at runtime by "backend"
```

The binary is `build/phshim`. The Makefile finds the database headers and libraries with `pg_config` and `mysql_config` (or `mariadb_config`) where available, and otherwise searches the usual installation directories. `CC`, `CFLAGS`, and `LDFLAGS` may be set as usual.

Objects are not rebuilt automatically when `BACKEND` changes, so run `make clean` before building with a different backend.

3.4 Installing

```
make install
make install PREFIX=/usr
```

This installs:

- the binary as `$(PREFIX)/bin/phshim` (`PREFIX` defaults to `/usr/local`);
- the manual page as `$(PREFIX)/share/man/man8/phshim.8`;
- the sample configuration as `/etc/phshim/phshim.conf`, only if that file does not already exist.

`DESTDIR` is honored for staged installs.

This manual is distributed as `doc/phshim-manual.pdf`. With a LaTeX installation, `make manual` regenerates it from `doc/phshim-manual.tex`; set `PDFLATEX` to use a command other than `pdflatex`.

3.5 Service management

3.5.1 systemd

`Linux/phshim.service` runs `phshim` as a forking service with a PID file:

```
cp Linux/phshim.service /etc/systemd/system/
systemctl daemon-reload
systemctl enable --now phshim
```

The unit runs `phshim` as root so that it can bind port 105. To run it as a dedicated user instead, see Section 3.5.3. `phshim` does not reload its configuration on `SIGHUP`, so the unit has no reload action; use `systemctl restart phshim` after changing the configuration.

3.5.2 SMF (Solaris and illumos)

Solaris/phshim.xml is an SMF manifest. It expects the binary at /opt/local/sbin/phshim and the configuration at /opt/local/etc/phshim.conf; edit it if your paths differ.

```
svccfg import phshim.xml
svcadm enable phshim
```

3.5.3 Port 105 without root

Port 105 is a privileged port. To avoid running phshim as root:

- On Linux, grant the capability to the binary:

```
setcap cap_net_bind_service=+ep /usr/local/bin/phshim
```

Under systemd, you can instead set AmbientCapabilities=CAP_NET_BIND_SERVICE and a User= in the unit.

- Or listen on an unprivileged port (port = 1055) and redirect port 105 to it with the packet filter, for example:

```
nft add rule ip nat PREROUTING tcp dport 105 redirect to 1055
```

Chapter 4

Database Setup

4.1 Creating the schema

phshim creates its own schema from the [field] stanzas in the configuration file:

```
phshim -c /etc/phshim/phshim.conf --init-db
```

This performs five steps, reporting progress on standard output:

1. Creates the field table if it does not exist.
2. Creates the entry table if it does not exist, with the structural columns entry_id, password, created_at, and updated_at.
3. Adds a column to the entry table for each [field] stanza that does not already have one. A field whose max_length is 4096 or more becomes a TEXT column; others become VARCHAR(max_length).
4. Inserts or updates a field-table row for each stanza.
5. Creates an index for each Indexed field: on lower(column) with PostgreSQL, and on the first 64 characters of the column with MySQL. Indexes are named idx_table_field.

--init-db is idempotent and additive. It is safe to run repeatedly and against an existing database, and it never drops or narrows a column. To remove a field, delete its column by hand.

The alias and acl fields must be defined: logins look up the alias column, and Hero status comes from acl.

4.2 Privileges

Run --init-db with an administrative database account, and give the account phshim uses at runtime only the privileges it needs. Do not let the runtime account create the tables: the owner of a table can always alter or drop it. A convenient way is to keep a second copy of the configuration file whose [database] user is the administrative account.

The runtime account needs:

- SELECT, INSERT, UPDATE, and DELETE on the entry table;
- SELECT on the field table;
- with PostgreSQL, USAGE on the entry table's sequence, `ph_entries_entry_id_seq`.

If the directory is read-only (`readonly = yes`), SELECT on both tables is enough.

4.3 PostgreSQL

```
createdb directory
psql -d directory -c "CREATE USER phshim WITH PASSWORD 'changeme';"
phshim -c /etc/phshim/phshim-admin.conf --init-db
psql -d directory -c "GRANT SELECT, INSERT, UPDATE, DELETE
                     ON ph_entries TO phshim;"
psql -d directory -c "GRANT SELECT ON ph_fields TO phshim;"
psql -d directory -c "GRANT USAGE ON SEQUENCE ph_entries_entry_id_seq
                     TO phshim;"
```

4.4 MySQL and MariaDB

```
mysql -u root -e "CREATE DATABASE directory CHARACTER SET utf8mb4;"
mysql -u root -e "CREATE USER 'phshim'@'localhost'
                 IDENTIFIED BY 'changeme';"
phshim -c /etc/phshim/phshim-admin.conf --init-db
mysql -u root -e "GRANT SELECT, INSERT, UPDATE, DELETE
                 ON directory.ph_entries TO 'phshim'@'localhost';"
mysql -u root -e "GRANT SELECT ON directory.ph_fields
                 TO 'phshim'@'localhost';"
```

phshim uses standard double-quoted identifiers in its SQL, and enables the `ANSI_QUOTES` SQL mode on each of its MySQL connections so that the server accepts them. It also sets the connection character set to `utf8mb4`. No server configuration is required.

4.5 Creating the first Hero

Heroes are the directory's administrators. The `acl` field that grants Hero status is `Sacred`, so it cannot be set over the network. Create the first Hero with `--add-entry`, which writes directly to the database:

```
phshim -c /etc/phshim/phshim.conf --add-entry \
    alias=admin name="Admin User" password=secret acl=hero
```

`--add-entry` performs no authentication and does not check field attributes; its arguments are column names and values. At most 64 pairs are used.

4.6 Adding fields later

1. Add a `[field]` stanza with a new, unused `index`.
2. Run `--init-db` again. It adds the column and the field-table row, leaving existing data alone.
3. Restart `phshim`.

Chapter 5

Configuration

5.1 File format

The configuration file, `/etc/phshim/phshim.conf` by default, is divided into sections. A section begins with a line containing its name in square brackets, and contains lines of the form *key = value*.

- Lines beginning with `#` or `;` are comments. A `#` outside double quotes starts an inline comment.
- A value may be enclosed in double quotes.
- Keys and section names are case-sensitive.
- Boolean values are true if they are `yes`, `on`, or `1`, and false otherwise.
- String values are limited to 511 characters.
- Unknown keys are ignored. If a key appears more than once, the last value is used.

The configuration is read once at startup. To apply changes, restart the server. `phshim -D` prints the effective configuration.

The sample configuration, `doc/phshim.conf.sample`, documents every setting and defines the standard fields; start from it.

5.2 [server]

Key	Default	Meaning
<code>port</code>	105	TCP port to listen on.
<code>listen</code>	0.0.0.0	IPv4 address to bind; the default is all interfaces.
<code>max_connections</code>	64	Maximum number of simultaneous client connections.
<code>pid_file</code>	none	File in which to record the server's process ID.
<code>log_file</code>	none	File to append log messages to. If unset, messages go to syslog with the <code>LOG_DAEMON</code> facility.
<code>log_level</code>	1	Logging detail; see Section 7.3.

Key	Default	Meaning
motd_file	none	File whose lines are sent in response to status.

5.3 [site]

These values are reported to clients by the `siteinfo` command.

Key	Default	Meaning
version	phshim/ <i>version</i>	Server version string. The default follows the version of the running binary.
maildomain	none	Mail domain that clients combine with the mailfield value to form an address.
mailfield	alias	Field that forms the local part of an address.
mailbox	email	Field holding the delivery address.
administrator	none	Contact address for the directory administrator.
passwords_contact	none	Contact for password problems, reported as passwords.

5.4 [database]

Key	Default	Meaning
backend	pgsql	pgsql (or postgresql) for PostgreSQL, or mysql for MySQL/MariaDB. The backend must be compiled in.
host	see text	Database server host. With PostgreSQL, leave it unset to use the local socket; localhost connects over TCP, and a directory name gives the socket's directory. With MySQL, localhost (the default) uses the local socket.
port	0	Database server port; 0 uses the library default (5432 or 3306).
dbname	directory	Database name.
user	library default	Database user name.
password	none	Database password.
socket	none	Path of the MySQL local socket.
ssl	no	Require an encrypted database connection.

The configuration file holds the database password, so make it readable only by the account phshim runs as (`make install` installs it with mode 640).

5.5 [directory]

Key	Default	Meaning
table	ph_entries	Name of the entry table.
field_table	ph_fields	Name of the field table.
default_search_field	name	Field searched by query terms that give no field name.
default_limit	40	Initial per-session limit on the number of entries a query may return or a change or delete may affect.
max_limit	200	Largest value a client may set the limit to.
readonly	no	If true, refuse login, add, change, and delete.
use_db_fields	no	If true, load field definitions from the field table even when [field] stanzas are present.

Table names are restricted to letters, digits, and underscores.

5.6 [local]

Key	Default	Meaning
network	none	An IPv4 network in CIDR notation, such as 10.0.0.0/8, whose clients are local. Repeat the key for each network, up to 32.

Local clients may see fields with the LocalPub attribute; see Section 9.4.

5.7 [field]

Each [field] stanza defines one directory field. Chapter 6 describes fields in detail.

Key	Default	Meaning
index	required	The field's number in the Ph protocol. Must be unique.
name	required	The field name, which is also the column name: letters, digits, and underscores.
max_length	256	Maximum value length.
attributes	none	Space-separated attribute keywords.
description	none	Text returned by the fields command.

Chapter 6

Directory Fields

6.1 Defining fields

A field is defined by a `[field]` stanza:

```
[field]
index      = 2
name       = email
max_length = 128
attributes = Lookup Public Default Change
description = Primary email address
```

The `index` identifies the field in the protocol, and clients may cache it, so never change an index once it is assigned. Indexes need not be sequential. Up to 256 fields may be defined.

Appendix C lists the fields defined in the sample configuration, which include the standard fields of RFC 2378 Appendix A.

6.2 Where field definitions come from

By default each connection takes its field definitions from the `[field]` stanzas in the configuration file. If there are none, or `use_db_fields` is set, it loads them from the field table instead. The field table is kept in step with the stanzas by `--init-db`; `use_db_fields` is for sites that manage the field table by other means.

6.3 Attributes

Attribute	Effect
Always	Returned when a query has no return clause (the same as Default).
Change	The entry's owner may change the field.
Default	Returned when a query has no return clause.

Attribute	Effect
Encrypt	The value is never sent; queries return <encrypted> with code 522. Changing the field requires force instead of make.
Indexed	Every query must include at least one Indexed field. --init-db creates a database index for it.
LocalPub	Visible only to clients on a [local] network.
Lookup	The field may be used in search terms.
NoMeta	Wildcards are not allowed in searches on the field.
Private	Visible only to Heroes.
Public	Visible without logging in. Other fields are visible only to logged-in clients.
Sacred	Cannot be set by add or change, even by a Hero.
Turn	A value beginning with * is shown only to the entry's owner and to Heroes.
Unique	add refuses a value already used by another entry.
Any, ForcePub, NoPeople	Accepted for compatibility; no effect.

6.4 Visibility

Whether a client may see a field is decided as follows:

1. A Private field is visible only to Heroes.
2. A LocalPub field is hidden from sessions marked external.
3. A field without Public is visible only to logged-in clients.
4. Otherwise the field is visible.

Fields a client may not see are left out of query results and of the fields listing. In addition, Encrypt values are never sent, and Turn values beginning with * are shown only to the entry's owner and to Heroes.

6.5 Multi-line values

A value may contain line breaks. Each line of the value is sent as a separate response line; lines after the first have an empty field name (Section 8.2), as with the original qi server.

Chapter 7

Running the Server

7.1 Command line

```
phshim [-c config] [-d] [-D] [--init-db] [--add-entry field=value ...]
```

Option	Meaning
<code>-c, --config <i>file</i></code>	Read the configuration from <i>file</i> instead of <code>/etc/phshim/phshim.conf</code> .
<code>-d, --daemon</code>	Detach from the terminal and run in the background.
<code>-D, --dump-config</code>	Print the effective configuration to standard error and exit.
<code>--init-db</code>	Create or update the database schema and exit (Chapter 4).
<code>--add-entry <i>field=value...</i></code>	Insert one entry directly into the database and exit. All following arguments are <i>field=value</i> pairs, even ones that look like options, so it must come last.
<code>-h, --help</code>	Print the version and a usage summary on standard error and exit.

Long options may be abbreviated to any unique prefix, and short options may be combined (`-dD`).

7.2 Startup

When started without `--init-db` or `--add-entry`, `phshim`:

1. reads the configuration file;
2. connects to the database and checks that the entry and field tables exist, exiting with a message suggesting `--init-db` if either is missing;
3. binds and listens on the configured address and port;
4. if `-d` was given, detaches into the background;

5. writes the PID file, if one is configured;
6. accepts connections.

Because the schema check and the bind happen before detaching, their errors are reported on standard error even with `-d`.

7.3 Logging

Messages go to `log_file` if one is set, and otherwise to syslog with the `LOG_DAEMON` facility. `log_level` controls the detail:

Level	Messages
0	Errors only
1	Also startup, shutdown, and client connections and disconnections (the default)
2	Also every command received, and each child process started

7.4 Signals

SIGTERM, SIGINT Stop accepting connections, remove the PID file, and exit. Children already serving clients are not affected unless they receive the signal themselves, in which case they close their connections.

SIGHUP Not handled; it terminates the server. The configuration cannot be reloaded without a restart.

7.5 Exit status

`phshim` exits with status 0 on success and 1 on any error.

Chapter 8

The Ph Protocol

This chapter describes the Ph protocol as phshim implements it. RFC 2378 is the full specification.

8.1 Sessions

A client connects to TCP port 105 and sends commands, one per line, each ending in a line feed (an optional preceding carriage return is ignored). Empty lines are ignored. The server answers each command before reading the next, and closes the connection after quit.

A command line is at most 4095 bytes; longer lines are truncated. A command is a keyword followed by arguments separated by spaces. An argument may be enclosed in double quotes to include spaces; within quotes, `\n`, `\t`, `\`, and `\"` stand for a line feed, a tab, a quote, and a backslash. Keywords and field names are not case-sensitive; values keep their case.

8.2 Response format

Each response line has the form

```
code:text
```

where *code* is a three-digit result code (Appendix A). A multi-line response consists of lines with negative codes, which mean “more follows”, and a final line with a positive code. Data lines carry an entry number and a field name:

```
-200:1:name:John Smith  
-200:1:email:jsmith@example.com  
-200:1::second line of a multi-line value
```

A response line longer than about 4 KB is truncated. A value’s embedded line breaks are sent as separate lines; any carriage return or line feed remaining in other text is replaced by a space, so that no value can break the line structure of the protocol.

8.3 Commands

8.3.1 status

Sends each line of the message-of-the-day file as a 100 line, then 200 Database ready., or 201 if the directory is read-only.

8.3.2 siteinfo

Reports the [site] settings, one per line, as version, maildomain, mailfield, mailbox, administrator, and passwords, followed by authenticate.

8.3.3 fields [*name ...*]

Describes every field the client may see, or only the named ones. Each field produces two lines, numbered with the field's index: its maximum length and attributes, then its description.

```
-200:2:email:max 128 Change Default Lookup Public
-200:2:email:Primary email address
200:Ok.
```

If named fields are requested and none is visible, the reply is 507.

8.3.4 id *text*

Records a client identification string for the session.

8.3.5 set [*option=value ...*]

Sets session options, or lists them if none are given.

Option	Meaning
echo	When on, each command is echoed back on a 101 line.
limit	The session's entry limit, from 1 to max_limit; larger values are reduced to max_limit.
external	When on, the session is external: LocalPub fields are hidden and cannot be searched. See Section 9.4.
addonly	When on, change is refused with 521.
verbose, nolog, charset	Accepted and reported, with no other effect.

Boolean options are on if the value is on or 1. An unknown option is refused with 513.

8.3.6 login, answer, clear, logout

login *alias* begins a login and replies 301 with a challenge string; the client then sends answer *password* or clear *password*. logout ends the login. See Section 9.1.

8.3.7 email, xlogin

Not supported. Both reply 529 and cancel any login in progress.

8.3.8 query

```
query term ... [return field ... | return all]
```

ph is a synonym for query. Each *term* is either *field=value*, or a bare *value*, which searches default_search_field for entries *containing* the value. All terms must match.

- Every field in a term must exist (507) and have the Lookup attribute (504). A LocalPub field cannot be searched from an external session (504).
- At least one field in the terms must be Indexed (515).
- Wildcards (Section 8.4) are refused with 504 on NoMeta fields.
- At most 64 terms are allowed (599).

The return clause chooses the fields to return; return all returns every visible field, and without a clause the Default and Always fields are returned. Fields the client may not see are left out; if none remains, the reply is 501.

If nothing matches, the reply is 501. If more entries match than the session limit, the query fails with 502; narrow it or raise the limit. Otherwise the reply is a 102 line giving the number of matches, the data lines, and 200.

```
query name=smith return name email
102:There was 1 match to your request.
-200:1:name:John Smith
-200:1:email:jsmith@example.com
200:Ok.
```

8.3.9 add

```
add field=value ...
```

Creates an entry. Only Heroes may add entries (511), and not in a read-only directory (517). Every field must exist (507) and must not be Sacred (505). A value for a Unique field must not already be in use (509). At most 64 pairs are allowed (599).

8.3.10 delete

```
delete term ...
```

Deletes the entries matching the terms, which work as in query. Only Heroes may delete entries (510), and not in a read-only directory (517). If no entry matches, the reply is 501; if more match than the session limit, the command fails with 518. At most 64 terms are allowed. On success the reply gives the number of entries deleted.

8.3.11 change

```
change field=value ... make field=value ...  
change field=value ... force field=value ...
```

Sets the fields after make in every entry matching the selectors before it. Selectors must name their fields.

- The client must have completed a login (506), and the directory must not be read-only (517).
- A client who is not a Hero may change only fields with the Change attribute (505), and only in their own entry.
- No one may change a Sacred field (505).
- An Encrypt field may be changed only with force in place of make (505).
- In addonly mode, change is refused (521).
- If no entry matches, the reply is 501; if more match than the session limit, the command fails with 518.
- At most 63 terms are allowed.

8.3.12 help

Summarizes the commands.

8.3.13 quit, exit, stop

Replies 200 Bye! and closes the connection.

8.4 Wildcards and matching

Search values may contain these wildcards:

Wildcard	Matches
*	Any characters, including none
?	Exactly one character
+	One or more characters
[<i>set</i>]	One character from <i>set</i>

phshim translates each search term into SQL:

Value	PostgreSQL	MySQL/MariaDB
No wildcards	<code>lower(col) = lower(value)</code>	the same
Only * or ?	ILIKE	REGEXP
Contains + or [<i>set</i>]	SIMILAR TO	REGEXP

Matching ignores case, except that MySQL REGEXP matches follow the column's collation; the default `utf8mb4` collations are case-insensitive. Search values are always passed to the database as query parameters, never as SQL text.

Chapter 9

Authentication and Access Control

9.1 Logging in

A login takes two commands:

```
login jsmith
301:<challenge>
clear secret
200:jsmith>Welcome.
```

phshim looks up the entry whose alias matches the login alias, ignoring case, and compares the password given with the entry's password column. `answer` and `clear` behave the same way: phshim does not implement challenge-response encryption, and the challenge is not used.

- `answer` or `clear` without a preceding `login` replies 523.
- A wrong password or unknown alias replies 516 and ends the login attempt.
- In a read-only directory, `login` replies 403.

9.2 Heroes

If the entry's `acl` field contains the word `hero` (in any case), the client becomes a Hero on logging in. A Hero may:

- see `Private` fields and suppressed `Turn` values;
- add and delete entries;
- change any field that is not `Sacred`, in any entry.

Starting a new login, or trying `email` or `xlogin`, discards any Hero status the session had.

9.3 Owners

The *owner* of an entry is the client logged in with that entry's alias. An owner may change the entry's fields that have the `Change` attribute, and see its suppressed `Turn` values. Only a completed login counts; a client that has sent login but not yet a correct answer or clear owns nothing.

9.4 Local and external clients

A session is *external* unless the client's address is in one of the `[local]` networks. External sessions cannot see or search `LocalPub` fields.

A local client may set `external=on` to see the directory as an outsider would, and `external=off` to return. An external client cannot turn `external` off; the request is refused with 512.

Chapter 10

Security

10.1 Passwords

Passwords are stored in the entry table as given, and compared by simple equality. The Ph protocol sends them unencrypted. Protect the database accordingly: grant the runtime account only the privileges it needs, and restrict who can read the entry table.

To store hashed passwords instead, add a database function that performs the hash comparison, and use it in place of the equality test in `do_auth()` in `src/commands.c`.

10.2 Transport

The Ph protocol has no encryption. For any deployment reached over untrusted networks, put `phshim` behind a TLS proxy such as `stunnel`, and use a Ph client that can connect through it.

10.3 SQL

All values from clients are passed to the database as query parameters. Table and column names come from the configuration and are restricted to letters, digits, and underscores.

10.4 Privileges

Run `phshim` with the least privilege practical: as a dedicated user where possible (Section 3.5.3), with a database account limited to the privileges in Chapter 4. Keep the configuration file, which contains the database password, readable only by that user.

10.5 Sensitive fields

In the sample configuration, `acl` is `Private Sacred`, so it is neither visible to regular users nor changeable over the network, and `password` is `Encrypt Sacred`, so it is never sent. Keep these attributes if you edit those fields.

Chapter 11

Troubleshooting

11.1 Startup errors

schema check failed: required table '...' does not exist The tables have not been created. Run `--init-db`.

DB connect failed: ... The database is unreachable, or the [database] settings are wrong. The rest of the message comes from the database library.

unknown backend '...' backend is not one of pgsq1, postgresql, or mysql.

PostgreSQL backend not compiled in (or MySQL) The binary was built without that backend. Rebuild with the right BACKEND.

bind: Permission denied Port 105 needs privileges; see Section 3.5.3.

bind: Address already in use Another server is using the port.

config line N: ... A syntax error in the configuration file.

config: [field] stanza ... missing required 'index' key Every field needs an index (and a name).

config: duplicate field index N Two fields share an index.

11.2 Client errors

475 Database temporarily unavailable The server could not connect to the database for this client, or could not load the field definitions. Check the database and the log.

515 No indexed field in query Include at least one Indexed field in the search terms. A bare value searches `default_search_field`, which is Indexed in the sample configuration.

502 Too many entries More entries match than the session limit. Narrow the query, or raise the limit with `set limit=n`.

506 Must be logged in change requires a completed login.

505 Not authorized to change field The field lacks the Change attribute, or the entry is not the client's own.

11.3 Testing a server

`test/test_phshim.py` exercises the protocol against a running server. It needs a Hero account, creates entries whose aliases begin with `phtest-`, and removes them when it finishes.

```
python3 test/test_phshim.py --host localhost --port 105 \  
    --hero-alias admin --hero-pass secret
```

`--verbose` prints every command and response, and `--stop-on-fail` stops at the first failure.

A server can also be tested by hand with a TCP client:

```
$ nc localhost 105  
status  
200:Database ready.  
quit  
200:Bye!
```

Appendix A

Response Codes

These are the result codes phshim sends. Codes in the 100s are informational, 200s mean success, 300s ask for more input, 400s are temporary errors, and 500s permanent errors.

Code	Meaning
100	Message-of-the-day line (from status)
101	Echo of a command (set echo=on)
102	Number of entries matching a query
200	Success
201	Database ready, but read-only
301	Login challenge; send answer or clear
401	Database error
403	Login refused: the directory is read-only
475	Database unavailable
501	No matches, or no visible fields to return
502	Too many matches for the session limit
504	Field not searchable, not available externally, or wildcards not allowed
505	Not authorized to change the field
506	Login required
507	No such field
509	Value of a Unique field already in use
510	Not authorized to delete entries
511	Not authorized to add entries
512	Illegal value
513	Unknown set option
515	No Indexed field in the query
516	Authentication failed
517	The directory is read-only
518	Too many entries would be changed or deleted
521	Refused in addonly mode
522	Field is encrypted; value not shown
523	Expected answer or clear after login
529	Authentication method not supported
598	Unknown command

Code	Meaning
599	Syntax error

Appendix B

Limits

Item	Limit
Command line length	4095 bytes
Arguments per command	128
Argument length	1023 bytes
Search terms or pairs per command	64 (63 for change)
Response line length	about 4 KB
Login alias length	127 characters
Field name length	63 characters
Field description length	255 characters
Fields	256
[local] networks	32
Configuration string values	511 characters
Pairs for --add-entry	64

Appendix C

Sample Configuration Fields

The sample configuration defines these fields.

Index	Name	Length	Attributes
0	address	128	Lookup Public Change
1	phone	64	Lookup Public Default Change Turn
2	email	128	Lookup Public Default Change
3	name	256	Indexed Lookup Public Default Any
4	type	64	Indexed Lookup Public Default
6	alias	32	Indexed Lookup Public Default Unique
7	password	128	Encrypt Sacred NoMeta
8	proxy	32	Lookup Private
9	department	64	Indexed Lookup Public Default Change
10	title	64	Lookup Public Default Change
13	acl	128	Private Sacred
14	hours	128	Lookup Public Change
16	other	256	Lookup Public Default Change
19	office_location	128	Lookup Public Default Change
22	office_address	128	Lookup Public Change
23	nickname	128	Indexed Lookup Public Change
27	pager	64	Lookup Public Change Turn
32	office_phone	64	Lookup Public Default Change
33	home_phone	60	Lookup Public Change Turn
36	fax	64	Lookup Public Change Turn
47	group	32	Indexed Lookup Public
48	www	256	Lookup Public Change
100	empno	16	Indexed Lookup Private
101	supervisor	64	Lookup LocalPub Change
102	secretary	64	Lookup LocalPub Change
103	public_key	4096	Lookup Public Change
104	forename	64	Indexed Lookup Public Change
105	surname	64	Indexed Lookup Public Default Change

Appendix D

License

Copyright © 2026, Large Scale Systems Museum

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.