Quintus Prolog Release 3.4

for Sun-4

Release Notes
and
Installation Guide

Swedish Institute of Computer Science (SICS)
PO Box 1263, SE-164 29 Kista, Sweden

Release Notes and Installation Guide, April 1999.

This manual corresponds to:

Quintus Prolog Release 3.4

SICS
PO Box 1263
SE-164 29 Kista, Sweden
+46 8 633 1500
http://www.sics.se/quintus/

Authorization codes:
Admin Department
qpadmin@sics.se

Problem reports and Product Suggestions:
Technical Support Group
qpsupport@sics.se

Additional product or service information:
Sales Department
qpsales@sics.se

Mailing list additions:
quintus-users-request@sics.se

RELEASE NOTES FOR PROLOG

R.1-1: Revision History

This section highlights the changes between this and Release 3.1 of Quintus Prolog.

From release 3.1 to 3.2:

- Atom garbage collection, whereby memory for atoms no longer referenced by the Prolog execution can be reclaimed. With the introduction of this feature, the maximum length of atoms has also been increased from 1024 bytes to 65532 bytes. library(strings) has been modified to support the long atoms, only allocating space for long atoms as needed.
- The TCP library has been significantly enhanced to provide more efficient passing of terms between Prolog processes, more functionality for server processes and support for input callbacks, where predicates can be called either when a specified file descriptor becomes ready or at a specified time.
- A new licensing system has been introduced with this release. A license manager program is provided to setup the licenses and provide information on licenses installed.
- The cross-referencer tool has been replaced by a new version that is tuned to finding unreachable code in large programs. For details, refer to the `xref.doc` file in the tools directory.

From release 3.1 to 3.2:, the following bugs were fixed:

- `library(avl)`: procedure avl_fetch_1/5 is missing; misc. bugs
- `format/2` segmentation faults with large float exponents
- `set_input(user_output)` raises exception
- float unification in .qofs fails
- `qpc` syntax error message differs from development system
- `library(lists):nth1/3` description incorrect
- `use_module/2` information missing from .qofs
- `library(tcp)`: connection requests on callbacks functionality
- `library(date)`: `datime/7` format inconsistent with `time_stamp/3`
- `ord_list_to_avl/2`: creates avl tree with incorrect functor

From release 3.2 to 3.3:, the following bugs were fixed:

- Pascal Foreign Function Interface bug for `-string`
- bug in `library(putfile)`
- memory leak in `garbage_collect_atoms/0`
- stack shifter bug

- atom_chars/2 with 256 element list bug
- qpc permanent variables limit detection
- Sun C calling convention bug
- QUI interrupt dialog box has no buttons
- ProXL runtimes don't work due to module qualification
- qpc permanent variables limit detection

Release 3.4 features an Emacs-based source-linked debugger, with approximately the same functionality as the QUI-based debugger. See manual(d-1-3).

The GNU Emacs interface was thoroughly revised and supports GNU Emacs 19 and 20.

From release 3.3 to 3.4, the following bugs were fixed, as well as a large number of documentation bugs:

- current_predicate/2 failed after loading QOF-file saved by save_predicates/2
- re-exported predicates could fail if the unknown flag was fail
- dynamic code generation bug for imported predicates in runtimes
- bug in finding the correct module of an already loaded QOF file
- user:message_hook/3 not obeyed on restore
- undefined profiler predicate
- qpc bugs: input file suffix, all_dynamic(true) in embedded loads, handling -N and -o flags
- runtimes did not accept Module:Head :- Body clauses
- arithmetic bugs
- retract/1 could leave a dangling pointer
- ensure_loaded/1 did not work on QOF-files saved by save_predicates/2
- advice checking and debugging did not work when invoking a saved state
- DCG expansion of {Goal} bug
- C calling Prolog error handling bug
- re-exportation and meta-predicate interaction bug
- save_modules/2 failed to save a dependency corresponding to :- use_module(file,[]).
- GC bugs
- indexing bug upon loading QOF-files
- QP_seek() bug
- SIGPIPE signals cause abort instead of exit
- library bugs: strings:span_*/*, strings:midstring/*, strings:concat_atom/*, strings:concat_chars/*,

`directory:file_member_of_directory/2`, `heaps:get_from_heap/4`, `date:time_stamp/*`

- `proxl:current_window/1`, `proxl:current_display/1` could crash
- `library(tcp)` bugs: floating point byte swapping, `tcp_shutdown()`, non-lingering `tcp_listen()`, file name size limitation

R.1-2: Platform Specific Notes

R.1-2-1: Sun 4

Quintus Prolog release 3.3 did not comply with the SPARC ABI; release 3.4 does. Quintus Prolog release 3.4 runs under SunOS 5.5 or later. QUI requires Motif 1.2.6 or later.

Use the C compiler option "-K pic" to generate foreign files, and the linker option "-G" to generate shared objects for **load_foreign_executable**/1.

INSTALLATION GUIDE

R.2-1: About the Installation Guide

This *Installation Guide* describes the installation of Quintus Prolog on Sun-4 workstations, standalone or in a network. Please read it carefully. If you have any difficulty with these instructions, please contact SICS support by Email at qpsupport@sics.se.

R.2-2: Installation Overview

There are three basic steps to installing Quintus Prolog:

- unloading the installation files from tape (see section R.2-4, page 7)
- running the installation script to unload the rest of the Quintus software and perform post installation tasks such as building executables (see section R.2-5, page 8)
- running the license manager to setup the license file (see section R.2-6, page 10)

The installation files are concerned directly with the installation process rather than with the software you are installing. These consist of:

install3.4	The installation script you will run to install the software
installd3.4	Directory of component information and postinstall scripts used by install3.4

It is possible to unload the installation files in any location on any machine, provided that they are then accessible to the Sun-4 machine that will execute the installation script. The installation script must be run on a Sun-4 machine running SunOS 5.X (Solaris 2.X).

R.2-2-1: Quintus Prolog Components

Quintus Prolog is organized into a number of components to enable you to easily choose which pieces of the release you wish to install. The installation script lists the available components which can be installed.

The following components comprise Quintus Prolog Release 3.4:

pro	the core component that contains the development system, runtime generator and associated tools, plus the Quintus library, tutorial and demo files and the main help system files.
edit	support files for the interface between GNU Emacs and Quintus Prolog.

proxl A low level interface to the X Window system, built directly on top of the Xlib library.

proxt An interface to the Motif toolkit that provides access to Motif functionality for developing graphical user interfaces in Prolog.

qui the Quintus User Interface, providing a Motif based development environment that includes a source linked debugger for Prolog.

sys this component updates certain operating system files, for example to enable make(1) and file(1) to understand QOF files.

The **pro** component must be installed in order to run Quintus Prolog. All other components are optional, although we suggest that they are all installed if space allows.

R.2-3: Preparing For Installation

If possible, we recommend that you do all installation tasks for Quintus products while logged in as 'root' on the file server where the release will be unloaded and installed. This avoids many potential problems with directory ownership and protection.

However, Quintus Prolog has been organized so that only the optional **sys** component must be installed by 'root' (see section R.2-7, page 11); all other components can be installed by any user.

If you are installing Quintus Prolog on a remotely mounted file system where 'root' on the local client machine does not have 'root' privileges on that file system, we recommend that you install while logged in as 'bin' or some user.

The *quintus-directory* referred to throughout this *Installation Guide* is the directory into which you will be unloading your Quintus software.

You can choose any directory as *quintus-directory*, so long as there is sufficient space. A popular location is `/opt/quintus`. Your *quintus-directory* must be mounted (or mountable) by all machines on which Quintus Prolog is to be used. At the beginning of the installation process you will be asked to confirm your choice of *quintus-directory*.

If you have an earlier version of Quintus Prolog Release 3 installed, you can use the same directory for installing 3.4, provided that enough space is available. You do not have to use that directory and can use another one if you prefer.

The following table shows approximately how much disk space (in megabytes) each of the components requires. This assumes installation on both operating systems.

Component	Description	Space Required
pro	Quintus Prolog for Sun Sparc Platform	14.7
qui	Quintus Prolog Motif-based User Interface	1.8
proxl	Quintus Prolog interface to X windows	2.1
proxt	Quintus Prolog interface to Motif toolkit	1.2
edit	Quintus Prolog Emacs support files	0.5
sys	Quintus Prolog System Files	0.1
Total		35.7

Note that several hardware-operating system combinations can be unloaded and installed in a single *quintus-directory*. When this is done parts of the installation are shared, so the space taken by several systems is somewhat less than the sum of the individual systems.

Use the UNIX command df(1) to see how much disk space you have available in *quintus-directory*.

If you already have an earlier release of Quintus Prolog, you may wish to delete it prior to unloading Quintus Prolog Release 3.4 in order to free disk space for your new release. Deleting an earlier version of Quintus Prolog Release 3 does not invalidate saved states or other QOF files built with that release. Older QOF files will work with Release 3.4.

However, saved states will no longer be executable from the UNIX shell if you do this. In order to make a saved state executable from the UNIX shell, it can be loaded into the 3.4 system and then saved out again using the **`save_program`**/1 predicate to create a new saved state dependent on the new release.

WARNING: Do not rename an old installation directory to make the name available for the new release directory (unless you don't mind the old release ceasing to work). If you rename a Quintus release directory you have to redo the installation process since the directory name is inserted into a number of the executable files.

R.2-4: Unloading the Installation Script

To unload the installation script from the Quintus tape, insert the tape in your drive. We use `/dev/rmt/0m` as an example. Your drive may have a different name. If in doubt, consult your system administrator. To unload the installation script, type:

```
timsbury% tar xvfp /dev/rmt/0m
```

This will unload the 'install3.4' and 'installd3.4' installation files into the current working directory.

These files can be unloaded into any directory that is accessible to the machine on which the installation script will be run. We recommend, however, that these are unloaded into the *quintus-directory*.

If you intend to unload Quintus Prolog from the tape immediately, leave the tape in the tape drive. Otherwise, you can remove the tape and resume unloading at another time.

R.2-5: Running the Installation Script

The next step is to invoke the 'install3.4' script. The script will verify that you are running on an appropriate platform and print some information:

```
% ./install3.4

        Quintus Prolog Release 3.4 Installation

   Installer            : <your login name>
   Hostname             : <your hostname>
   Machine type         : <your machine type>
   Operating System     : <your OS release version>
   Date of installation : <date you installed Quintus>
```

The installation script then asks for the path of the *quintus-directory* offering the current working directory as the default:

```
Enter name of Quintus home directory [<default directory>]:
```

If the current working directory returned by pwd(1) is on an automounted filesystem mounted in `/tmp_mnt`, the script will recognize this and offer the network directory path (i.e. without the `/tmp_mnt`) as default. If your automounter mounts file systems in a directory other than `/tmp_mnt` then you should specify the network path for *quintus-directory* as this is needed during the installation process. The script will then print the path chosen for *quintus-directory* and the installation log file.

```
   Quintus directory    : quintus-directory
   Installation log file: quintus-directory/<log file>
```

Next, the script lists the available components to install:

```
   Select components that you wish to install.  You may type 'all'
   to select all components or else type the individual components
   on one line (e.g. pro edit qui).  Components marked with '*'
   appear to be already installed.

   The following components are available:

         edit  - Quintus Prolog Emacs support files  (0.3Mb)
         pro   - Quintus Prolog for Sun Sparc Platform  (22.5Mb)
         proxl - Quintus Prolog low level interface to X windows (2.6Mb)
         proxt - Quintus Prolog interface to Motif toolkit  (1.6Mb)
         qui   - Quintus Prolog Motif-based User Interface  (3.4Mb)
         sys   - Quintus Prolog System Files  (0.0Mb)

Select [all]: pro  qui edit
```

Some components depend on others - for example, most components depend on the **pro** component. The script will inform you if you select a component that is dependent on another component that is not selected or installed.

The **sys** component will only appear if you are running as 'root'. It will appear to be installed already (i.e. marked with '*') if `/usr/include/quintus/quintus.h` has been installed from a previous release of Quintus Prolog. We recommend that you install this again as this release contains an updated version of this file.

The script will then print the list of components it is about to install and ask if you wish to continue. If you type 'n' to this then the installation script will abort. Otherwise it will proceed to the next stage.

```
The following components will be installed: pro qui edit

Do you wish to continue? (y/n) y
```

It will prompt you for the host on which the tape drive is located, offering the local host as the default, and the tape device. If you specify remote host then a username is requested on that host as well. The script then waits for you to put the tape in the drive, if it is not already there, and hit return. Unloading will then proceed. For example:

```
Enter name of host with tape drive [timsbury]: paulton
Enter login name of user on paulton [bin]:
Enter name of tape cartridge device [/dev/rmt/0m]: /dev/rst8

Hit return when tape is mounted and unloading may begin...

% Rewinding tape
% Unloading files from tape on host paulton
```

If all components are selected, unloading the tape takes about 20 minutes. The files are then verified to ensure they are have been unloaded properly onto your system. If all components are selected then this takes about 5 minutes.

Once the files have been unloaded and verified, the postinstall scripts are then executed for each component that has one. With the exception of the **sys** package, these are not interactive and will just print informational messages about what they are doing. For example:

```
% Executing pro install script
% Setting paths in executables
% Building prolog
% Executing qui install script
% Building qui
% Executing proxt install script
% Building proxt shared objects
```

The **sys** postinstall script will request some information about adding links to Prolog executables from a 'bin' directory. Refer to section Chapter R.2-7 for more details.

The install3.4 script has two options that control its actions:

-i	execute the postinstall scripts for the selected components only (do not unload any files from tape)
-d	delete installed components

The '-i' option is useful if you wish to move the quintus directory to another location in which case the post installation tasks must be re-executed.

R.2-6: Running The License Manager

Having installed Quintus Prolog, the license file must then be set up with a license code in order to run *prolog*, *qpc*, etc. This is achieved with the license manager program *qplm*, which resides in the Quintus runtime directory (for example *quintus-directory*/`bin3.4/sun4-5`).

The first step is to initialize the license file with *qplm -i <site name>*. The site name given to *qplm* here should be exactly the same as supplied with the code from Quintus since it is used to generate the code. For example:

```
timsbury% qplm -i Hallatrow Designers Inc.
```

Then the next step is to add the products and codes. This is done with the command *qplm -a <product> <users> <code>*. The *<product>* argument is a string that comprises the product name ('prolog' in this case) the platform and the version number. The Sun-4 release thus requires a code for `prolog/sun4-5/3.4`.

The *<users>* argument specifies the number of regular users of Quintus Prolog at this site and *<code>* is a 16 character string that is supplied by Quintus. To add a code for a 5 user license, issue the command:

```
timsbury% qplm -a prolog/sun4-5/3.4 5 thiscodewontwork
```

Quintus also provides temporary licenses that expire on a given date. The expiration date can be added with the command *qplm -a <product> <users> <expire> <code>*, where *<expire>* is an expiration date with the format YY-MM-DD. The following example adds a 2 user license that expires on 17 May 1994:

```
timsbury% qplm -a prolog/sun4-5/3.4 2 94-05-17 thisalsowontwork
```

You can print the information in the license file, including site name and products licensed with the *qplm -p* command.

You can delete regular users from the license file if they no longer use the product with the command *qplm -d <product> <user>*.

R.2-6-1: The License Directory

The licensing system maintains two files in the directory *quintus-directory/*`license3.4`:

license.qof	contains the license code(s)
users.qof	maintains a list of users of Quintus Prolog

The `users.qof` file is written to each time an "occasional" user invokes *prolog* or *qpc*, where an "occasional" user is one who has not used the product more than five times.

If *quintus-directory* resides on a file system that is mounted read-only on any machines where Quintus Prolog is run, then the license directory should be replaced with a symbolic link to a directory on a filesystem that is writable before running *qplm*. This is illustrated by the following commands, where *quintus-directory* is `/opt/quintus` and the license directory is moved to `/var/quintus/license3.4`:

```
timsbury% rmdir /opt/quintus/license3.4
timsbury% mkdir /var/quintus/license3.4
timsbury% ln -s /var/quintus/license3.4 /opt/quintus/license3.4
```

> **NOTE:** The `users.qof` file is intended as an *aid* for you to remain in compliance with your product license. Circumventing it, for example by deleting regular users or making it unwritable does not bring you in compliance with your license.

R.2-7: The Sys Component

The **sys** component performs installation operations that affect global system files and therefore can be done while logged in as 'root'. These operations are optional, but useful. Specifically, the **sys** component attempts to do the following:

- install quintus.h in `/usr/include/quintus/quintus.h`
- add lines to the default rules file the enables the make(1) command to recognize how to make QOF files

- add lines to the `/etc/magic` file to recognize QOF files and saved states.
- (optionally) create links to the executables in the Quintus runtime directory from the `/usr/local/bin` directory.
- (optionally) create links to the man pages in the Quintus directory from the `/usr/local/man` directory.

This component does not need any files from tape, so it can be run at any time after the **pro** component has been installed on a file system accessible to the local machine.

If, in your computing environment, operating system files reside on local disks then you may wish to install this component on each machine where Quintus Prolog is going to be used. Consult with your system administrator if you have any doubts regarding these procedures.

See section R.2-7-1, page 12 through section R.2-7-4, page 14 for descriptions of each installation task. If you can not become 'root' on your system, those sections also give advice on how you can get similar functionality without becoming 'root'.

R.2-7-1: quintus.h

The quintus.h "include" file contains definitions and declarations that are useful when using the Quintus Prolog Foreign Interface, embedding Prolog in another language, managing interrupts, etc.

The **sys** component postinstall script attempts to install quintus.h in `/usr/include/quintus` (creating that directory if necessary).

Once this is done, users writing C code to be used with Quintus Prolog can simply hash-include the file in their C source using:

```
#include <quintus/quintus.h>
```

If you do not install the **sys** component on your system then users who wish to use quintus.h can simply copy the file

quintus-directory`/generic/qplib3.4/embed/quintus.h`

into their working directories as needed.

R.2-7-2: Updating Default Rules File

The UNIX utility make(1) may be configured to ensure that QOF files are up to date by modifying `/usr/share/lib/make/make.rules` to recognize certain kinds of files generated and used by Quintus Prolog and various Quintus utilities.

The **sys** component postinstall script attempts to modify this file, adding the following rules at the end:

```
# Quintus Prolog Compiler (qpc) section
.SUFFIXES: .qof .pl
QPC=qpc
QPCFLAGS=
.pl:
         $(QPC) $(QPCFLAGS) -o $@ $<
.pl.qof:
         $(QPC) $(QPCFLAGS) -cN $<
```

Once in place, make(1) will be able to automatically update QOF files. QPCFLAGS can be set to pass *qpc* any flags needed when building your application.

If you do not install the **sys** component on your system then the same effect can be produced by including the above rules in any makefiles involving *qpc* and Prolog code.

R.2-7-3: Updating /etc/magic

The UNIX command file(1) determines the type of a file by examining its contents. By modifying the file `/etc/magic`, which records characteristics of files that file(1) recognizes, file(1) can be used to identify QOF files and QOF files which are saved states.

The **sys** component postinstall script attempts to modify this file, adding the following lines to the end:

```
0       string      QOF             Quintus Prolog Object File,
>10     short       x               Ver %d
>8      short       x               Rev %d
>12     long        0x04030201      (byte-swapped)
256     string      QOF             Quintus Prolog Saved-state,
>266    short       x               Ver %d
>264    short       x               Rev %d
>268    long        0x04030201      (byte-swapped)
```

If `/etc/magic` has been updated, and the file "camerton.qof" is a QOF file that has been built using Quintus Prolog Release 3.4, you should see:

```
timsbury% file camerton.qof
camerton.qof: Quintus Prolog Object File; Ver 1 Rev 71
```

If you do not install the **sys** component on your system then the same effect can be achieved by creating your own copy of the system's `/etc/magic` file and adding the above lines to it.

Your version of `/etc/magic` can then be used with the -m option to file(1) to recognize QOF files. If you have made your own copy of `/etc/magic` in `~/mymagic` and modified that copy as suggested, you can type:

```
timsbury% file -m ~/mymagic camerton.qof
camerton.qof: Quintus Prolog Object File; Ver 1 Rev 71
```

R.2-7-4: Links for Executable Files and Man Pages

The **sys** postinstall script will ask whether you wish to install links to Quintus executables in the directory given by the environment variable QPBINDIR, or `/usr/local/bin` if this is not set. If you answer no to this question then no links are installed.

In SVR4 environments, applications are usually installed in subdirectories of `/opt` and executables are located in a `bin` subdirectory of the application directory. This can be achieved with Quintus Prolog by creating a link from *quintus-directory*`/bin` to *quintus-directory*`/bin3.4/sun4-5`.

If you do not install these links then Quintus Prolog users will need to add the the Quintus runtime directory to their PATH in order to find the executables *prolog*, *qpc*, etc. For example:

`PATH=$PATH:`*quintus-directory*`/bin3.4/sun4-5; export PATH`

The **sys** postinstall script will then ask whether you wish to install links to Quintus man pages in the directory given by the environment variable QPMANDIR, or `/usr/local/man` if not set. If you answer no to this question then no links are installed.

Again, in SVR4 environments applications usually leave man pages in a `man` subdirectory of the application directory. This can be achieved with Quintus Prolog by creating a link from *quintus-directory*`/man` to *quintus-directory*`/generic/q3.4/man`.

Quintus Prolog users may need to add the the Quintus man directory to their MANPATH in order to find the man pages for *prolog*, *qpc*, etc. For example:

`MANPATH=$MANPATH:`*quintus-directory*`/generic/q3.4/man;`
`export MANPATH`

R.2-8: If Unloading Fails

If for any reason unloading fails, you will see diagnostic messages describing the reason for failure. Failure of unloading is usually due to insufficient space in *quintus-directory*, incorrect protections on *quintus-directory*, or an incorrect login name when installing.

If unloading failed because of protection problems with *quintus-directory* or an incorrect login name, it is unlikely that anything at all was unloaded from the tape. Once the protection problem is corrected, you can reattempt unloading.

If unloading failed because of insufficient space, you either need to free additional space for *quintus-directory* or choose a new *quintus-directory* where more space is available. In order to determine how much additional space you will need to unload Quintus Prolog, you may need to delete the partially-unloaded release using the -d option to install3.4. See section R.2-3, page 6 for the disk requirements of Quintus Prolog. Be sure that there will be enough space in *quintus-directory* to install as well as unload. Once sufficient space has been provided, you can reattempt unloading.

R.2-9: The GNU Emacs Interface

The **edit** component provides support files for an an interface between Quintus Prolog and GNU Emacs version 19 or 20. You can 'ftp' GNU Emacs from various sites free of charge, or buy it for media and duplication costs from:

```
Free Software Foundation          Voice:  +1-617-542-5942
59 Temple Place - Suite 330       Fax:    +1-617-542-2652
Boston, MA  02111-1307,  USA      gnu@gnu.org
http://www.fsf.org
```

In addition, to use the GNU Emacs interface, you must set an environment variable which tells Prolog where GNU Emacs is located:

```
timsbury% setenv QUINTUS_EDITOR_PATH /opt/gnu/bin/emacs
```

Now to invoke *prolog* with GNU Emacs:

```
timsbury% prolog +
```

After the Quintus Prolog banner appears, press the Control-x key followed by the Control-c key (^X^C) to exit.

More details can be found in `manual(d)`.

RELEASE NOTES AND INSTALLATION GUIDE