Quancept CATI 7.9.1

Table of contents

(Download this Readme document.)

What's New

Quancept CATI 7.9.1 includes patches and features that were issued since the release of Quancept CATI 7.9.

- New features:
  - `rqtip` supports remote interviewers connected via VoIP.
- Closer integration between systems:
  - Easier migration of projects between different UNIX® platforms (big endian / little endian).
  - Better support for scripts written in character sets other than ASCII or Latin-1.
  - Integration of PTF translations (Qolyglot, Quanquest, etc.) in the `qcset`-based utilities.
  - Full integration of customized SMS algorithms derived from Quancept versions v7.6.1 and earlier.
- Significantly increased limits:
  - In `qtip`, the maximum number of questions variables has been increased to 64533.
  - In the `.ddr`/`.drs`-files, the maximum size of a `mp` question has increased to 64 kB.
  - On Linux (only), the size of the `.drs`-files can exceed the 2 GiB limit.
  - `qcprt` can print maximum size questions (1000 iterations, 1000 responses).
- Other enhancements:
  - Better performance for scripted quotas.

- Various bug fixes, see below for details.

C compiler support

This Quancept version requires an ANSI C compiler. A pre-ANSI ("K&R") C compiler will still work in most cases, but not in `makeqtip`.

Installation

Quancept CATI 7.9.1 is installed in the same manner as previous Quancept CATI releases. Follow these steps to upgrade from Quancept CATI 7.9 or 7.8:

1. Login as superuser (`root`).
2. Navigate to the directory where Quancept 7.9.1 is installed (usually `/usr`; tar adds the relative path `qtime/qc/v7.9.1`):
   ```
   cd $QCHOME/../../..
   ```
3. Expand the tar file:
   ```
   zcat pathname/v7.9.1-unix.tar.Z | tar xf -
   ```

Use this file..	..for these operating systems..	..and processors
v7.9.1.hpux.tar.Z	HP® HP-UX 11.0	HP® PA-RISC
v7.9.1.linux.tar.Z	Red Hat® Linux® 7.2, Red Hat Enterprise Linux 3.0	Intel® Pentium®
v7.9.1.scounix.tar.Z	SCO® OpenServer™ 5.0.5, 5.0.6 and 5.0.7	Intel® Pentium®
v7.9.1.sol2.tar.Z	SUN® Solaris™ 2.5, 2.6, 7, 8 and 9	SUN® UltraSPARC®

4. On UNIX variants other than Linux, ensure that the desired `CompilerOption` is set in `$QCHOME/bin/qsms.conf`.
5. Verify that the selected compiler is in your `PATH`; add it if necessary.
6. Navigate to the new QCHOME and run the `qcinitial` script:
   ```
   cd $QCHOME/../v7.9.1
   bin/qcinitial
   ```
7. On Linux, `qcinitial` may need to run `makeqtip` in order to find the shared libraries. If prompted about shared libraries, allow the running of `makeqtip`.
8. Change `QCHOME` in all login scripts to point to the new Quancept version.
9. If you have not already done so, you are advised to change the file locking strategy from `LINK` to `LOCK`:
   ```
   echo LOCK >/ip/.qclock
   ```

If you re-install the operating system, you need to re-run `qcinitial`.

Compiling custom *callfunc*s

If you use custom callfuncs, copy the source files from your old `$QCHOME/qtip/src` to the new one. Afterwards, go to that directory and run `makeqtip`:

```
cd $QCHOME/qtip/src
cp ../../../v7.9/qtip/src/* .
makeqtip
```

Note: Set the `QCHOME` variable to the new install directory before running `makeqtip`.

Linux only: Setting character set

Quancept only supports single-byte character sets. Using multi-byte character sets like UTF-8 (Unicode) will cause data corruption, incorrect display of data, or program crash (core dump). The only exception is that `qparse` accepts script files in Unicode format, see below, section *Automatic Unicode conversion in qparse*.

Most Linux distributions will install UTF-8 (Unicode) as their default character set; they set the environment variable `LANG=en_US.utf8` or similar. Before running Quancept, you must change LANG to select a single-byte character set. To minimize text conversions, it is best to use a Windows™ character set (code page).

Code page	Description	For example used for (ISO 639-1 alpha-2 language code)
cp1250	Central European	bs cs hr hu pl ro sk sl sq sr
cp1251	Cyrillic	az be bg kk ky mk mo ru sr tt uk uz
cp1252	Western European	af ca cy da de en es eu fi fo fr ga gd gl id is it ml ms nl no pt qu rm se sv sw tn xh zu
cp1253	Greek	el
cp1254	Turkish	az tr uz
cp1255	Hebrew [1]	he
cp1256	Arabic [1]	ar fa ur
cp1257	Baltic	et kl lt lv

[1] Hebrew and Arabic require a BIDI-aware terminal or terminal emulator.

The LANG environment variable can be set in the user's login script or in a system system configuration file, often `/etc/sysconfig/i18n` (see `/etc/profile.d/lang.sh` for details). Simply substitute `utf8` with the code page from the table, for instance:

```
LANG="en_US.cp1252"
```

The locale must be defined in the operating system, otherwise collation (`atoz` sorting) and case folding (`'ABC'='abc'`) won't work. The command `locale -a` shows the currently defined locales. To add a definition, use the `localedef` command like this:

```
localedef -f cp1252 -i en_US en_US.cp1252
```

On some Linux distributions, you must install the “glibc-i18ndata” package before running `localedef` (look for directories `/usr/share/i18n/charmap` and

`/usr/share/i18n/locales`).

Environment variables

New variables in *qtip*

Name	Description
QCASKOVERQUOTA	If =0 (no) or =1 (yes), `qtip` will not ask *Interview now overquota - accept anyway (y/n)* when restarting a stopped interview.
QCCHKDUMMY	If =1, `qtip` allows *append/check verbatims* to change `dummyask` questions. This was the default behavior in Quancept 7.9 and older versions.
QCDEBUG	For diagnostics purposes, make `qtip` create a `debug.`*`pid`* trace file: =0 no file (default), =1 keep if crash, =2 keep if errorstop or crash, =3 keep if abnormal termination, =4 always keep.
QCQUOTAMARGIN	Allow overquota for the indicated percentage of pending quotas.

New variables in *qsms*

SMSDEFPATH	Colon-delimited list of directories with custom definition subdirectories.
SMSINCLUDE	Space-delimited list of directories for `#include`'d definition files.

Other new variables

Name	Description
QCLANGUAGE	Language, overrides LANGUAGE (which has a predefined meaning on Linux). Translations are taken from directory `$QCHOME/$QCLANGUAGE`.
QCSCRIPTPATH	Colon-separated list of directories for `qparse` to search for `include` files.
VVLANG	Locale setting for `vquac`; used for case insensitive compare (filter functions).

New default values

Name	Description
QCENV	Name of file with environment variables for `qtip`. In previous versions, if QCENV was not set, no file was used; now `qtip` and `supermen` look for a file `qcenv` (in the project directory).
VVTERMCAP	Name of file with Xenix-style terminal capabilities file for `vquac`. If VVTERMCAP is not set, `vquac` looks for `/etc/vvtermcap`. Previously, if this file did not exist, `vquac` would not start; now the program uses the distributed `$QCHOME/lib/vvtermcap`.

New commands

Quancept CATI 7.9.1 includes the following new commands:

`qtsxfer` (QTS only) - full three-party service
: Transfers a call to a third party, for instance, to a representative of the client on whose behalf the survey is conducted.
The respondent is put on hold while the interviewer dials the third party.
All three parties can be conferenced together, allowing the interviewer to introduce the respondent to the third party.

`qtsextn` (QTS only) - test whether the extension is busy
: Prevents a malicious interviewer from proceeding with an interview after the respondent has hung up.

`stopdial` (SMS only) - stop or start dialing for multiple projects (35117)
: Supervisors can use this script to stop or start dialing for all projects (end of day for instance), or for a specific list of projects.

`qtipcore` - diagnostics tool to analyze `qtip` core dump
: Identifies the last script statements `qtip` ran before it crashed.
This command is useful when a script is looping, or crashes in a custom callfunc. See the description in section *Support tools*.

qtsxfer command

This command transfers a call to a third party. For example, in a customer satisfaction survey, you can transfer particularly dissatisfied customers to a service representative of the client on whose behalf the survey is conducted.

- The respondent is put on hold while the interviewer dials the third party.
- While dialing the second call, the interviewer can send touch tones (DTMF) via the keyboard (for IVR systems).
- If the third party is unreachable, the interviewer can redial the second call or return to the respondent.
- After talking with the third party, the interviewer can conference all three together.
- The interviewer can clear the call to the third party, or drop off the call.

When used in a Quancept script with `callfunc('subprog')`, the scriptwriter can test whether a transfer took place, and how long the second call was active.

Usage:

```
qtsxfer [options...] phonenumber ["headerline"...]
```

Option	Description
`-c`	Display colors-on-black. Requires a `/etc/termcap` entry `Co#8` (Linux only).
`-l "headerline"`	Header line. This option can be repeated. The text can contain newlines.
`-f filename`	Name of file with additional header lines. Maximum line length 80 characters.

`-q`	Read additional header lines from `qc_`*`XX`*`out.txt` and delete this file. Use `callfunc('varlist')` to create the file.
`-h` *`hostname`*`:`*`port`*	Dialer's hostname. If not specified, use the value of environment variable `QCDIALER=`*`hostname`*`:`*`port`*`/`*`extension`*
`-x` *`extension`*	Local extension. If not specified, use the value of environment variable `QCDIALER=`*`hostname`*`:`*`port`*`/`*`extension`*
`-t` *`trunkgrp`*	Trunk group to dial *`phonenumber`* on (default: 0).
`-i` *`callerID`*	Originating telephone number, or `"blocked"`.
`-n` *`seconds`*	No-answer timeout for *`phonenumber`* (default: 120).
`-d` *`dtmf`*	Touch-tones to send when *`phonenumber`* answers.
`-r` *`filename`*	Sound file (on dialer) that will record the (last) call to *`phonenumber`*.
`-p` *`projname`*	Project name for which to account the call to *`phonenumber`* (in `call.log`).
`-g` *`extngrp`*	Extension group for calls on hold (default: 996).
`-o` *`filename`*	Create file showing the return status, and the time connected to each call. Use `callfunc('readfile')` to read the file.
`-y` *`filename`*	Export all localizable strings to this `.lng` file. After translation, copy the file to `$QCHOME/$LANG/qtsxfer.lng`.

The output file (option `-o` *`filename`*) provides a log of how the interviewer spent their time:

Content	**Description**
`exit=0`	Value =0 successful transfer, =1 error, =2 still connected to respondent.
`idle=`*`secs`*	Seconds not connected. Idle time before redialing a failed call.
`call1=`*`secs`*	Duration (seconds) connected to the respondent (first call).
`call2=`*`secs`*	Duration (seconds) connected to the third-party (second call).
`both=`*`secs`*	Duration (seconds) connected to both the respondent and the third-party (conferenced together).
`dial2=`*`secs`*	Duration (seconds) waiting for the second call to complete.

Independent of `qtip`, the program can be used with the `remote` utility to dial two numbers and conference them together:

```
pqtsxfer phnumber1 [options...] phnumber2 ["headerline"...]
```

Alternatively, using a remote extension:

```
rpqtsxfer [-D dialer] phextension phnumber1 [options...] phnumber2
["headerline"...]
```

User interface

When entering `qtsxfer`, the *`headerlines`* are displayed. You can use the arrow keys to scroll up and down in cases where there are more header lines than can fit on the screen. At the bottom of the screen, the following menu displays:

```
                                                             1st call: Connected
 R = dial 2nd call
 Q = quit
```

When you press `r`, the first call (respondent) is put on hold and you hear the second call (third-party) being dialed. The program continuously shows how long the first call has been on hold, which helps you avoid leaving the respondent waiting for an unreasonable length of time.

When the second call is answered, the menu changes to:

```
 J = join conference                                           1st call: On hold   0:35
 H = hangup 2nd call
 Q = blind transfer
 0..9*# = send touch tone
```

The keyboard can be used to send touch tones when connected to an IVR system. Pressing `j` joins the two calls in a 3-party conference; the menu changes to:

```
                                                                  1st call: Joined   0:08
 H = hangup 2nd call
 Q = leave conference
```

Pressing `q` leaves the conference and returns to `qtip`. The respondent and third-party remain connected through the dialer until either party hangs up. Alternatively you can press `h` to hangup the third-party and return to the respondent.

Example

The following script sample, from a customer satisfaction survey, shows the hand-off to a service representative:

```
comment Phone number of service center
        set phcenter = '15551234567'
comment Status file name must be unique for this record (data serial number)
        set statfile = 'tmp[+respondent+]'

comment Don't attempt a call transfer in test mode or during a snap
        set x = ifsnap
        set y = testingrun
        if (x >< 0 .or. y >< 0) { goto noxfer }

        callfunc('subprog', 'qtsxfer',
          '-l "I will now put you on hold while calling the service 
presentative."',
          '-l "Please stay on the line, it will only take a moment."',
          '-o [+statfile+]', phcenter)
comment Read and delete status file; interview is complete if transfer took place
        callfunc('readfile', statfile, 'exit=', result)
        callfunc('subprog', 'rm -f [+statfile+]')
        if (result = '0') { signal 1 }

comment Transfer did not take place, continue survey
```

```
noxfer  continue
```

Note: This example will not work if environment variable `QCSET=0` (Quancept v7.6.1 compatibility mode).

Sample script

The script `$QCHOME/qctest/xfer` demonstrates how to use `qtsxfer`. The comments contained in the script explain specific details (snapback, review of completed interviews, etc.).

Note: Before using this script, insert the phone number to be dialed in the line `set phcenter = '15551234567'`

Preparing the dialer

- Minimum software version for using `qtsxfer`: mrDialer 2.1 or QTC 1.6.20.
- Call transfer seats: These are extra extension pairs for transferred calls. The number of *Call transfer seats* must at least match the number of simultaneous calls to the third-party. Use the dialer's `setup` program: *Base Parameters*, *Advanced options*, *Call transfer seats*.
- Sound file `parked.wav`: The dialer plays this file to calls that are on hold. You can create a sound file in one of the formats the dialer supports, and place it in the dialer's audio directory. A typical file contains quiet, non-intrusive waiting music. A single playback serves all held calls; a respondent might not hear the recording from the beginning.

Warning: Contact your Support representative before using the `setup` program if the dialer is configured in a non-standard fashion (the configuration files have been hand-edited).

qtsextn command (41484)

This command retrieves the dialer extension status (busy and/or off-hook). The command can be used in the Quancept script to determine whether the interviewer is actually connected to the respondent.

Usage:

```
qtsextn [-h hostname] [-o filename] [-p port] [-t seconds] [-v] [-x extension]
```

Option	**Description**
`-h` *`hostname`*	Dialer hostname; if not specified, the value of environment variable `QCDIALER=`*`hostname`*`:`*`port`*`/`*`extension`* is used.
`-o` *`filename`*	Output file (default: standard output).
`-p` *`port`*	Dialer port number (default 7000).
`-t` *`seconds`*	Seconds to wait for reply from dialer (default: 10).
`-v`	Verbose, show trace of QSAMP messages.
`-x` *`extension`*	Extension number from which to gather status; if not specified, the value of environment variable `QCDIALER=`*`hostname`*`:`*`port`*`/`*`extension`* is used.

The output file (option `-o` *`filename`*) contains lines indicating the extension's status:

Content	**Description**

`error=0`	Value =1 if failing to contact the dialer.
`busy=0`	Value =1 if the extension is busy.
`offhook=0`	Value =1 if the extension is off-hook.
`avail=0`	Value =1 if the extension is available for group dialing.

Example

The following script sample writes the extension's status to a temporary file `busy`*`54321`*`.tmp` where *54321* is the record's serial number. If the call is no longer connected, the interview is terminated with signal 8 (abandon).

```
comment Write the extension's status to tmpfile
        set tmpfile = 'busy[+serial+].tmp'
        callfunc('subprog', 'qtsextn -o [+tmpfile+]')
        callfunc('readfile', tmpfile, 'busy=', result)
comment Remove tmpfile to avoid cluttering the directory
        callfunc('subprog', 'rm -f [+tmpfile+]')
comment Abandon the interview if the extension is not busy
        if (result = '0') { signal 8 }
```

stopdial command

A new shell script `stopdial` can be used to stop and restart dialing for all projects or for a list of projects. The command behaves exactly like SMS supervisor options `d` (*Stop all dialing*) and `e` (*Restart all dialing*).

The `stopdial` command can only be run by supervisors with "all" permissions. To this end, the `$QCHOME/sms/permissions` file must contain an entry similar to the following:

`username``:!*-`

Usage:

`stopdial [-u | -m `*`message`*`] [-a | `*`projects...`*`]`

Option	Description
`-u`	Restart dialing. If this option is omitted, dialing is stopped.
`-m` *`message`*	Message displayed in `qtip` when the interviewer requests another interview. If option `-m` is omitted, the script prompts the user to type-in the message.
`-a`	Start or stop dialing for all SMS servers. If this option is omitted, dialing is only started/stopped for the specified *`projects...`*

For example, to stop dialing for all projects at the end of the day, you could use the command:

```
stopdial -a -m "Time to go home!"
```

Data handling updates

See also section *Data handling in qtip*.

Feature changes

Maximum line length in `.ddr`-files increased from 4.5 kB to 64 kB (3156)

> Previously, the length of lines in stop files and in `.ddr` and `.drs`-files was limited to 4.5 kB. Data would be truncated when the aggregate length of the selected precoded texts, plus the length of the `other` text, exceeded this limit.
> The limit now is 64 kB in `qtip` and `quac`, and unlimited (except by memory) in `qcformat`.

Support for `.drs`-files larger than 2 GiB on Linux (32119)

> Until now, the size of the `.drs`-file was limited to 2,147,483,647 bytes. On Linux, this limit has been removed; `qtip` and `qcformat` write and read the `.drs`-file in 64-bit mode.
> Before generating `.drs`-files >2 GiB, ensure that the programs you use to analyze the data are capable of reading large files.

Enhancements

New option `qcformat -t` to recreate the `.tex` file

> This option is primarily intended for data recovery following corruption. It can also be used to reformat the `.dat` and `.tex` files (for example, after changing `datamap cardcols` or after adding or removing `nodata`).
>
> - As in previous releases, `qcformat` writes the closed responses to *`projname`*`.qdt` (same format as the `.dat`-file)
> - The new option `qcformat -t` also writes the open-ended texts to *`projname`*`.qdx` (same format as the `.tex`-file)
>
> You can rename the `.qdx`-file to `.tex` and use it for coding (run `qcdedup` and `precode`).

The option `qcformat -i` no longer requires a `condatf` file

> When reading individual `.ddr`-files with option `qcformat -i`, you were previously required to create a file *`projname`*`.ddr/condatf` with a list of all serial numbers to be processed, one number per line. This method can still be used. However, if no `condatf` file is found:
>
> - `qcformat -i` automatically uses all data files in all *`projname`*`.ddr/sub`*`N`* directories.
> - Without option `-i`, it uses all records in the *`projname`*`.drs` and *`projname`*`.drx` files (if they exist).

Better error reporting from `qcformat`

> `qcformat`'s log file output has been improved. In particular, `qcdf.log` contains more detailed diagnostics regarding errors in individual `.drs`-records or in individual `.ddr`-files.

Bugfixes

Precoded texts containing single quotes (`'`) could cause data truncation

> This affects `qtip` (restart of stopped interview, review of completed interview), `quac` (option `?`) and `qcformat`.
> Certain combinations of single quote followed by semicolon would prematurely terminate the

reading of stop files (`.drs` and `.ddr`-files). In the affected question, the `other` text and all `mp` responses mentioned later than the `'`-response would not be read.

In `qcformat`, a `mp` question where code 127 is mentioned could cause data truncation

Code 127 caused `qcformat` to stop processing the question, skipping the `other` text and all later mentioned responses.

With `qcformat`, data containing an asterisk (`*`) were written incorrectly (28884)

As a consequence, subsequent `mp` variables on the same card could have wrong values.

Data handling - database segments

Overview

- Live projects and test projects can share database segments. This simplifies administration of database files.
- All database files can be stored in a common directory, allowing them to be loaded with the command `shm *` at system start.
- You now can use 200 `resp dbase` questions per project. When hitting the system limit of `SHMSEG` segments per process, `qtip` detaches and re-attaches database segments.
- Database segments can be replaced on-the-fly while interviewing is in progress, see below for details.

Feature changes

New environment variable QCSHMPATH to search for database segments

The QCSHMPATH variable syntax is similar to PATH. When running a `resp dbase('`*`name`*`')` statement, `qtip` tries to locate the *`name`*`.shm` database segment file in the following sequence:

- in each directory in QCSHMPATH (left-to-right)
- in the project directory

For instance, if you want to place all databases in a common directory, set QCSHMPATH to that directory's pathname. Alternatively, if a test project should look for database segments in the main project's directory, create a `qcenv` file in the test project's directory containing this line:

```
QCSHMPATH /path/to/main/projectdir
```

Allow up to 200 `resp dbase` questions in a script, independent of SHMSEG

Previously, when exceeding the system configuration limit SHMSEG, `qtip` would errorstop (*Cannot attach* etc., *Too many open files*). Now it detaches and re-attaches segments as necessary to stay below the limit.

Do not reload segments on-the-fly (next paragraph) in projects with ≥SHMSEG `resp dbase` questions, otherwise the re-attach might fail, causing errorstop: *Can't map DBASE filename(segmentID): Identifier removed*.

Database segments can be replaced on-the-fly while interviewers are working

In previous Quancept versions, there was a brief period where neither the old nor the new database segment were available, causing `qtip` to errorstop. Now both `shm` and `supermen` create the new database segment before deleting the old one.

- Existing `qtip` sessions that have already reached this `resp dbase` question continue to use the old database.

- New `qtip` sessions use the new database segment.

On-the-fly replacements are only safe for scripts with up to (SHMSEG − 1) `resp dbase` questions such that database segments never are detached (see previous paragraph).
Most operating systems have SHMSEG=16 or higher. Refer to your system documentation for information on changing SHMSEG.

Quancept interviewing program - *qtip*

Data handling in *qtip*

See also section *Data handling updates*.

Feature change

`qtip` now refuses to start if environment variable `LANG` is set to Unicode (UTF-8)

Quancept does not support multi-byte character sets like UTF-8 (Unicode). If `LANG=en_US.UTF-8` (or similar), the data files might be truncated with error EILSEQ (invalid multi-byte sequence). This happens on newer Linux and Solaris systems when the data contain non-ASCII characters (like `àñß€`).
To prevent loss of data, `qtip` refuses to start, printing: `Cannot run with UTF-8 character set`. If this occurs, change environment variable `LANG` to a single-byte locale as described under *Installation*, section *Setting character set*.

Append/change verbatims no longer shows `dummyask` variables (33809)

This prevents interviewers from changing `dummyask` variables that are used, for example, to store sample data.
To allow *append/change* of all verbatims, set environment variable QCCHKDUMMY=1.

Bugfixes

In Quancept 7.9, open-ends from a stopped interview were missing in the de-duplicated `.dtx`-file

The restarted interview would always write a blank card, making `qcdedup` remove the original data (41229)

In Quancept 7.9, a `fix` variable could affect a subsequent `mp` variable (43360)

`qtip` was off by one column when converting a `fix`'ed asterisk `*` into its EBCDIC code 4-8-11:

- If the first column of a `fix` contained an asterisk (`*`), it was be not converted, but misrepresented as digit `6`.
- If a `fix` was followed by an `mp force` variable, its first column was always coded as responses `4`, `8` and `dk` (4-8-11).

Only the `.dat` and `.ddt` files were affected; the `.ddr` and `.drs` files were correct.

Snapback in a restarted interview could cause “dirty” open-ends in the `.dtx`-file

This happened when open-ended texts from the original interview became non-valid:

- If routing was changed such that the `coded` or `other` question came off-path.
- If `other` and all `promptoth` responses were de-selected. (8537)

In these cases, `qtip` now writes a card with blank text to the `.tex` file.

Snapback could take a `#SUB`-survey question off-path (4037)

This happened when snapping back past the `#SUB` question and moving forward.

With `noblank`, snapback in a restarted interview could cause “dirty” data in `.ddt`-file

If a card was omitted because all questions came off-path, `qcdedup` would re-use the previously saved card.

To prevent this, `qtip` now always rewrites previously saved cards, even if they have become blank.

The `noblank` statement had no effect when `datamap serialcols=6` or higher

`qtip` only omitted cards from the `.dat`-file when columns 8-80 were blank.

Appointments made from `gotosms` in a restarted interview caused an error exit from `qtip`

If a stopped and restarted interview is terminated, with an appointment, without writing a stop record (`gotosms` + select *Appointment*), `qtip` failed to reset the `stoploc` variable in SMS.

When the appointment was due, the new `qtip` would try to retrieve the non-existing stop record, causing an error exit with the message *Could not restart interview 1234 - Hit <rtn> to continue*.

When reviewing a completed interview, `fix` variables changed the data type to `dbase`

This affected the `.drs` file (data type 9 instead of 8) and the evaluation of `if` or `route` statements testing the value of a `fix` variable.

Failure to write data (for instance, disk full) was not always detected

`qtip` now prints a message on the terminal: `Cannot write` *`file`*.

Core dump if restart or review of interview with a `mp` question with >128 responses (34382)

The buffer for multi-punched responses was too small.

Core dump due to buffer overrun in `callfunc('putsmvar')` (46202)

`qtip` now errorstops if attempting to put more than 4096 bytes of data (incl. terminating NUL bytes).

Errorstop if two `qtip` sessions tried to simultanously create directories (41672)

This could happen, for example, when creating the *`projname`*`.ddr/sub100` directory or its `USED` subdirectory.

Lock files `.L` were not deleted after errorstop

Errorstop could leave the data files locked, preventing other `qtip` sessions from working.

`qtip` now attempts to delete its `.L` files before an abnormal termination.

In test mode (option `-t`), `qtip` did not always write an accounting record (34826)

The `.qca` record was only written if `qtip` failed to get a sample (timeout) or was stopped by a signal (`supermen` option 4 suboptions 2-5).

The `.qca` record is now always written for test interviews, just like for ordinary interviews.

Escape sequences like `'\n'` (newline) were not recognized by `callfunc('rfprint')`

Now the C escape sequences in the table are accepted. Hexadecimal encodings like `'{#0a'` are still working.

Escape	**ASCII**	**Meaning**		**Escape**	**ASCII**	**Meaning**
`'\a'`	BEL	Audible bell		`'\r'`	CR	Carriage return
`'\b'`	BS	Backspace		`'\t'`	HT	Horizontal tab
`'\f'`	FF	Formfeed		`'\v'`	VT	Vertical tab
`'\n'`	LF	Newline		`'\\'`	'\'	Backslash

Scriptwriting with *qparse* and *qtip*

Feature change

Ability to specify output file with `callfunc('rfopen')` (42442)

This `callfunc` now accepts one, two or three arguments:

Argument number	Description
1	Report file identifier (`1..4`). This argument cannot be omitted.
2	Report file name. If omitted, the file name is `REP`*`n`*`00123`, where *`n`* is the report file identifier (argument number 1), and `00123` is the interviewer number specified in `/ip/users`.
3	Access mode. Must be `'w'` for write or `'a'` for append. If omitted, the file is opened for append access.

Example:

```
COMMENT Write interviewer's name to file 'tmp1234.txt' (1234 = record serial)
        callfunc('rfopen',  1, 'tmp[+RESPONDENT+].txt', 'w')
        callfunc('rfprint', 1, '%s\n', '[+INTNAME+]')
        callfunc('rfclose', 1)
```

New keyword `scriptorder` to arrange the data map like the script

Option `qparse -m` was accidentally omitted in Quancept 7.9, causing error *Unrecognized option*. (35701)

Previously the option `qparse -m` determined how the variables (`ask`, `dummyask`, `fix`) should be arranged in the data map: either in the order they are defined, or in the order they are referenced. Using a command line option to this end is dangerous; if a scriptwriter accidentally omits option `-m` when re-parsing the script, the data are ruined.

To allow the option to be coded directly in the script, a new keyword `datamap scriptorder` has been introduced. It acts as if option `-m` has been specified. See the table below.

With option `-m` and/or with `datamap scriptorder`	**Without option `-m` and without** `datamap scriptorder`
Variables are in the order they are defined in the script (that is, in the same order as their labels).	Variables are in the order they first appear in the script (defined or referenced; for example, `unset`, `callfunc`).
OTHER columns are placed immediately after the question variable they belong to.	OTHER columns might be separated from the categorical variable if the latter is referenced before it is defined. (4412)

Enhancements

Allow larger scripts

The script size was often limited by the size of the data map in the `.qoc` file. The map contains one item per iteration of a question, `other` or `fix`. Exceeding the map space could cause sporadic core dumps in `qtip`.

- Overflows of the data map and of subscripts in nested loops are now detected by `qparse`.

- The data map space in the `.qoc` file has been increased from 32,765 to 65,533 elements.
- Items without data (`nodata` and `other(0)`) are no longer stored in the map.

Combined with `datamap cardcols=3`, these changes allow for much larger scripts.
The `.sum` file shows the data map utilization:

```
QCPARSE - 0 errors, 0 warnings
   18 Files read
 2112 Lines read
55470 Questions
    0 Questions with other
  173 Variables
  108 Temporary variables
  196 Labels
   19 Gotos
   23 Loops
    3 Max loop nesting level
  320 Max iterations per level
51200 Max iterations nested
55472 Map items
51442 Map items without data
Qoc size  577564
Data recs 75
Data cols 5419
```

Clarification

Always place a label on statements with randomization or rotation

To preserve routing, `for rot`, `for ran`, `ranstart` and `rotstart` statements should always have a label.
The label is used to compute the randomization seed. If the statement itself does not have a label, the preceding label is used, or the position in the `.qoc` file. In either case, a simple change further up in the script might change the seeding. This can change the path such that you will not be able to restart stopped interviews, or review completed ones (QCCOMP=1).

If you translate the PTF-file with Qolyglot, or manually, place a label on all `'`*`text`*`'`-driven `for` statements

For multi-lingual projects that do not use `ptftran`, you should place a label on all `for` statements of the type:

```
for i='this'/'that'/'the other'
```

The label serves to preserve translations across script changes. After changing the script, you can import the translations from the existing PTF-file into the newly parsed project:

```
qparse -ptfi projname.ptf projname
```

Without the `for` labels, `qparse -ptfi` will not import the translations of the iteration list (`'this'`, `'that'` etc.), or might assign them to the wrong `for` loop.

Scripted quotas - *qtip*

Enhancement

Performance improvement for quotas

To reduce contention for quota cells, the Temp count is no longer incremented when a quota check fails. The old behavior can be restored by setting environment variable QCOLDQUOTA=1.

Feature change

`stop decrm` does not provide control over which quotas are decremented (44596)

This statement stops an interview but counts the specified quota(s) as complete (incrementing the Temp and Complete counts). For example, you use `stop decrm` at the end of the recruitment phase of a callback project such that you know how many recruitments you have made.

The particulars of `stop decrm` have changed multiple times:

- Until Quancept 7.8.10, all quotas were completed, even those after the `stop` statement. (3296)
- From Quancept 7.8.11 to 7.9, only the specified quota was completed. (44596)
- In Quancept 7.9.1, you can specify which quotas to complete:
  - `stop 'stop' decrm='b2,z1,z2'` completes the quotas with suffix `b2`, `z1` and `z2`.
  - `stop 'stop' decrm='z*'` completes all quotas with a suffix starting with `z`.
  - `stop 'stop' decrm='*2'` completes all quotas with a suffix ending with `2`.
  - `stop 'stop' decrm='*'` completes all quotas.

When restarting an interview, suppress the question *Interview now overquota - accept anyway (y/n)*

`qtip` asks the question when a `quota` that was checked (passed) before the interview was stopped meanwhile has been filled (quota check fails). If you do not want to leave the answer to the interviewer's discretion, set environment variable QCASKOVERQUOTA:

Value	Acts like	
QCASKOVERQUOTA=0	No	Terminate with signal 5 (quota exceeded)
QCASKOVERQUOTA=1	Yes	Continue without incrementing the quota

Allow the quota value to be exceeded in surveys with a low cooperation rate

When quotas are almost full, you can improve performance by allowing quotas to exceed their limit. Normally the Temp count prevents new interviews from entering a quota cell if there already are enough active interviews to fill the Target. If you expect a significant percentage of the active interviews to be abandoned, it can be beneficial to allow some excess interviews to start, even at the risk of exceeding the Target. To this end, set environment variable QCQUOTAMARGIN to the percentage of extra interviews to permit. A quota cell passes when:

```
Temp + (Temp - Current) * QCQUOTAMARGIN / 100 ≤ Target
```

Bugfixes

Errorstop *closequotas: unknown exit flag* when using unusual `signal` values (2632)

This occurred when using `signal` values 3, 4, 7 or ≥12. Now a message is printed and the quotas are rolled back (like `signal 2`).

Errorstop *Unknown quota (do_quota)* after adding `quota` statement (41470)

When a stopped interview was restarted after the number of quotas had changed, `qtip` reverted to the old number of quotas. This subsequently caused an errorstop when attempting to access the no longer valid quotas.

Counter quotas (`quota pass`) increment even though the quota check failed

Counter quotas are used to skip individual sections in an omnibus survey. Example:

```
hobby   ask 'Are you interested in...  [INT: READ LIST]'
        resp mp ran 'cars'/'sport'/'music'/'travel'
comment -- If sport was selected, run the sport section --
sporty  dummyask 'Remember quota across stop/restart'
        resp num 1 nodata
        set ok = bit(qintr/2)
        if ( .not.ok ) { set sporty=0  goto nosport }
comment -- Check whether the sport section's quota is full --
        quota sex/age file='z1' pass=ok
        if ( ok ) { set sporty=1 }
        if ( sporty=0 ) { goto nosport }
        include 'sport.qqi'
nosport continue
        ... other questions ...
        signal 1
        end
```

In interviews terminated by `signal 1` (complete), the Complete counts for all quotas were incremented, even for counter quotas that had failed to check (Temp ≥ Target). This inflated the Complete count such that too few interviews completed this section.
Now quotas are only incremented when necessary (see the table below). The old behavior can be restored by setting environment variable QCOLDQUOTA=1.

Synopsis

This table compares the old and new quota logic. To restore the old behavior, set environment variable QCOLDQUOTA=1.

Interview terminated with	**Quota passed or full?** [1]	**Quota count changes**								**Comments**
		Quancept 7.9 or QCOLDQUOTA=1				Quancept 7.9.1 and QCOLDQUOTA=0				
		Check	End of interview			Check	End of interview			
		Temp	Temp	Curr	Over	Temp	Temp	Curr	Over	
Complete SIGNAL 1 or 6 STOP DECRM	Passed	+1	0	+1	0	+1	0	+1	0	Completed
	Full	+1	0	+1	0	0	0	0	+1	Failed attempt
Quota full SIGNAL 5	Passed	+1	−1	0	+1	+1	−1	0	0	Rolled back
	Full	+1	−1	0	+1	0	0	0	+1	Failed

										attempt
QUOTA w/o PASS Quit or abandon SIGNAL 2 or 8 STOP w/o DECRM	Passed	+1	−1	0	0	+1	−1	0	0	Rolled back
	Full	+1	−1	0	0	0	0	0	0	Rolled back

[1] *Passed* means that `Temp < Target` for at least one cell, or - in the case of a restarted interview - that the question *Interview now overquota - accept anyway (y/n)* was answered `y`.

Clarifications

When using counter quotas, always use `signal 1` before `end`.

If you want to Complete an interview, even though a `quota pass=`*`varname`* quota check failed (see above), you must explicitly use `signal 1` to terminate the script. Without the `signal`, the script implicitly terminates with signal 5 (quota exceeded) when:

- the last `quota` statement before `end` returned `pass=false`, and
- no snapback or snap forward took place between that `quota` statement and `end`.

Termination with signal 5 causes all quotas, even screener quotas, to be unchecked (rolled back).

Reviewing a completed interview with counter quotas requires explicit routing

When reviewing a completed interview (QCCOMP=1), `qtip` does not access the quota file; it always simulates that all `quota` statements returned as `pass=true`. Scriptwriters should remember the outcome (for instance, in a `dummyask` variable) and `route` around the section, otherwise the review will stop on the first question that was never asked.

Do not rely on `cell` being preserved when a `resp mp` question is quota axis

When multiple punches are selected, `cell` returns the cell number of one that was below Target. When re-evaluating the `quota` statement following a snapback, or during *change or inspect*, data may have changed such that another `cell` is selected.

In particular, if you use `cell` to decide routing, a snap might take the already answered questions off path (data lost). If you cannot avoid using `cell` for this purpose, set QCNOSNAPBACKS=1.

Other topics - *qtip*

Feature change

Environment variable QCENV now defaults to `qcenv`

The `qcenv` file (or the file whose name is `$QCENV`) is used for passing environment variables to `qtip`. Previously you needed to set environment variable QCENV to use this feature. This still works, but when QCENV is unset, it defaults to the (optional) file `qcenv`, placed in the project directory. Here are two examples that demonstrate its use:

- If a project requires QCASKNUMBER=1 (prompt interviewer for record key), place a `qcenv` file in the project directory with the line:

  ```
  QCASKNUMBER 1
  ```

- If a test project should use the same `resp dbase` database segments as the main project, give the test project a `qcenv` file containing the line:

```
QCSHMPATH /path/to/main/projectdir
```

If a custom algorithm requires particular environment variables, place a `qcenv` file in its definition directory; the `qsms` compiler will copy it to the project directory.
You can use `supermen` option 14 *Environment-variable manager* to add, delete, and change entries in the project-specific `qcenv` files; you can also generate a project-specific `qcenv` file from the user-supplied template file `$QCHOME/lib/qcenv`.

Bugfixes

The sign-on prompt *Enter extension number* did not accept the value `0`

When prompting for an extension number (QCSTATION=−9999), `qtip` now interprets the value `0` (zero) to mean “no dialer”.

Scrolling response texts spanning multiple lines left fragments of text on the screen (3261)

This could happen with `callfunc('setcols',1)` when the penultimate page had responses spanning multiple lines of text.

Errorstop *out of heap space* in `resp dbase` question (33525)

This occurs when entering a search pattern such as `e!`, selecting more than 3000 records. `qtip` now prints an error message *Too many responses* and continues.

Callfunc `subprog` did not allow escape out of the external program (control-C)

- If the subprogram did not explicitly enable signal handling, the interviewer could not escape (control-C).
- If the subprogram enabled signal handling, typing the suspend character (control-Z) terminated `qtip` (*Error reading input*).

To resolve this problem, the subprogram is now started in an isolated process group with signals enabled.

Echo and backspace are turned off after `qtip` timed out, waiting for SMS

`qtip` did not restore the terminal to “cooked” mode after error *Cannot contact sms server - Emergency exit*.

Appointments do not always use the best matching short-cut text (32021)

In German, entering `mi` made an appointment in a `M#inute` even though it matches exactly with `Mi#ttwoch` (Wednesday).

Missing translations added to `qctip.lng` language file

The message `stop forced by supervisor` is now translatable (stopped by `supermen` option 4 *Signal interviewer*).
The special responses `cancel`, `*pause`, `*redial`, `*hangup` have been added.
Note: If you want to disable a special response, change its text in `qctip.lng` to “`\b`”. This applies to the following texts:

Text number	English	Description
0321	ABANDON	Terminate interview early without saving data
0322	ACROSS	Display response lists ordered line-wise
0981	CANCEL	Unset a variable
0323	COMPLETE	Terminate early, marking interview as complete
0326	DOWN	Display response lists ordered row-wise

0218	QUIT	Terminate interview early and save data
0332	REGN	Redraw (regenerate) screen
0335	STOP	Stop interview, optionally save data and make appointment
0309	*COMMENT	Display comments (requires SMS)
0310	*DATE	Display current day and time
0311	*DEBUG	Enter debugger
0312	*DK	Forced *Don't know* response
0984	*HANGUP	Clear call without terminating script (QTS only)
0694	*LN	Ask for new language (0..9)
0315	*MSG	Send message to supervisor (not working)
0346	*NETWK	Show information about network connection (no action)
0316	*NOTE	Additional open-ended text to SP/MP/NUM question
0317	*NULL	Forced *No answer* response
0983	*PAUSE	Force ask for another interview (overrides QCANOTH)
0982	*REDIAL	Redial last respondent immediately (QTS only)
0344	*ROT	Ask for value of the rotation sequence
0345	*SEED	Ask for seed of the random number generator
0764	*SMS	Force GOTOSMS (requires SMS)
0319	*TIME	Display current time and when interview started
0320	*VERS	Display version of qtip and qoc file

Clarification

Available options in `qtip`'s SMS menu (`tipresps` screen)

The introductory SMS menu may, or may not, have two extra options. The prompt indicates which options are available: *Please enter choice (append p for pause), or r to redial*.

Option	Description
`p`	Pause, prompt for *Another interview (y/n)* at the end of the current interview. Used when environment variable QCANOTH=1.
`r`	Redial (autodialer, modem dialer) or redisplay from top (if the `tipresps` text and the comments are too long to fit on one screen)

With QCANOTH=1, `qtip` does not display the question *Another interview (y/n) ?* at the end of an interview, but automatically requests another record from SMS. The interviewer can force the question *Another interview* to appear:

- By entering `p` in the SMS menu, as indicated above.
- By entering the special response `*pause` during the interview.

Quancept script parser - *qparse*

Automatic Unicode conversion in *qparse*

This feature is useful if your Quancept scripts use character sets other than Latin-1. It allows you to write the script using Windows™ programs (*notepad* or your favorite word processor) and parse the output file with no prior conversion.

The prerequisites are:

- The Quancept script is saved as a Unicode text file (UTF-8, UTF-16 big or little endian).
- The interviewing terminals use one of the following 8-bit character sets (codepages):

Auto Conversion from Unicode to:

Codepage	Description
1250	Central Europe (Windows)
1251	Cyrillic (Windows)
1252	Latin 1 (Windows)
1253	Greek (Windows)
1254	Turkish (Windows)
1255 [1]	Hebrew (Windows)
1256 [1]	Arabic (Windows)
Auto Conversion is *not* possible to:	
Not 850	DOS Multilingual (also used on SCO OpenServer)
Not 878	KOI8-R (Cyrillic, RFC 1489)
Not 923	ISO 8859-15 (Latin 9)
Not 1051	Roman8 (specific to HP-UX)
Not 1257	Baltic (Windows)

[1] Hebrew and Arabic require a BIDI-aware terminal or terminal emulator.

To make use of this feature, write your Quancept script in *notepad* or your favorite word processor for Windows. To save the file in a folder that is mapped to your UNIX file system, follow these steps:

1. In the *File* menu, select *Save As*.
2. For *File name*, enter `projname.qqc`.
3. For *Save as type*, select *Plain Text* or *Text file*.
4. For *Encoding*, select *Unicode* (one of UTF-8, UTF-16BE, UTF-16LE).

The saved file can be parsed with the UNIX command `qparse projname`.

How it works

When presented with an input file in Unicode format, `qparse` determines which 8-bit character set is suited to represent the text, and generates a `.qc8`-file in this format. In Unicode mode, `qparse` also handles some word processor idiosyncracies, such as converting ‘smart’ quotes to 'straight' ones. `qparse` accepts the additional `.txt` suffix that some (older) Windows programs add to file names:

- The command `qparse projname` also reads `projname.qqc.txt`.

- The statement `include` *`script.qqi`* also reads *`script.qqi`*`.txt`.

Diagnostics

The *`projname`*`.err` file shows which character set has been auto-selected. If a script can be represented in more than one 8-bit character set, the alternatives are shown:

```
Unicode text converted for codepages 1252 (Latin 1) or 1254 (Turkish)
```

Warnings are issued if the script cannot be represented in any 8-bit character set. For instance, if a Greek script contains the character `ß` (U+00DF, sharp s) which cannot be represented in codepage 1253 (Greek), you will receive a message similar to:

```
projname.qqc:47: Warning: Unicode character not in the auto-selected codepage
 - auto-selected: 1253 (Greek)
 -  U+00DF needs: 1250 (Central Europe), 1252 (Latin 1) or 1254 (Turkish)
Warning: Conflicting codepages, Unicode text probably converted incorrectly
```

The defective script parses without any errors, but the text may not display as intended.

Other changes in *qparse*

Feature change

New options `-T` and `-F` to produce indented listings

These options print a listing of the script on the terminal or in the file *`projname`*`.lst`. This behavior is similar to the options `-t` and `-f`, except that `for` - `next` loops are indented. This can be useful in complex scripts with long loops.

Enhancements

New environment variable QCSCRIPTPATH with search path for `include` files

This feature can be used to keep `include` files shared between multiple projects in a common directory.
If an `include` file cannot be found in the current directory, `qparse` searches all directories indicated in `QCSCRIPTPATH`. The name of the chosen file is shown in the *`projname`*`.err` file:

```
projname.qqc.txt:9: Using INCLUDE screener.qqi.txt from /home/scripts
```

More questionnaire text elements are now translatable

The following table shows which text elements are stored in the PTF-file.

Stored in PTF-file	Simple texts	`ask`, `dummyask`, `display`, `protect`
	Lists of texts	`define`, `resp`, `for`, `set x=and()`, `set x=or()`, `set x=xor()`

Not stored in PTF-file	Missing	`set x='text'`
		`fix(10) 'a'+var+'b'`
		`callfunc('f','a','b')`
		`setstr(var,x,y,'text')`

		`onresp 'text'`	
	Special	`if (x='text')`	compares base language
	Non-translatable	`dbase('filename')` `set x=('uniqid')` `callfunc('funcname')`	and many others

Bugfixes

Lines starting with `include` produced errors (34135, 34638)

If a text continuation line started with the seven letters `include`, the `qparse` preprocessor would try to interpret the subsequent text as file name.
This example produces an error because `qparse` cannot find the file `d` (the first word after `include`):

```
q17a    ASK 'Would you like to be
included in our panel?'
```

Trailing quotes were missing in string constants (13266)

This occurred in strings like `'John''s and James'''`.

Large scripts could cause `qtip` to crash during start-up

For some statements, `qparse` overestimated the space `qtip` would need for string tables:
`dlist1  DEFINE(900) 'first'/'second'` (calculated as 900 string elements, correct is 2)
`FOR iter=dlist1` (calculated as 900 string elements, correct is 0)
`FOR iter=1 to 3000` (calculated as 3000 string elements, correct is 0)
If the miscalculation caused integer overflow (>32768), `qtip` would crash due to insufficient table space.

Literal `reals` with >9 digits were miscomputed (12671)

This was caused by integer overflow when computing the mantissa.

SMS-compiler *qsms*

Integration of Quancept v7.6.1 algorithms

Feature description

Previously, you could use the `qsmsold` shell script to compile custom algorithms written for Quancept v7.6.1 or older. This method had some limitations:

- The user compiling a SMS algorithm needed to know when to use `qsmsold` instead of `qsms`. Since the queue file (`.tq`) format is different, recompiling an ongoing project with the wrong command could destroy the queues. The `qsms` command now transparently compiles both old and new algorithms.
- When migrating to Linux, you could not continue to use customized v7.6.1 algorithms because

there were not any v7.6.1 libraries for Linux. Now, `qsms` always links with the latest version of the libraries.

Prerequisites

1. Install Quancept v7.6.1 (if not done already). The last component in the directory name must be the version number:

   ```
   /usr/qtime/qc/v7.6.1
   ```

2. If you are using an autodialer, also install the QTS files.
3. If the installed v7.6.1 is for a different machine architecture, delete its `bin` directory. This avoids confusion if a user accidentally tries to run the wrong program.
4. If the installed QTS files are for a different machine architecture, delete the following file (the path is relative to v7.6.1's installation directory):

   ```
   qts/include/expect2.o
   ```

Preparing the custom definition

5. Create a directory with your v7.6.1 definition files (`algorithms`, `tipresps`, ...), for example:

   ```
   /home/scripts/radio
   ```

6. In the definition directory, create a file:

   ```
   qsms.conf
   ```

   In this file, add a line indicating which QCHOME to use:

   ```
   QCHOME=/usr/qtime/qc/v7.6.1
   ```

   If you are using an autodialer, and this is a non-dialer algorithm, add:

   ```
   SMSNOQTSDIALER=T
   ```

   You also can add other compilation options, similar to Quancept 7.9.1 projects.
7. We recommend that you include the path to the definition directory in SMSDEFPATH. To this end, edit your Quancept 7.9.1 configuration file:

   ```
   $QCHOME/include/qsms.conf
   ```

   If SMSDEFPATH is not yet defined, or is commented out (`#`), add the line:

   ```
   SMSDEFPATH=/home/scripts
   ```

   Otherwise append the new entry to the existing search path, separated by **:** (colon):

   ```
   SMSDEFPATH=other entries:/home/scripts
   ```

   Note: This is not the name of the definition directory, but of its parent directory.

Compiling the custom definition

8. Go to the project directory.
9. Give the command `qsms`, followed by the name of the definition directory:

   ```
   qsms radio
   ```

   If you have not updated SMSDEFPATH (step 7), use the full pathname:

   ```
   qsms /home/scripts/radio
   ```

10. If you later need to recompile the custom definition with the same options, simply type:

    ```
    qsms
    ```

How it works

- `gener` recognizes the source version from the last component in the QCHOME pathname.
- `gener` selects the record layout (`.tq` file format) for this version (any version from v7c.3 and up).
- For source versions before v7.7, `gener` uses the newest `$QCHOME/sms/src/local.h` header file.
  This allows compilation even if the "C" compiler no longer supports 4.3BSD or System V extensions.
  To avoid conflicts, `gener` emits compiler flags to ignore ANSI C specific keywords (`const`, `__const`).
- `runsms` is always linked with the newest version of the Quancept libraries (`libnewt`, `libtrans`).
- For dialer algorithms, if `expect2.o` exists, it is linked into `runsms` (as SMSLOCALLIB).
  If `expect2.o` is missing, `gener` inserts replacement code in the generated `sms/local.h`.
  The replacement code makes `predictalg=3` a non-predictive algorithm; use `predict1` or `predict2` instead.
- The generated `sms/signature.c` file contains information how `runsms` was compiled.
  The command `qsms` (without any arguments) uses this information to recompile the project the same way.

Other changes in *qsms*

Feature changes

`qsms.conf` can be used on a per-project or per-definition basis

> You now can force particular compilation options by placing a `qsms.conf` file in the project directory, or in the directory with the definition files (`algorithms`, `menus`, etc.). Options that are not specified in a per-project or per-definition `qsms.conf` file are taken from the "global" `$QCHOME/include/qsms.conf` file.

SMSDEFPATH - search path for definition directories

> Custom algorithms that are used across many projects are easier to maintain when the definition files are kept in a central location. For instance, you might have two definition directories

```
/home/defs/radio
/home/defs/tv
```

By specifying `SMSDEFPATH=/home/defs` in the global `$QCHOME/include/qsms.conf` file, you can compile the `radio` definition with the shortcut `qsms radio` instead of `qsms /home/defs/radio`.
If a particular project requires customization of one of the definition files (for instance, a modified `greeting.tip`), just copy the modified file to the project directory before running `qsms radio`.
You do not need to use the old `qts-`*`algname`* scripts to copy files. Files in the definition directory that are required in the project directory (`.sc` prompts, `intquals` file etc.) are automatically copied by the `qsms` compiler.

SMSINCLUDE - search path for `#include` files

If custom algorithms are used by many definitions or projects, it is unpractical to maintain multiple copies of the same files. Place them all in a common directory and define the SMSINCLUDE path list such that it contains the directory:

```
SMSINCLUDE="/home/defs/include /home/john/custdefs"
```

Place the above line in one of the `qsms.conf` files: per-project, per-definition or “global” (`$QCHOME/include/qsms.conf`).

The `qcenv` file is now copied from the SMS definition to the project directory

For example, if a custom definition requires environment variable QCASKNUMBER=1 (prompt interviewer for record key), you can place this `qcenv` file in the SMS definition directory (there is no equals sign = in the line):

```
QCASKNUMBER 1
```

The command `qsms` without any arguments recompiles the algorithm

The comments in the generated `sms/signature.c` file keep track of which command line options were used.

Bugfixes

DOS-style line endings in `algorithms` caused compilation errors

This caused problems with editors that hide DOS-style CR/LF (noticeably `vim`). To prevent this, `qsms` now converts DOS-style CR/LF to UNIX-style newlines (LF).

Core dump in `qsms` if long lines in the `variables` file

This happened if a variable had >127 characters in the pre-initialized data, like:
`int weights[60]={1,4,3,0,1,6,7,1,2,` ...etc... `,5,2,3,1,7,3,6,3,2};`

On some systems, `reg` and `tab` could not be used as SMS variable name (34495)

Although they are not reserved words, `reg` and `tab` caused `qsms` to fail when compiling `supervisor.c`.
Now the `#include` sequence has been rearranged to avoid the clashing macro definitions in `<curses.h>`.

Special procedures caused compilation errors in K&R-mode

Missing function declaration (for use in `rtarray[].routine`). This happened when the `tipresps` file defined user-supplied special procedures.

Wrong error message when failing to find an `#include` file

The message implied that the error occurred in the file to be included, not in the including file:
```
"filename.alg", line 16: ERROR 905: Can not find 'filename.alg'
file
```

Sample Management System - SMS

SMS supervisory menu

New feature

New supervisory menu option 722 *Call Result All Dialers*

This report provides a compact overview of the dialing performance for all dialers. Whereas menus 720 *Call Result Report* and 721 *Call Result by Time Slot Report* show nuisance hangups as percentage of dialed calls, this new report shows them as a percentage of answered calls, making it easier to compare with statutory “silent call” regulations.

The report has two subheadings, `Since sign-on to dialer` and `Recent values`:

- Under each subheading, the following values are shown:
  - `Mode` - momentary mode: `Auto` (autodialing), `Pred` (predictive), `Over` (temporarily throttled to autodialing) or `SUSP` (dialing suspended).
  - `Start time` - the time span the values are collected from.
  - `Asked` - the number of records sent to the dialer
  - `Indial` - the number of records currently being dialed
  - `Not dialed` - records not dialed due to out-of-trunks or other bottlenecks
  - `Bad number`, `Busy`, `No answer`, `Answer` - call outcomes;
    the percentages are based on the number of dialed records
  - `Nuis hangup` - fraction of `Answer`'ed calls that became nuisance hangups
- **`Since sign-on to dialer`** subheading:
  Dialing statistics since the project signed on to each dialer.
  - Per-dialer values: Dialing statistics since the server signed on each dialer (menu 701 *Add dialer*).
  - `TOTAL`: Statistics since the server started, for all dialers (even those that no longer are signed on).

  The rightmost column shows:
  - `Intv Wait`: Average wait time for all interviewers since start of the server; number of successful waits (base for calculating the average wait time).

  In multidialer configurations, the rightmost columns shows:
  - `TOTAL`: Wait time statistics since start of server across all dialers (even those no longer signed on).
  - Per-dialer values: Wait times for the most recent `qtip` sessions on each extension. (SMS does not accumulate wait time statistics on a per-dialer basis.)
- **`Recent values`** subheading:

Dialing statistics for the current time slots (last 36-40 minutes).

- Per-dialer values: Dialing statistics for the current time slots (36-40 minutes, see column `Start time`).
- `TOTAL`: Sum of the per-dialer values (even if their durations differ).

The rightmost column shows:

- `Intv Wait`: Average wait time for the `qtip` sessions currently signed on.
  Useful as short-term feedback when tuning the dialer with menus 741-752 (and 780-783),
  but not precise (it aggregates data from sessions of differing age).

In multidialer configurations, the rightmost column shows the `TOTAL` and a break-down per dialer.

Example:

```
Project: j765432
Generated: 1/8/07   11:48 AM by johndoe

                           Since sign-on to dialer

Dialer           Mode| Asked|  Not |  Bad    Busy   No   Answer|  Nuis| |  Intv
 Start time          |Indial|dialed|number         answer       |hangup| |  Wait
---------------------+------+------+-------------------------------+------| |------
qtc1             Pred|   263|     0|     0     55     48    160|     0| |   110
 1/8/07   11:20 AM  |     1|  0.0%|  0.0%  20.9%  18.3%  60.8%|  0.0%| |  13.9
mrdialer2        Pred|     7|     0|     0      0      1      6|     0| |     4
 1/8/07   11:45 AM  |     1|  0.0%|  0.0%   0.0%  14.3%  85.7%|  0.0%| |   8.8
TOTAL                |   270|     0|     0     55     49    166|     0| |   255
 1/8/07   11:20 AM  |     2|  0.0%|  0.0%  20.4%  18.1%  61.5%|  0.0%| |  13.7

                                Recent values

Dialer           Mode| Asked|  Not |  Bad    Busy   No   Answer|  Nuis| |  Intv
 Start time          |Indial|dialed|number         answer       |hangup| |  Wait
---------------------+------+------+-------------------------------+------| |------
qtc1             Pred|   263|     0|     0     55     48    160|     0| |    10
 1/8/07   11:20 AM  |     1|  0.0%|  0.0%  20.9%  18.3%  60.8%|  0.0%| |  11.0
mrdialer2        Pred|     7|     0|     0      0      1      6|     0| |     4
 1/8/07   11:45 AM  |     1|  0.0%|  0.0%   0.0%  14.3%  85.7%|  0.0%| |   8.8
TOTAL                |   270|     0|     0     55     49    166|     0| |    14
 1/8/07   11:20 AM  |     2|  0.0%|  0.0%  20.4%  18.1%  61.5%|  0.0%| |  10.4
Hit <rtn> to continue:
```

Feature change

Supervisory menu option 14 *Change Variable Values* sets dialer variables (`noansw` etc.)

In Quancept 7.6.1, dialer variables like `noansw` were changed via menu option 14 *Change Variable Values*.
In Quancept versions 7.7 through 7.9, menu option 14 only sets the initial value, used by the first sign-on to a dialer.
The current value of a dialer variable is set via menu options 741, 750-752 or 780-783. This distinction has caused some confusion.
In Quancept 7.9.1, if a dialer variable is modified with menu option 14, the change is automatically propagated to all dialers currently signed on, exactly as if the supervisor also had used the corresponding menu option 741, 750-752 or 780-783:

Setting this variable...	...acts as if this menu option was used	
	#	Title
noansw	741	Modify No-Answer Time
predictive	750	Select Dialing Method
predictalg	751	Select Predictive Algorithm
predictadjust	752	Predict Adjust Percentage
minhangups	780	Set Minimum Non-Predictive Hangups Count
maxhangups	781	Set Maximum Non-Predictive Hangups Percentage
minprehangups	782	Set Minimum Predictive Hangups Percentage
maxprehangups	783	Set Maximum Predictive Hangups Percentage

Notes:

- Use menu option 706 *Display dialer specific variables* to see the actual values. The values displayed by menu option 14 may differ, either because a dialer was signed off while changes were made, or because changes have been made with menu options 741, 750-752 or 780-783.
- Changes made with menu option 14, while a dialer is signed off, are not propagated when it is signed back on with menu option 701 *Add dialer*. The dialer resumes its old values, saved in `sms/dialer.sms`.
- The propagation of dialer variables from menu 14 can be turned off by recompiling SMS with feature test macro NO_DIALER_MENU14 in `qsms.conf`:

```
SMSCFLAGS="-DNO_DIALER_MENU14 ...other options..."
```

Bugfixes

Supervisory menu options 9 and 10 did not save variables (4426)

When options 9 *View and Modify Record* and 10 *View/Mod Several Recs* span more than one screen page, record variables that had been modified, on a page other than the one where you typed `w` or `done`, were not saved.

Redirecting input caused *not authorized supervisor* (32646)

When redirecting standard input to `runsms supervisor`, supervisors whose UNIX user names are longer than 8 characters were rejected with the message *Sorry, you are not listed as an authorized supervisor.*

QCWHOAMI environment variable had no effect

The login user name was always used for checking the `$QCHOME/sms/permissions` file. It is now possible to instead set QCWHOAMI (like with `qtip`).

Exceeding maximum line length in *permissions* file (2613)

The limit for `permissions` entries has been increased to 1022 characters. Previously, the limit was 253 characters; when a project-specific `permissions` file and the `$QCHOME/include/permissions` together had an entry longer than 256 characters, `runsms supervisor` could overwrite itself (core dump).

Other changes in SMS

New feature

New function `tipcodeText` to retrieve a tipcode's display text

Synopsis:

```
#include "tipcodetext.alg"
char *ptr;
ptr = tipcodeText(value)
```

This function returns the display text associated with a tipcode in the `tipresps` file. For instance, if *`value`*=4 (`tiprna`), the function call `tipcodeText(`*`value`*`)` returns a pointer to the read-only string `"Resp not available"`.

Bugfixes

SMS variables with ≥12,000 bytes were truncated (16777)

This occurred when restarting a stopped SMS server. Now there is no limit, except by the amount of available memory.

When loading multiple quota files, `quota_read_cell()` would return the wrong cell number.

If a custom report loaded multiple quota files into memory by calls to `quota_read_array()`, subsequent calls to `quota_read_cell()` returned the cell number corresponding to the dimensions and axes of the last loaded file, not to the dimensions of the requested file.

Autodialer specific

See also section *New commands* about `qtsxfer` and `qtsextn`.

Feature change

VoIP support for remote interviewers and supervisors

All variants of the `remote` program (`rqtip`, `rsupermen` etc.) now accept a SIP URI as "phone number":

```
rqtip [-D dialer] sip:user@hostname:port projname
```

Typical SIP URIs could be `sip:192.168.12.34:5060` (local area network) or `sip:+15557654321@example.net` (via a SIP provider). The optional *`dialer`* parameter specifies which dialer initialization file to use (default: `qts-sms.ini`).

Example (Linux only):

- Call center with Linux-based Quancept server.
- Booths with Windows PCs with sound card and SIP software (softphone).
- Interviewers use `telnet` to login to the Quancept server.
- Use this command to establish the VoIP connection to the softphone:

```
rqtip sip:$REMOTEHOST projname
```

This works because on Linux, the telnet daemon (`in.telnetd`) sets environment variable REMOTEHOST to the IP address of the connected telnet client (same address as the softphone).

VoIP support requires mrDialer software version 4.0 or higher.

Leading plus + in international telephone numbers

In `qts-sms`, `remote`, and `qtsclean`, telephone numbers can now be formatted as plus (+) followed by the country code, area code (if applicable) and subscriber number. This allows multidialer projects with dialers in different countries to use a common sample file. This feature requires dialer software versions mrDialer 3.0, QTC 1.6.30, or higher.

New options in `qtsclean`:

Option	**Description**
`-r` *`trunkgrp`*	Trunk group where the numbers should be dialed, default: 0
`-i` *`callerID`*	Originating telephone number, or `blocked` to suppress caller ID
`-w` *`filename`*	Name of a sound file to be played before producing a nuisance hangup

In multi-dialer configurations, `accounts.sms` identifies the dialer

For calls dispositioned by the dialer (call failures, no-answers), `accounts.sms` always reported the user name `qts`; it wasn't possible to tell which dialer was used. `qts-sms` now reports its dialer information file name as the user name. You can override this by specifying the user name in the dialer initialization file (`qts-sms.ini`):

```
[main fixed]
        dialerusername=qts
```

Bugfixes

The `remote` program could not record from remote extensions

In Quancept 7.9, the `remote` program (also known as `aqmonitor`, `aqtip`) used the wrong extension for recording (950 instead of 900). This produced an error when attempting to call the remote supervisor (phone 1234567) and record the entire monitoring session:

```
$ raqmonitor 1234567 myfile.wav
aqmonitor: Extension not busy
```

Wrong time calculation when evaluating the dialer's `call.log` (34476)

The scripts for processing the dialer's `call.log` files computed the fractions of a second incorrectly. This affects `busytime.awk`, `callhist.awk`, `callrep.awk` and `dialtime.awk`.

On SCO UNIX, `dialtime.awk` reported *Illegal reference to array qsamp* (45146)

A naming conflict with a built-in symbol has been resolved.

`qts-sms` left 12 unread bytes in dialer message queue (35794)

The command `ipcs -q` would sometimes show one message (12 bytes) in the dialer queue. This happened with dialer software versions older than QTC 1.6.20 because the dialer interface `qts-sms` attempted to send an error message to itself.

On the dialer, project groups could survive an SMS crash

Sometimes the extension monitor `emon` still showed a project after its SMS server had crashed. This happened when `qts-sms` failed to detect the crash. The detection has been improved, in particular to cope with Linux idiosyncrasies (wrong `errno`).

A full message queue could cause `qts-sms` to hang.

In particular, a stuck (looping) SMS server could start a domino effect, causing all other SMS servers to hang. To remedy this, `qts-sms` now applies timers when writing to queues. The timeouts are configurable in the dialer initialization file (`qts-sms.ini`), section `[main fixed]`:

Parameter	Used when writing to	Action upon timer expiry
`timeoutcampaign=30`	SMS campaign queue	The jammed message is discarded, but `qts-sms` continues.
`timeoutdialerqueue=10`	Dialer queue	`qts-sms` shuts itself down (irrecoverable error, should never occur)

In spite of this change, a “disk full” error will almost inevitably cause all active SMS projects to jam.

The *qcset*-based utilities

Enhancements

Ability to select PTF translations in `qcset`-based utilities (24591, 31762)

For `ptftran`-based multi-lingual projects (`'}0English}1Deutsch}2Français'`), Quancept 7.9 added the ability to select a translation via its `setlang` number. In the previous example, French has number 2 (`}2`), meaning you can generate output in French with the command `qcset -p 2`.

Quancept 7.9.1 adds the ability to use translations from Qolyglot, Quanquest, and so on, using the three-letter language code from the PTF-file. For example `fra` for French: `qcset -p fra`. Some texts are not stored in the PTF file, (for example arguments to `set` and `callfunc`). Therefore you receive better results with `-p 2` than with `-p fra`. On the other hand, the latter expands embedded iteration variables:

Construct	Expanded with	
	-p 2	**-p fra**
`set color = '}0red}1rot}1rouge'`	Yes	No
`callfunc( 'append', varname, '}0car}1Auto}2automobile' )`		
`display` or `protect '}0Speaking of }1Bezüglich }2A propos ' + topic`	Yes	
`ask '}0Which model?}1Welches Model?}2Quel model?'`		
`for iter = '}0bigger}1größeres}2plus grand' /` ... *etc.* ...		

`ask '}0A }1Ein }2Une voiture ' + iter + ' }0car?}1 Auto?}2?'`		
`ask '}0A [+iter+] car?}1Ein [+iter+] Auto?}2Une voiture [+iter+]?'`	No	Yes
`resp` ... *etc.* ... `other dk ref null`		

The option `qcset -p` *lng* can also translate predefined responses like *Other (specify)* and *Refused*. To this end, create a `fields-`*lng*`.def` file with translations and place it in `$QCHOME/include` or in the project directory. If you do not need to distinguish between national variants of a language (such as `eng` British English and `enu` American English), you can shorten the language code to two letters `fields-`*ln*`.def`. The search for files follows this order:

#	Field definition file	Comments
1	`fields-`*lng*`.def`	Project specific, three-letter language code
2	`fields-`*ln*`.def`	Project specific, first two letters of language code
3	`fields.def`	Project specific, any language
4	`$QCHOME/include/fields-`*lng*`.def`	Global, three-letter language code
5	`$QCHOME/include/fields-`*ln*`.def`	Global, first two letters of language code
6	`$QCHOME/include/fields.def`	Global, can be customized (default: English)
7	Use built-in English field definitions	Last resort if no field definitions file can be found

Ability to print long response lists with `qcprt` (3416)

Previously, `qcprt` was unable to print looped categorical questions with:

- more than 99 cards per question. This limit now is 1000.
- more than 292 iterations. This limit has been increased to 1000.
- more than 297 responses. This limit now is 1000 (in practice, 729).
- more than 6144 characters in a row. This limit has been eliminated.

Now the only real limitation is that each row of output must fit on one page, otherwise you will see:

```
Item 'Q27B      ...' is too big for page size 80x66.
```

To increase the page size for all projects, edit the print configuration file:

```
$QCHOME/include/default.qpf
```

To increase the page size of a single project, place a configuration file in the project directory:

```
cp $QCHOME/include/default.qpf projname.qpf
```

In either configuration file, replace the parameter `MaxLines=66` with a larger value:

```
MaxLines=10000
```

Bugfixes

Incorrect iteration numbering in sublists (4493, 27450, 36772)
: For questions in a loop of type *`sublist`* `in` *`masterlist`*, the `qcset`-based programs (`qqpunch`) numbered the iterations according to the positions in the sublist. This has been corrected to use the positions in the masterlist.

Loops with negative `step` were ignored
: The `qcset`-based programs (`qcqq`) treated loops of the type `for i=5 to 1 step -1` as having 0 iterations.

Supervisory program - *supermen*

New feature

Large screen support
: `supermen` can now utilize screen sizes larger than 80×25. This is primarily useful when viewing large reports from option 13 *Interviewer statistics*. The screen size can be specified via the `termcap` file or via the standard UNIX environment variables `COLUMNS` and `LINES`.
In windows that can be resized (`xterm`, or `ssh` clients like PuTTY), `supermen` automatically adjusts to the momentary page size. However, if you resize the window during a multi-page output, the adjustment may not take effect until the next output is started.

Option 13 *Interviewer statistics* - new functions `mmss` and `hhmmss`
: In `.smr` report scripts, the formatting of durations (seconds) into hours, minutes and seconds has been simplified with two new functions `hhmmss` and `mmss`. For example, with an argument `t` = 7355 seconds, `hhmmss(t)` returns "`2:02:35`" and `mmss(t)` returns "`122:35`". Example:

```
Time spent in actual interview body (excl. SMS screen)
# Count number of interviews with a non-zero duration of body
filter             durint!=durtel
var intvcnt        true
var intvtime       durint-durtel
# Display the sum as HH:MM:SS and the average duration as MM:SS
col 'Nbr Intv'     intvcnt
col 'Sum Intv'     hhmmss(intvtime)
col 'Avg Intv'     mmss(intvtime/intvcnt)
sum 'Grand Total'
```

This script produces an output similar to:

```
                 Nbr Intv   Sum Intv   Avg Intv

ahmed                  14    2:42:33      11:36
cristian                8    1:29:34      11:11
dorothee                1      13:13      13:13
kazumi                 27    5:02:28      11:12
Grand Total            50    9:27:48      11:21
```

Enhancements

Option 2 *Monitor interviewer* and option 3 *Realtime display* - long lists

When there were many list files, or a list file contained many entries, the display would scroll off the top of the screen. The output is now shown page-wise and sorted in alphabetical order.

Option 4 *Signal interviewer* - long lists

When many interviewers are working, the selection list would sometimes scroll off the top of the screen. The names are now displayed page-wise in alphabetical order.
Also, a flaw has been corrected whereby the signal could be delivered to the wrong interviewer if the selected interviewer already had signed off before the signal was sent.

Option 9 *SMS menu* - recovery from hung SMS server (19014)

When many `qtip`s were blocked due to a hanging SMS server, `supermen` option 9 *SMS menu* would not always wait long enough to give all `qtip`s time to return a sample before starting SMS recovery. The non-returned samples would remain stuck in the `inuse` or `indial` queue. `supermen` now sets the message queue to read-only, forcing the jammed `qtip`s to write `SALVAGE.SMS` immediately.

Option 9 *SMS menu* - intermediate shell

`supermen` no longer uses the shell (`sh -c`) to start the SMS server (or other programs except `$EDITOR`, `$QCPRINTCMD`). This avoids problems with custom SMS algorithms that use `ps` to verify that no other instances of the SMS server are running. This verification would fail because `ps` would always show `sh -c runsms` *`projname`*.

Option 11 *Topline tabulation* - multi-lingual PTF-files

In multi-lingual projects, `supermen` now always asks which language to print topline tabulation in. In Quancept 7.9 it would only ask for `ptftran`-based projects (`'}0English}1Español'`), not for projects translated with Qolyglot or other means.

Bugfixes

Infinite loop upon loss of telnet connection

`supermen` did not exit when its telnet connection was closed (end-of-file on standard input). This could cause `supermen` to loop if started from a login script that traps SIGHUP (the hang-up signal).

Error *user not in /ip/supperms* if input is redirected (32646)

If standard input has been redirected to a file, the login name was truncated to 8 characters, causing errors for supervisors with longer names.

Option 2 *Monitor interviewer* and option 3 *Realtime display* failed with long lists (33956)

`qmonitor` and `userscan` did not accept more than 31 user names in a list. The limit has been increased to 1024 (some operating systems report E2BIG *Arg list too long* before reaching this limit).

Option 5 *Shared memory operations* - replacing segments on-the-fly

Replacing a shared memory segment while interviewing is in progress could cause data corruption in `qtip`. The procedure is now safe, provided the script does not have more than MAXSHMSEG-1 `resp dbase` questions. MAXSHMSEG is an operating system configuration parameter, usually 16 or more.

Option 10 *Display quotas* - multiple lock strategies

When using option 1 *Select a project* to switch between projects belonging to different `$QCHOME`s, configured with different lock strategies (`$QCHOME/include/.qclock`), `supermen` option 10 would continue to use the original lock strategy.

Clarification

How to translate option letters in one-line menus in `supermen`

One-line menus are similar to the following, where the option letters are `a`, `d` and `c`:

```
0059 00 \nA)dd D)elete C)lear 'q'=quit:
```

In the German version, the same options are invoked with the option letters `n`, `e` and `z`:

```
0059 00 \nN)eu E)ntfernen Z)urücksetzen 'q'=quit:
```

The following information was omitted from the original documentation:

- The option letters do not need to be identical to the English option letters.
- Each option letter must be followed by `)`.
- The options must be in the same order as in `$QCHOME/english/qcsmen.lng`.
- None of the options can be the *quit* character, defined in `qcsmen.lng`:
  `0192 00 q`

Coding tools

Feature changes

Import of coding projects from other system architectures

You could not previously move coding projects between “little endian” and “big endian” architectures. The coding tools now automatically recognize and convert the foreign files. This affects `precode`, `ldascode`, `quac`, `vquac` and `decode`.

	Big endian		Little endian	
Hardware architecture	SUN SPARC	HP PA-RISC	Intel x86	
Operating system	SUN Solaris	HP HP-UX	GNU Linux	SCO OpenServer

Changes in *quac*

A new coding option `-u` suppresses conversion to upper case (4450)

This allows entry of lower-case letters in open-ended codes and when editing texts.
Option `-u` can be used with the commands `code` (for the `*n` action), `newcode` and `text`.

Changes in *vquac*

The *Filter* function gave unexpected results for non-ASCII characters

When splitting text into words and folding them to upper-case for case independent comparison, `vquac` now uses the character map for the locale specified by the environment variable LANG. Previously `vquac` used the character map for DOS codepage 437 (IBM® original PC). It also folded accented lower-case characters into their non-accented upper-case equivalent, such that éèêëeE were all considered the same letter (but different from ÉÈÊË). For backward compatibility, this behavior is preserved if you `unset LANG` before starting `vquac`, except that the obsolescent codepage 437 has been replaced with codepage 850 (IBM multilingual PC) which is more commonly used (noticeably in Windows™ telnet and older versions of SCO OpenServer).

It is no longer necessary to set the *VVTERMCAP* environment variable

If the environment variable VVTERMCAP is not set, `vquac` searches for one of these files:

- `/etc/vvtermcap` (like in previous Quancept versions),
- `$QCHOME/lib/vvtermcap` (new in Quancept 7.9.1).

If you customize the VVTERMCAP file, we recommend you to install it as `/etc/vvtermcap`.

Bugfixes

Bugfix in *decode*

In Quancept 7.9, `decode` did not write the card number to the `.acd` file (33942)

The card number was omitted from the amalgated coding file.
As a consequence, `qccon` failed to merge the open-ended data with the precoded data.

Bugfixes in *quac*

Non-ASCII letters like *á* or *ñ* were treated as non-letters

The editing command `text` converted the word `erróneo` incorrectly to upper case: `ERRóNEO`.
The word matching commands `kwic` and `wc` considered `erróneo` to be two words: `err` and `neo`.
The text is now processed according to the character set specified by the environment variable `LANG`.

Command `code -v` caused program corruption (2802, 2892, 4438, 4443)

The program wrote data outside the allocated buffer, causing corruption and possible core dumps.

Command *Code* showed incorrect `EDITED` status

When the `.dtx` file had changed, the program showed `DTXCHG EDITED`.
When the text was edited, the program showed nothing.

Command `print` gave error message *Code no longer exists* (33603)

This occurred with OTHER codes larger than (maximum code) − (number of precodes).

Command `print -k >`*`file`* did not redirect output to *`file`* (2585)

The program interpreted *`file`* as a keyword instead of a file name.

Command `print` option `-w` never ceased its effect

After using `print -w` (print without codes), subsequent `print` commands would also print without codes.

Command `deltext` did not ask for confirmation

Before deleting a text with codes assigned, `quac` should ask *Do you really want to delete this text?* This only worked for certain code values.

Command `text` did not work on *longtext* variables (3428, 3429)

It would edit (and corrupt) the *longtext* reference rather than the actual data content.
Note: It is not possible to use keywords on *longtext* (commands `wc`, `kwic` and option `-k`).

Linux only: Command `wc` corrupted the heap, causing core dump at some later stage

A file stream was closed (and deallocated) twice, corrupting the free list.

Bugfixes in *vquac*

The *Code* options did not show data serial numbers >99999 (37029)
: This affected projects using more than five columns for the data serial number (`DATAMAP SERIALCOLS`).

Some *Code* options showed incorrect `DTXCHG EDITED` status
: When scrolling through multiple records, some *Code* options did not update these fields.

Option *Code Frame Modify* showed wrong text (33935)
: Instead of displaying the OTHER text to be modified, `vquac` showed one of the precoded texts, or another OTHER text. If all OTHER codes were 32 or higher, `vquac` would crash.

Precodes longer than 72 characters could cause data corruption
: The corruption occurred when displaying the long precode in *Code Frame*

Option *Filter* showed wrong code numbers for OTHER questions
: Filters on precoded responses were missing the leading `C`; filters on open-end responses showed a wrong code value (off by the number of precodes).

Clarification

For `vquac`, the following table shows some working `VVTERM` terminal type emulations.

VVTERM	**Termcap entry**		**Example**	**What to type**			
TERM	**k8**	**k9**	**when to use**	**Toggle**	**Accept**	**Cancel**	**Quit**
vt100	ESC 9	ESC 0	Telnet in Windows 2000	ESC 9	ESC 0	ESC ESC	ESC a q
vt220	ESC [20~	ESC [21~	Telnet in Windows XP	F9	F10	F1 F1	F1 a q
ansi	ESC [U	ESC [V	BSD/SCO emulations	F9	F10	ESC ESC	ESC a q
wy50	SOH H CR	SOH I CR	Wyse *x*0 terminal	F9	F10	ESC ESC	ESC a q
Built-in emulation			DOS version of `vquac`	F9	F10	ESC	ALT+Q

The `vvtermcap` file is in Xenix format; you cannot cut-and-paste standard UNIX `termcap` entries into it.

Other utilities

Bugfixes

`lg_trans` crashed or terminated prematurely
: Premature termination *Files are mismatched* in the following cases:
  - When the `english` files contained comments with version information (all distributed `.lng`-files do).
  - When errors (typos) in the `english` file have been corrected.

: Crash (core dump) if comments `##C` were missing from the translated file:
  - When modifying translations in the distributed `german` texts.
  - When custom translations have been updated by `qcinitial`.

Environment variable `$QCHOME` was ignored; you had to change directory to `$QCHOME` before running the program.
The lock file `english/qc`*`file`*`.lck` was not removed if the user escaped out of the program (control-C).

`makeqtip` failed on SCO UNIX with compilation errors
Incorrect values of the environment variables `CFLAGS` and `STDLIBS`.

`qditum` generated erratic Quantum scripts
The `base-.qin` file was missing. It has been added for compatibility with `qctum`.
The `uniqid=` keyword was preceded by a (redundant) percent sign `%uniqid=`.
Some single column ranges were printed as `(5,5)` instead of the correct `(5)`.
The code generated for `mp mcodscheme` included `other` when it should not have.

`qmonitor`'s *MESSAGE FROM SUPERVISOR* was always preceded by the letter `a`
Option > (send message to interviewer) should have sent a header (`qcsmen.lng` text 206) with `"\a"` (audible bell) but instead sent `"a"` (lower case letter A).

`shm` and `rmshm` printed wrong statistics
In the messages `Added` or `Removed` *`N`* `databases (`*`X`*` records, `*`Y`*` bytes),` *X* and *Y* were swapped.

`ch03` sample script was missin' the letter `g` (42483)
Throughout the example `$QCHOME/qctest/ch03`, the letter `g` had been deleted (e.g., `eatin`, `reek`).

Clarifications

Do not use `quota` option 8 *Change current values* while interviewing is in progress
This `quota` option is not safe. It overwrites (discards) updates done by `qtip` or SMS during the editing process. Instead, use `supermen` option 10 *Display quotas* suboption 3 *Edit quotas*. It keeps track of concurrent updates by interviewers and merges the results together.

`shm` and `rmshm` accept multiple arguments
This is useful for loading and removing all database segments in a directory in QCSHMPATH:

Command	Action
`shm *`	Load all database files into shared memory. Segment ID files (*`database`*`.shm`) are silently ignored.
`rmshm *`	Remove all databases segments from shared memory. The suffix `.shm` is silently removed; duplicate arguments are ignored.

Support Tools

The tools described in this section are used by SPSS Support.

`makeconvtq` and `convtq` - convert SMS queue files to plain text and vice versa
This shell script generates and compiles a "C" program to import and export queue files (`.tq`). This can be used to:

- recover corrupted `.tq`-files (example: after disk full).
- move projects between little-endian and big-endian achitectures.

SUN SPARC and HP PA-RISC are big-endian, Intel (Linux and SCO UNIX) are little-endian.

The `convtq` script automatically recompiles the program when the project has changed; `makeconvtq` forces a recompilation.

The most useful options are:

Command	Description
`convtq -e *.tq`	Export all `.tq` queue files to `.ttq` text files.
`convtq *.ttq`	Import all `.ttq` text files as `.tq` queue files.

The export stops cleanly with an error message if it detects corruption. If it is more important to get the project up and running again quickly, rather than spending time trying to save the corrupted data, you can simply re-import the exported `.ttq`-files.

`qtipcore` - analyze a `qtip` core dump

This program can be used to find out what `qtip` was doing when it crashed. This can be useful:

- after `qtip` crashed because of a bug, for instance in a custom `callfunc`
- if a script error caused an endless loop and you used `kill -SEGV` to create a core dump.

Usage: `qtipcore` *`corefile`* `[`*`projname`*`.qoc]`

The program searches the core file for `qtip`'s event log. The event log is a cyclic buffer that remembers the last 1024 events; the `CURRENT` entry indicates the last event logged. Events are keystrokes typed, or `.qoc` statements executed or passed *en route* to the target of a snap. The program does a coarse disassembly of the `.qoc` statements, not enough to reconstruct the script, but sufficient to tell where in the script `qtip` might be.

Type	Example of output	Description
Key	`K  '2'  32`	What the interviewer typed
Do	`D     7747   set arithmetic`	`.qoc` statement to be executed
Walk	`W     4085   callfunc 'setcols'`	`.qoc` statement passed (snap, restart)
#	`<==== CURRENT <====`	Last event logged into cyclic buffer

In this example the core dump was caused by a custom callfunc:

```
D     3239   label Q34
D     3246   ask
K  '1'  31
K CR    0a
D     3274   label Q35
D     3281   ask
K  '1'  31
K CR    0a
D     3367   unset
D     3372   label QDUMAGE
D     3379   dummyask
D     3431   unset
D     3436   set = nbit()
D     3446   callfunc 'local'      <==== CURRENT <====
W     1840   route
W     1855   set arithmetic
```

```
    W     1865   label _Y0004
    W     1872   continue
    W     1874   set = '...'
    W     1884   callfunc 'getsmvar'
```

Copyright © 2008 SPSS Ltd., UK. All rights reserved.