CCP4 Update

Update Setup and Preparation Procedure

Eugene Krissinel, August-September 2012

Contents

Abbreviations

UD update directory per OS/Release/Architecture (Section 1)
GUD 'general' update directory per OS (Section 1)
UDF update description file (Section 3)
UID update ID (Section 3)
ITD incremental template directory (Section 3.9)
TTD target template directory (Section 3.9)
RTD reference template directory (Section 3.9)

1. Server-side configuration

CCP4 Update Client (CCP4 Package Manager, PM) hard-codes OS-specific URLs, which are used to access server-side update structures. The URLs are defined in PM's source file *setup_defs.h*, and currently are as follows:

OS	Base Update URL
Mac OSX	http://www.ccp4.ac.uk/setup/macosx/updates/
Linux (32 and 64-bit)	http://www.ccp4.ac.uk/setup/linux/updates/
Windows	http://www.ccp4.ac.uk/setup/windows/updates/

If order to change these URLs, PM needs to be recompiled.

These URLs correspond to directories in the server's file system (general update directories, GUDs) that contain file *task.xml* and directories with update data (update directories, UDs). Because of having separate 32- and 64-bit builds for Linux, Linux's GUD contains architecture-specific UDs:

GUD for Mac OSX and Windows:

```
-rw-r--r--    1 ekr      dlccp04a     2254 Aug 29 12:23 task.xml
drwxr-xr-x    2 ekr      dlccp04a     4096 Aug 25 18:48 6.3.0
```

GUD for Linux:

```
-rw-r--r--    1 ekr      dlccp04a     2253 Aug 29 12:28 task.xml
drwxr-xr-x    2 ekr      dlccp04a     4096 Aug 25 18:57 6.3.0-x86_64
drwxr-xr-x    2 ekr      dlccp04a     4096 Aug 25 18:53 6.3.0-x86
```

Any given GUD may contain as many UDs for different releases as necessary. PM identifies CCP4 version on the host system and looks for the corresponding GUD/UD combination automatically.

In GUD, file *task.xml* contains PM's layout and program of actions (see Appendix 1). This file may need modification only if there are changes in PM. UD names are taken to be CCP4 release numbers, with architecture suffixes (-x86 or -x86_64) for Linuxes. This is hardcoded in PM, see constructor body in *setup_window.cpp*:

```
  version   = "6.3.0"; // needs to be identifiable from the host
  updateDir = version; // update directory name on server
#ifdef Q_OS_LINUX
#ifdef __BUILD_32BIT
  updateDir.append("-x86");
#else
  updateDir.append("-x86_64");
#endif
#endif
```

A typical UD has the following content:

```
-rw-r--r--    1 ekr      dlccp04a 12159453 Aug 25 18:48 6.3.0-001.pck
-rw-r--r--    1 ekr      dlccp04a      286 Aug 25 18:48 6.3.0-002.pck
-rw-r--r--    1 ekr      dlccp04a   304753 Aug 25 18:48 6.3.0-003.pck
-rw-r--r--    1 ekr      dlccp04a   223449 Aug 25 18:48 6.3.0-004.pck
-rw-r--r--    1 ekr      dlccp04a     2539 Aug 25 18:48 updates.xml
```

File *updates.xml* contains update descriptions, which are used by PM for presentation, location and download of updates. The file name, as well as extensions **.pck*, are fixed in PM's code, file *update_data.h*. Files **.pck* represent compressed update packages. Their names are made from Update ID (UID) and extension *.pck*. UIDs are chosen at time of update preparation, see Section 2.

All UDs are created and maintained using *mkupdate* utility (see Section 2). *mkupdate* is used also to generate update packages and add them to the UD. Files in UD are not supposed to be manually curated.

2. Preparation of updates

All maintenance of UDs is done using *mkupdate* utility, no manual intervening with UD's files is expected, except for copying UDs between various locations on the server.

2.1 Initialization of Update Directory

```
mkupdate –init UD GUD
```

This command creates a new UD (e.g., '*6.3.0*') in location given by 3rd parameter. Although *mkupdate* does not enforce any particular UD name, and will work with any name given, UD names must be chosen as described in Section 1.

UD directory needs to be initialized only once per release and platform/architecture. Note that initialization only causes placement of *updates.xml* file into the directory. If UD is re-initialized, this procedure may leave unwanted pre-existing files in it, which must be removed manually.

2.2 Addition of updates

After UD is initialized, one may add updates to it:

```
mkupdate [-f] update.dsc UD
```

File *update.dsc* is called Update Description File (UDF), which contains a plain-text description of update. UDF is described in details in Section 3.

UDF contains one particular parameter, Update ID (UID), which must be chosen incrementally. *mkupdate* does not allow one to remove or substitute existing updates except the last one (with highest UID). This is done intentionally in order to forbid modification of updates that are already released. For replacing the last update, force the command by providing the optional '`-f`' parameter.

As a result of the above command, *mkupdate* modifies *UD/updates.xml* file, and creates update package *UD/UID.pck*, which contains instructions from *update.dsc* and all files that need to be shipped to the client setup.

It is advisable to make updates in a dedicated location, different from the production GUD directory (in a work GUD, WGUD). Once modifications are done, WUD's content may be copied or moved to GUD/UD for release.

2.3 Applying updates

In order to check the actual content of update package, and to imitate the process of applying updates on client setups, the following command may be used:

```
mkupdate -apply UID UD /.../local_ccp4_root
```

where the last parameter specifies where the content of *UD/UID.pck* should be unpacked (this corresponds to $CCP4 in client setups).

It is possible to apply all updates in one go:

```
mkupdate -apply-all UD /.../local_ccp4_root
```

which may be useful for the generation of cumulative patching archives.

3. Update Description File

Update description file (UDF) contains update specification (Update ID, date, blocking status) and list of actions (files to change/add/remove, messages) to perform. UDF template may be obtained by running

```
mkupdate -desc-template > template_udf.dsc
```

which will contain explanatory notes to all UDF items.

UDF items contain key-worded sentences and data blocks. Keywords start with exclamation mark "!". All subsequent white spaces are disregarded (unless in file names),

while line endings are significant. Consider UDF items in order of their appearance in the file.

3.1 Update ID

```
#    UPDATE_ID is a mandatory update ID, which is also used as a file
#  name of update bundle. The only requirement is that update IDs
#  are strictly incremental. Use of given example is encouraged.

!UPDATE_ID:  6.3.0-001
```

UID must satisfy the following requirements:

- be acceptable as a file name in all OSes
- not to contain white spaces
- allow for incremental modifications

It is advisable to form UID from two parts: symbolic ID and a (zero-padded) increment. This form allows for the automatic generation of UIDs in automated UDF preparation (described in 3.9).

3.2 Update date

```
#    UPDATE_DATE is a mandatory field with the date of update
#  release.

!UPDATE_DATE:  31.08.12
```

Update date may be in any format and is used only for presentation purposes.

3.3 Blocking status

```
#    BLOCKING is an optional field, containing either 'yes' or 'no'.
#  The updater will not show updates with IDs higher than that of
#  uninstalled blocking update(s). By default, updates are not
#  blocking.

!BLOCKING:  NO
```

Most of regular updates are non-blocking. This means, all of them may be installed in one go. However, in certain situations, you may want to insist that update procedure stops at a certain update, and may be resumed only after updater is re-started. At time of writing, there are two situations where blocking updates may be needed:

a) initial integration of updater with ccp4i in release 6.3.0
b) updating the updater, when the existing updater is not capable with the installation of further updates

It should be considered a good practice that all updates to the updater are blocking.

3.4 Update description

```
#    UPDATE_DESCRIPTION is a mandatory section containing free text
#  describing the update. The text will appear in Package Manager
#  when updates are listed for the user.

!UPDATE_DESCRIPTION
Latest features and bug fixes to:{br}
{b}Refmac{/b} - jelly-body refinement{br}
{b}Aimless{/b} - space-group determination
{b}PISA{/b} - surface area calculations.
!END
```

Update description is shown in PM on the right from the list of updates, when users select update in the list. It is a good idea to provide a brief and concise description of all updates. Note that html markup may be used in the description, however, remember to use curly brackets for html tags as shown in the above example.

3.5 Source base

```
#    SOURCE_BASE is an optional path prefix for all source files and
#  directories listed in UPDATE_COPY section, unless they start with
#  slash.

!SOURCE_BASE: /home/eugene/cmp/ccp4-6.3.0
```

Using source base allows one to improve readability of UDFs. It is a good idea to keep source files (i.e. those that reflect the latest state of ccp4 setup) in the same hierarchy as they would have in a real setup. Then, having defined source base as a path to that hierarchy (equivalent to $CCP4), target files in the Update File List (*cf* 3.6) may be denoted by mere asterisks.

3.6 Update file list

Update file list specifies which files from local file system (where the updates are being compiled and prepared, or source files) should be put in the update package *UID.pck,* and where they should be placed in the client setup (target files).

Update file list provides a wealth of options of how the source and target files may be specified. One can use:

- absolute file path specification
- environment variable based specification
- source-base based specification
- directory-level specification

```
#    UPDATE_COPY is an optional list of files and/or directories
#  to be copied into client installation. Each line must start with
#  a path relative to the installation root, i.e. $CCP4, followed by
#  separator '<' and full path to the file (or directory) on local
#  system, from where it will be packed into the update bundle. Files
#  and directories are identified automatically, no slash-endings are
#  required. If files or directories are to be placed at $CCP4 root,
#  the recipient file/directory name must be left empty. Local path
#  may be specified through an environmental variable as shown
#  (always in hex). If SOURCE_BASE is defined, it will be prepended
#  to each source file name, unless the latter starts with slash.
#  If the destination file name is given by asterisk, the asterisk
#  will be replaced by the source path before SOURCE_BASE is
#  prepended. Note that asterisks may be used only for file names,
#  not for directories.
#    File/directory names may have permissions attached through a
#  colon as shown below. In case of directories, permissions will
#  be applied to all files in that directory. If permissions are
#  not specified, they will be kept as for the substituted file, or
#  guessed, if the file/directory is newly added.

!UPDATE_COPY
bin/refmac:5555  < /local-path/bin/refmac
bin/aimless:5555 < $CCP4_LOCAL/bin/aimless
bin/pisa:5555    < bin/pisa
*:6644           < bin/watertidy
                 < /local-path/root_patch_1/
:5555            < /local-path/root_patch_2/
!END
```

In each statement of the list, left-hand side represents the target path (relative to $CCP4 on the client), "<" is separation symbol, and right-hand side represents the source path (from where the file or directory will be copied into the update package). The following general rules apply:

a) if source path refers to a directory, target path is considered to be a directory as well.
b) all target paths are relative to $CCP4 on client system
c) if source path starts with slash, it is interpreted as absolute path
d) if source path starts with "$", leading word up to the first slash is considered to be the name of an environmental variable, which value is then substituted
e) if source path does not start with slash or "$", it is considered as a tail to the path given by the source base
f) empty target path refers to $CCP4 directory on the client and may be used only if the corresponding source path refers to a directory
g) target "*" means same path as the one given in the source part, less of source base header
h) target files and directories may contain file permissions (hexagonal), appended through a colon (no spaces)

Beware that PM may create all needed paths on the client, therefore, using "*" target with absolute path will place file outside client's $CCP4 and may potentially harm the system. The following constructs are illegal:

```
# NEVER DO THIS!
!UPDATE_COPY
*    < /local-path/bin/refmac
*     < $CCP4_LOCAL/bin/refmac
      < /local-path/bin/refmac
      < bin/refmac
      < $CCP4_LOCAL/bin/refmac
bin/  < /local-path/bin/refmac
bin/* < /local-path/bin/refmac
!END
```

It is advisable to always specify file permissions for all targets explicitly.

3.7 Remove file list

```
#    UPDATE_REMOVE is an optional list of files and/or directories
#  to be removed on the client. All paths must be relative to the
#  installation root, e.g. $CCP4. Do not schedule for removal any of
#  the files mentioned in UPDATE_COPY section.

!UPDATE_REMOVE
share/xxx/file1.dat
lib/xxx/file2.o
!END
```

If files or directories need to be deleted in client setup, they may be listed in the remove file list as shown above, with paths relative to client's $CCP4. PM will delete non-empty directories as well, therefore be careful.

3.8 Update message

```
#    UPDATE_MESSAGE is an optional message to be displayed in a
#  popup window after the update.

!UPDATE_MESSAGE
Check this and that and {b}restart ccp4i{/b}
!END
```

Update message, if given, will be displayed in popup window after the update is installed. It should be used to notify users of specific actions that need to be performed after the update, e.g., to restart ccp4i. Html markup may be used with curly brackets for html tags.

3.9 Auto-generation of UDF

Most of update description file content can be automatically generated using template CCP4 setups. Two types of template setups may be used.

1) Incremental cumulative updates.

This is a CCP4 directory, which contains subset of files changed after the release. For example, if only *$CCP4/bin/refmac5* has changed, then the incremental template directory (ITD) contains directory *bin/* with only one file *refmac5* in it. As more files need to be changed, they are added to the template directory without removing the previous files that have been already released. When changes mature for the next update, UDF is generated with the following command:

```
mkupdate -suggest-inc ITD UD tmpDir
```

Here, *tmpDir* is scratch directory which should allow to accommodate all content of ITD. This command will create *UID.dsc* in current directory, where the following UDF fields are auto-completed:

- update ID (UID), auto-incremented from the highest ID in UD; new UID is used to form UDF name *UID.dsc*
- update date is set to the current one
- blocking status set to unblocking
- source base set to ITD
- update file list as a difference between the content of ITD and cumulative effect of updates currently in UD. When calculating the difference, both file presence and content are taken into account (timestamps disregarded)
- remove file list as a difference between updates in UD and the content of ITD, where only the file presence is counted. Note that this will spot only files removed from updates, and miss files removed from the original release.

The following UDF fields need to filled in manually before update may be released:

- update description
- update message.

Note that the generated update and remove file list needs to be manually inspected and possibly revised. It is Ok to leave out files from these lists for the current update: they will be brought in automatically again for the next update.

2) Comparative updates.

This is represented by a pair of full CCP4 directories, reference and target (RTD and TTD) ones. All updates are meant to be applied in TTD, while RTD represents CCP4 state after the last update. Given that, UDF file may be generated with the following command:

```
mkupdate -suggest-cmp UD TTD RTD
```

This command will produce the same sort of results as the previous one, but, in addition, it will also spot files removed from the original release. After the update release, the update must be applied to RTD (see 2.3) in order to keep bring it to new reference state. Note that not all update suggestions need to be released. It is Ok to exclude files from update and remove lists, in which case they will be spotted as changed again next time.

4. Update maintenance protocol

Here we list actions to be performed for setting up and maintaining updates following CCP4 release.

4.1 Initial update setup

1) Define GUD and UD paths:

OS	Base Update URL	Path on server (GUD)
Mac OSX	http://www.ccp4.ac.uk/setup/macosx/updates/	/public2/www/setup/macosx/updates
Linux (32 and 64-bit)	http://www.ccp4.ac.uk/setup/linux/updates/	/public2/www/setup/linux/updates
Windows	http://www.ccp4.ac.uk/setup/windows/updates/	/public2/www/setup/windows/updates

2) Edit server statements in *task.xml* files for each system and copy *task.xml* files into GUDs

3) Arrange for work copies of GUDs (replace '/work' for the actual path):

OS	Working update area (WGUD)
Mac OSX	/work/macosx/updates
Linux (32 and 64-bit)	/work/linux/updates
Windows	/work/windows/updates

4) Arrange for CCP4 template directories. For incremental updates, create:

OS	CCP4 template areas
Mac OSX	/work/macosx/templates/6.3.0-updates
Linux 32-bit	/work/linux/templates/6.3.0-x86-updates
Linux 64-bit	/work/linux/templates/6.3.0-x86_64-updates
Windows	/work/windows/templates/6.3.0-updates

and leave all these directories empty.

For comparative updates, copy $CCP4/ into both target and reference template directories:

OS	CCP4 template areas
Mac OSX	/work/macosx/templates/6.3.0-target /work/macosx/templates/6.3.0-ref
Linux 32-bit	/work/linux/templates/6.3.0-x86-target /work/linux/templates/6.3.0-x86-ref
Linux 64-bit	/work/linux/templates/6.3.0-x86_64-target /work/linux/templates/6.3.0-x86_64-ref
Windows	/work/windows/templates/6.3.0-target /work/windows/templates/6.3.0-ref

5) Initialize work UD copies (WUDs) and copy them to release area:

```
mkupdate -init 6.3.0        /work/macosx/updates
mkupdate -init 6.3.0-x86    /work/linux/updates
mkupdate -init 6.3.0-x86_64 /work/linux/updates
mkupdate -init 6.3.0        /work/windows/updates
cp -r /work/macosx/updates/6.3.0       /public2/www/setup/macosx/updates/
cp -r /work/linux/updates/6.3.0-x86    /public2/www/setup/linux/updates/
cp -r /work/linux/updates/6.3.0-x86_64 /public2/www/setup/linux/updates/
cp -r /work/windows/updates/6.3.0      /public2/www/setup/macosx/updates/
```

6) Arrange for directories to keep update description files:

OS	CCP4 update description areas
Mac OSX	/work/macosx/updates/6.3.0-dsc
Linux 32-bit	/work/linux/updates/6.3.0-x86-dsc
Linux 64-bit	/work/linux/updates/6.3.0-x86_64-dsc
Windows	/work/windows/updates/6.3.0-dsc

This completes the initial update setup.

4.2 Update cycle

1) Add updates to either “update” or “target” template directories, depending on the model chosen
2) Generate update description files. For cumulative incremental updates (*cf* 3.9):

```
cd /work/macosx/updates/6.3.0-dsc
mkupdate -suggest-inc /work/macosx/templates/6.3.0-updates \
                      /work/macosx/updates/6.3.0 /work/tmp
cd /work/linux/updates/6.3.0-x86-dsc
mkupdate -suggest-inc /work/linux/templates/6.3.0-x86-updates \
                      /work/linux/updates/6.3.0-x86 /work/tmp
cd /work/linux/updates/6.3.0-x86_64-dsc
mkupdate -suggest-inc /work/linux/templates/6.3.0-x86_64-updates \
                      /work/linux/updates/6.3.0-x86_64 /work/tmp
cd /work/windows/updates/6.3.0-dsc
mkupdate -suggest-inc /work/windows/templates/6.3.0-updates \
                      /work/windows/updates/6.3.0 /work/tmp
```

For comparative updates (*cf* 3.9):

```
cd /work/macosx/updates/6.3.0-dsc
mkupdate -suggest-cmp /work/macosx/updates/6.3.0 \
                      /work/macosx/templates/6.3.0-target \
                      /work/macosx/templates/6.3.0-ref
cd /work/linux/updates/6.3.0-x86-dsc
mkupdate -suggest-cmp /work/linux/updates/6.3.0-x86 \
                      /work/linux/templates/6.3.0-x86-target \
                      /work/linux/templates/6.3.0-x86-ref
cd /work/linux/updates/6.3.0-x86_64-dsc
mkupdate -suggest-cmp /work/linux/updates/6.3.0-x86_64 \
                      /work/linux/templates/6.3.0-x86_64-target \
                      /work/linux/templates/6.3.0-x86_64-ref
cd /work/windows/updates/6.3.0-dsc
mkupdate -suggest-cmp /work/windows/updates/6.3.0 \
                      /work/windows/templates/6.3.0-target \
                      /work/windows/templates/6.3.0-ref
```

This procedure generates suggestive UDFs in **dsc* directories. Newly created UDFs are named as UID.dsc, where UID is aouto-incremented Update ID. For first update, UID is set to be ‘update-001’ and should be manually changed in (3).

3) Check and revise update file lists and remove file lists in generated UDFs (found in **-dsc* directories), and add update descriptions and update messages (if necessary). For 1st update, also edit Update ID and rename UDFs as *UID.dsc*.
4) Generate update packages:

```
mkupdate /work/macosx/updates/6.3.0-dsc/UID.dsc  \
         /work/macosx/updates/6.3.0
mkupdate /work/linux/updates/6.3.0-x86-dsc/UID.dsc  \
         /work/linux/updates/6.3.0-x86
mkupdate /work/linux/updates/6.3.0-x86_64-dsc/UID.dsc  \
         /work/linux/updates/6.3.0-x86_64
mkupdate /work/windows/updates/6.3.0-dsc/UID.dsc  \
         /work/windows/updates/6.3.0
```

5) Apply updates to test setups and check them:

```
mkupdate -apply UID /work/macosx/updates/6.3.0 \
                    /work/macosx/ccp4-6.3.0
mkupdate -apply UID /work/linux/updates/6.3.0-x86 \
                    /work/linux/ccp4-6.3.0-x86
mkupdate -apply UID /work/linux/updates/6.3.0-x86_64 \
                    /work/linux/ccp4-6.3.0-x86_64
mkupdate -apply UID /work/windows/updates/6.3.0 \
                    /work/windows/ccp4-6.3.0
```

6) If corrections are needed, go to (3) and use enforcement (-f key) in (4) to re-generate update packages

7) Copy update packages to the release area and make UDFs read-only:

```
cd /work/macosx/updates; chmod a-w 6.3.0-dsc/*
cp 6.3.0/* /public2/www/setup/macosx/updates/6.3.0
cd /work/linux/updates; chmod a-w *-dsc/*
cp 6.3.0-x86/*    /public2/www/setup/linux/updates/6.3.0-x86
cp 6.3.0-x86_64/* /public2/www/setup/linux/updates/6.3.0-x86_64
cd /work/windows/updates; chmod a-w 6.3.0-dsc/*
cp 6.3.0/* /public2/www/setup/windows/updates/6.3.0
```

8) In case of comparative updates, apply update to RTDs:

```
mkupdate -apply UID /work/macosx/updates/6.3.0 \
                    /work/macosx/templates/6.3.0-ref
mkupdate -apply UID /work/linux/updates/6.3.0-x86 \
                    /work/linux/templates/6.3.0-x86-ref
mkupdate -apply UID /work/linux/updates/6.3.0-x86_64 \
                    /work/linux/templates/6.3.0-x86_64-ref
mkupdate -apply UID /work/windows/updates/6.3.0 \
                    /work/windows/templates/6.3.0-ref
```

This completes the update cycle.

Appendix 1. Update Task File

This file describes the “task” for PM manager. Task consists from the description of data to be fetched from the server (<fetch> xml categories), and pages, or functional blocks, which PM should display in given order (<page> xml categories). The task file allows one to change certain elements in PM, such as legends and buttons, and even introduce additional pages/steps, without having PM replaced on client machines.

Any text in the task file, that needs to be displayed in PM pages, may include html tags for advanced formatting. In order not to interfere with xml tags of the file, html tags should be taken in curly brackets, e.g. {b}...{/b} for bold font.

Task file is meant to be self-explicable, please read pseudo-comments between $$ signs in the file.

```
<task>
  $$============================================================$$
  $$                                                            $$
  $$    Task description for CCP4 Package Manager Update        $$
  $$                  MAC OS X VERSION                          $$
  $$                                                            $$
  $$============================================================$$

  $$ "version" referes to the version of Package Manager.       $$
  $$ Package Manager will refuse improper versions and          $$
  $$ prompt the user to update                                  $$
  <version>1.04</version>

  $$ "fetch" records specify data that Package Manager          $$
  $$ downloads from CCP4 server and keeps internally. Data      $$
  $$ type is identified by the "type" attribute, which must     $$
  $$ have one of predefined values.                             $$

  <fetch type="updates_list">updates.xml</fetch>

  <page template="title">

    $$ text to be displayed on the right-hand side of the page $$
    <rhs type="text">
{b}Collaborative Computational Project No. 4{/b}{br}
{font size="-1"}
Science and Technology Facilities Council{br}
Rutherford Appleton Laboratory{br}
Didcot, Oxon, OX11 0FA, United Kingdom{br}
http://www.ccp4.ac.uk{br}
{/font}
{br}
{h1}CCP4 Software Suite{/h1}
{h3}Version 6.3.0{/h3}
{h2}Update{/h2}
{br}
This program will guide you through the update process for{br}
```

```
CCP4 Software Suite on your computer.
{p}
    </rhs>

  $$ If “Next” button is removed from the screen, Package    $$
  $$ will show “title” page for 1.5 secs and them move on to $$
  $$ the next one.                                           $$
    <next>0</next>

  $$ Request the presence of "Cancel" button on the page.    $$
    <cancel>1</cancel>

  </page>

  <page template="update_list">

    $$ Base URL of the server from which the updates will be $$
    $$ This may override the URL hard-coded in the Package   $$
    $$ Manager, if necesary                                  $$
    <server>http://www.ccp4.ac.uk/setup/macosx/updates</server>

    <title>
{h2}CCP4 update, step 1 out of 3{/h2}
{h3}Choose updates to install or remove:{/h3}
    </title>

    <next>1</next>
    <cancel>1</cancel>

  </page>

  <page template="downloads">
    <title>
{h2}CCP4 update, step 2 out of 3{/h2}
{h3}Download selected updates{/h3}
    </title>

    $$ Allow for the "Back" button here                      $$
    <back>1</back>

    $$ When the "Next" button is switched off here, Package  $$
    $$ Manager will advance to the next page once all        $$
    $$ downloads are successfully completed. Otherwise, the  $$
    $$ user have to press "Next" in order to proceed. Note,  $$
    $$ that even when switched off, "Next" button still may  $$
    $$ be forced on the page at some navigation scenarios.   $$
    <next>0</next>
    <cancel>1</cancel>

  </page>
```

```
  <page template="update">
    <title>
{h2}CCP4 setup, step 3 out of 3{/h2}
{h3}Apply changes{/h3}
    </title>
    <back>1</back>
    <cancel>1</cancel>
  </page>

</task>
```

Appendix 2. Command-line options for *mkupdate* utility

Abbreviations:

UD update directory (*cf* Section 1)
UDF update description file (*cf* Section 3)
UID update ID (*cf* Section 3)
ITD incremental template directory (*cf* 3.9)
TTD target template directory (*cf* 3.9)
RTD reference template directory (*cf* 3.9)

Option key	Parameters	Description
none	UDF UD	Adds update, described in UDF, to UD. This results in the generation of update package *UD/UID.pck*, and putting service information into update contents file *UD/updates.xml* .
-f	UDF UD	Same as above, but enforces the replacement of update with UID, given in UDF, if it exists in UD. This may be applied only to the update with highest UID. Updates with lower UIDs cannot be replaced.
-desc-template	*none*	Prints template UDF into standard output
-apply	UID UD *rootDir*	Applies update with given UID from given UD to the content of *rootDir*, the letter being considered as a proxy for $CCP4.
-apply-all	UD *rootDir*	Applies all updates from given UD to the content of *rootDir*.
-suggest-inc	ITD UD *tmpDir*	Suggests UDF based on comparison of ITD and UD contents, *tmpDir* is a scratch directory. The UDF named *UID.dsc*, where UID is auto-incremented from UD content, is written in the current directory. See Section 3.9 for details.
-suggest-cmp	UD TTD RTD	Suggests UDF based on comparison of TTD and RTD. The UDF named UID.dsc, where UID is auto-incremented from UD content, is written in the current directory. See Section 3.9 for details.
-help	*none*	Prints usage instructions into standard output
none	*none*	Prints usage instructions into standard output