

# The NCBI C++ Toolkit

## 3: Retrieve the Source Code (FTP and Subversion)

Created: April 1, 2003.

Last Update: May 16, 2013.

### Overview

The overview for this chapter consists of the following topics:

- [Introduction](#)
- [Chapter Outline](#)

#### Introduction

The first step in working with the C++ Toolkit is getting the source code, which can be either downloaded from anonymous FTP or checked out from a Subversion repository. This chapter describes both methods and the use of utility scripts that can help getting only the necessary source code components.

If you are interested in downloading source code from the C Toolkit instead of the C++ Toolkit, please see [Access to the C Toolkit source tree Using CVS](#).

#### Chapter Outline

The following is an outline of the topics presented in this chapter:

- [Public Access to the Source Code via FTP](#)
- [Read-Only Access to the Source Code via Subversion](#)
- [Read-Write Access to the Source Code via Subversion \(NCBI only\)](#)
  - [NCBI Source Tree Contents](#)
  - [Source Code Retrieval under Unix](#)
    - ◆ [Retrieval of the C++ Toolkit Source Code Tree](#)
      - [Checking Out the Development NCBI C++ Toolkit Source Tree](#)
      - [Checking Out the Production NCBI C++ Toolkit Source Tree](#)
      - [svn\\_core: Retrieving core components](#)
      - [import\\_project: Retrieve Source for an Existing Project](#)
      - [update\\_core: Update the Portable and Core Components](#)
      - [update\\_projects: Check out and Update Sources of Selected Projects](#)
    - [Source Code Retrieval under MS Windows](#)
    - [Source Code Retrieval under Mac OS X](#)
- [Source Tree Structure Summary](#)

### Public Access to the Source Code via FTP

- [FTP Download Now](#)

- **Available FTP Archives:** Select the archive for your system. When the dialog box appears, choose the destination in your file system for the downloaded archive.  
**Note:** With some browsers, you may need to right-click-and-hold with your mouse and use the 'Save Link As...', 'Copy to Folder...', or similar options from the drop-down menu to properly save the archive. For a current list of the source code archives for different operating system/compiler combinations consult the current Release Notes available at [ftp://ftp.ncbi.nih.gov/toolbox/ncbi\\_tools++/CURRENT/RELEASE\\_NOTES.html](ftp://ftp.ncbi.nih.gov/toolbox/ncbi_tools++/CURRENT/RELEASE_NOTES.html)
- **Unpack the Source Archive**
  - *Unix and Macintosh Systems*  
The Unix distributions have been archived using the standard tar command and compressed using gzip. When unpacked, all files will be under the directory `ncbi_cxx--<version_number>`, which will be created in the current directory. (**Caution:** If `ncbi_cxx--<version_number>` already exists, tar extraction will overwrite existing files.) To unpack the archive: `gunzip -c ncbi_cxx--*.tar.gz | tar xvf -`
  - *Windows Systems*  
The Microsoft Windows versions of the source distribution have been prepared as self-extracting executables. By default a sub-folder `ncbi_cxx--<version_number>` will be created in the current folder to contain the extracted source. If `ncbi_cxx--<version_number>` already exists in the folder where the executable is launched, user confirmation is required before files are overwritten. To actually perform the extraction, do one of the following:
    - ◆ Run the executable from a command shell. This will create the sub-folder in the shell's current directory, even if the executable is located somewhere else.
    - ◆ Double-click on the archive's icon to create `ncbi_cxx--<version_number>` in the current folder.
    - ◆ Right-click on the archive's icon, and select 'Extract to...' to unpack the archive to a user-specified location in the filesystem.

## Read-Only Access to the Source Code via Subversion

The following options for read-only access to the C++ Toolkit Subversion repository are available to the public:

- Checking out the source tree directly from the repository (e.g. `svn co http://anonsvn.ncbi.nlm.nih.gov/repos/v1/trunk/c++`).
- Browsing the repository with an HTTP browser (e.g. <http://www.ncbi.nlm.nih.gov/viewvc/v1/trunk/c++>).
- Accessing the repository with a WebDAV client (also using `http://anonsvn.ncbi.nlm.nih.gov/repos/v1/trunk/c++` – although some clients may require `dav://` instead of `http://`).

## Read-Write Access to the Source Code via Subversion (NCBI only)

**Note:** This section discusses read-write access to the Subversion repository, which is only available to users inside NCBI. For public access, see the section on [read-only access](#).

Subversion client installation and usage instructions are available on separate pages for [UNIX](#), [MS Windows](#), and [Mac OS](#) systems.

For a detailed description of the Subversion Version Control System please download the book "Version Control with Subversion" or run the command `svn help` on your workstation for quick reference.

The following is an outline of the topics presented in this section. Select the instructions appropriate for your **development** environment.

- [NCBI Source Tree Contents](#)
- [Source Code Retrieval under Unix](#)
  - [Retrieval of the C++ Toolkit Source Tree](#)
    - ◆ [Checking Out the Development NCBI C++ Toolkit Source Tree](#)
    - ◆ [Checking Out the Production NCBI C++ Toolkit Source Tree](#)
    - ◆ [svn\\_core: Retrieving core components](#)
    - ◆ [import\\_project: Retrieve Source for an Existing Project](#)
    - ◆ [update\\_core: Update the Portable and Core Components](#)
    - ◆ [update\\_projects: Check out and Update Sources of Selected Projects](#)
- [Source Code Retrieval under MS Windows](#)
- [Source Code Retrieval under Mac OS X](#)

## NCBI Source Tree Contents

The NCBI C++ Toolkit Subversion repository contains all source code, scripts, utilities, tools, tests and documentation required to build the Toolkit on the major classes of operating systems (Unix, MS Windows and Mac OS).

## Source Code Retrieval under Unix

### *Retrieval of the C++ Toolkit Source Code Tree*

This section discusses the methods of checking out the entire source tree or just the necessary portions. An important point to note is that the entire NCBI C++ tree is very big because it contains a lot of internal projects. There are also numerous platform-specific files, and even platform-specific sub-trees, which you will never need unless you work on those platforms. Therefore it is frequently sufficient, and in fact, usually advisable, to retrieve only files of interest using the shell scripts from the path (it is in the default \$PATH):

`/am/ncbiapdata/bin`

They can also be checked out directly from the Subversion repository at:

`https://svn.ncbi.nlm.nih.gov/repos/toolkit/trunk/internal/scripts/common`

The auxiliary script `svn_core` checks out only the core NCBI C++ Toolkit sources for a desired platform. A similar auxiliary script, `import_project`, can be used to import the source from a single project. To facilitate the creation of a new project, use the script `new_project` which generates new directories and makefiles for the new project from templates. This script also checks out a specified sample application from the source tree that may be adapted for the new project or built directly as a demonstration.

Checking out the whole Toolkit source tree using a Subversion client can take 15 minutes or more. However, the script `svn_toolkit_tree` (available to NCBI users via the default PATH on most UNIX hosts) produces the same result in only 10-30 seconds. The `svn_toolkit_tree` script combines a daily archive with an update of the working copy to bring it up-to-date. This

produces the same set of files and revisions as running `svn checkout`, but in much less time. Besides speed, the differences between using a Subversion client and the `svn_toolkit_tree` script include:

- The `svn_toolkit_tree` script may not be compatible with your Subversion client. If your client is older than the version used to create the archive, you may not be able to access the archive.
- The `svn_toolkit_tree` script requires that your current directory does not contain a subdirectory with the name that the script is about to create (see below for the subdirectory names created by the script).

There are three archives currently available:

- `trunk`
- `trunk-core`
- `production`

which correspond to the following flavors of the C++ Toolkit trees:

- <https://svn.ncbi.nlm.nih.gov/repos/toolkit/trunk/internal/c++>
- <https://svn.ncbi.nlm.nih.gov/repos/toolkit/trunk/c++>
- <https://svn.ncbi.nlm.nih.gov/repos/toolkit/production/candidates/production.HEAD/c++>

which the script will deploy to the local subdirectory named, respectively:

- `toolkit-trunk/`
- `toolkit-trunk-core/`
- `toolkit-production/`

For example, to retrieve the current TRUNK version of the "core" part of the C++ Toolkit tree (the part without the GUI and INTERNAL projects), run:

```
$ svn_toolkit_tree trunk-core
/net/snowman/vol/projects/ncbisoft/toolkit_trees/trunk-core.tar.gz ->
toolkit-trunk-core/
Updating toolkit-trunk-core/...

$ ls toolkit-trunk-core/
compilers configure doc include scripts src
```

### *Checking Out the Development NCBI C++ Toolkit Source Tree*

You can check out the entire development NCBI C++ source tree from the repository to your local directory (e.g., `foo/c++/`) just by running:

```
cd foo
svn checkout https://svn.ncbi.nlm.nih.gov/repos/toolkit/trunk/c++
```

For internal projects use:

```
cd foo
svn checkout https://svn.ncbi.nlm.nih.gov/repos/toolkit/trunk/internal/c++
```

**Caution:** Be aware that sources checked out through the development source tree have the latest sources and are different from the public release that is done at periodic intervals. These sources

are relatively unstable "development" sources, so they are not guaranteed to work properly or even compile. Use these sources at your own risk (and/or to apply patches to stable releases). The sources are usually better by the end of day and especially by the end of the week (like Sunday evening).

### *Checking Out the Production NCBI C++ Toolkit Source Tree*

Besides the development NCBI C++ source tree, there is the C++ Toolkit "production" source tree that has been added to the public Subversion repository. This tree contains stable snapshots of the "development" C++ Toolkit tree. Please note that these sources are lagging behind, sometimes months behind the current snapshot of the sources.

You can check out the entire "production" NCBI C++ source tree from the public repository to your local directory by running:

```
svn co https://svn.ncbi.nlm.nih.gov/repos/toolkit/production/latest/c++
```

This repository path corresponds to the latest production build of the Toolkit. If you want to check out sources for an older production build, please specify the exact date of that build as follows:

```
svn co https://svn.ncbi.nlm.nih.gov/repos/toolkit/production/20031212/c++
```

where 20031212 is the date when this specific build was originated. You can easily find out the available production builds by running

```
svn ls https://svn.ncbi.nlm.nih.gov/repos/toolkit/production
```

This command will print directories under production/, which correspond to the production builds.

### *svn\_core: Retrieving core components*

The NCBI C++ Toolkit has many features and extensions beyond the core of portable functionality. However, one often wants to obtain a set of core sources that is free of non-portable elements, and the `svn_core` script performs this task across the range of supported platforms. Options to the basic command allow the developer to further tailor the retrieved sources by including (or excluding) certain portions of the Toolkit.

For usage help, run `svn_core` without arguments.

**Note:** `svn_core` is not available on Windows.

Table 1 describes the arguments of `svn_core`. Only the target directory and SVN branch arguments are mandatory.

Some directories are always checked out, regardless of command-line arguments. These are shown in Table 2. (All paths are relative to the repository path `https://svn.ncbi.nlm.nih.gov/repos/toolkit/trunk/c++`.)

Other directories may or may not be checked out, depending on the `<branch>` and `<platform>` options. These are shown in Table 3.

Still other directories may be checked out depending on the `--with/--without-<feature>` options. These are shown in Table 4.

*import\_project: Retrieve Source for an Existing Project*

Usage:

```
import_project <SVN_relative_tree_path> [builddir]
```

In many cases, you work on your own project which is a part of the NCBI C++ tree, and you do not want to check out, update and rebuild the entire NCBI C++ tree. Instead, you just want to use headers and libraries of the pre-built NCBI C++ Toolkit to build your project.

The shell script `import_project` will check out your project's `src` and include directories from the repository and create temporary makefiles based on the project's customized makefiles. The new makefiles will also contain a reference to the pre-built NCBI C++ Toolkit.

For example:

```
import_project serial/datatool
```

will check out the `datatool` project from the NCBI C++ tree (`trunk/c++/{src,include}/serial/datatool/`), and create a makefile `Makefile.datatool_app` that uses the project's customized makefile `Makefile.datatool.app`. Now you can just go to the created working directory `c++/src/serial/datatool/` and build the application `datatool` using:

```
make -f Makefile.datatool_app
```

*update\_core: Update the Portable and Core Components*

Usage:

```
update_core [--no-projects] [<dirs>]
```

Once you have obtained the core C++ Toolkit sources, with `svn_core` or otherwise, the local copies will become out of sync with the master SVN repository contents when other developers commit their changes. `update_core` will update your local core source tree with any changed files without the side-effect of simultaneously checking out non-core portions of the tree. Subdirectory `*/internal` does not get updated by this script.

The `--no-projects` switch excludes any Windows or MacOS project files from the update. Specifically, those subdirectory names of the form `*_prj` are skipped during the update when this flag is set.

The list `<dirs>`, when present, identifies the set of directories relative to the current directory to update. The default list of updated directories is:

- .
- compilers
- doc
- include
- scripts
- src

Note that the default list is not pushed onto a user-supplied list of directories.

***update\_projects: Check out and update Source of Selected Projects***

Usage:

```
update_projects <project-list> [<directory>]
```

Script `update_projects` facilitates the original retrieval and subsequent updates of selected parts of the Toolkit tree. Because the source code and makefiles are distributed over more than one subdirectory under repository path `trunk/c++`, this script assembles the set of required files and places them in your local C++ source tree.

The projects to be retrieved (or updated) must be specified in the command line as the `<project-list>` parameter. Its value can be either of the following:

- Explicit specification of the pathname of the project listing file. This project listing file can contain project directory names as well as references to other project listings and must be formatted according to the simple syntax used by the `configure` script.
- Specify one of the standard project names. Standard projects are those whose project listing files are located in one of the system directories, which are `trunk/c++/scripts/projects` and `trunk/c++/scripts/internal/projects`. When a project name is specified on the command line, the “.lst” extension is added to it and the resulting file name is searched for in the above mentioned system directories.

The parameter to `update_projects` indicates the target directory where the sources will be checked out to and where the project will be configured and built. This parameter is optional and is set to the current directory by default.

**Source Code Retrieval under MS Windows**

- 1 In NCBI, the SVN clients must be set up and ready to use. Ask Systems if you don't have the client installed on your workstation. If you are working outside of NCBI, then you can download the latest version of Subversion from <http://subversion.tigris.org/servlets/ProjectDocumentList?folderID=91>. Run the Subversion installer and follow the instructions. The latest version may not come with an executable installer though. In this case, please unpack the zip archive with the latest Subversion binaries to a local directory, for example `C:\Program Files\svn-win32-1.4.2`. Change the PATH environment variable so that it points to the bin subdirectory under your Subversion installation directory, for example set `PATH=%PATH%;C:\Program Files\svn-win32-1.4.2\bin`
- 2 Start your favorite command shell. Change current directory to the designated working directory. At the command prompt, type: `svn co https://svn.ncbi.nlm.nih.gov/repos/toolkit/trunk/c++`
- 3 Modify source files as required. Refer to `Svnbook` for the documentation on particular Subversion commands. Monitor your changes using `svn diff`, synchronize your working copy with the trunk using `svn update`, and finally commit them using `svn commit`.

The rest should be the same as when using Subversion under UNIX systems. See [Source Code Retrieval under Unix](#).

**Source Code Retrieval under Mac OS X**

Download and install the latest Subversion binaries for MacOSX from <http://subversion.tigris.org/>.

The rest should be the same as when using Subversion under UNIX systems. See [Source Code Retrieval under Unix](#).

## Source Tree Structure Summary

To summarize the Getting Started page, the source tree is organized as follows:

- The top-level has configuration files and the directories include/, src/, scripts/, compilers/ and doc/
- The src and include directories contain "projects" as subdirectories. Projects may contain sub-projects in a hierarchical fashion.
- src/ additionally contains makefile and meta-makefile templates.
- Projects contain "modules" and various customized makefiles and meta-makefiles to control their compilation.



Table 1. svn\_core Arguments

Argument	Description	Permitted Values
<dir>	Path to where the source tree will be checked out. This argument is <b>required</b> .	A valid writable directory name (must not exist already); name cannot start with "-".
<branch>	Which branch of the source tree to check out. This argument is <b>required</b> .	core - toolkit/trunk/c++ development - toolkit/trunk/internal/c++ production - toolkit/production/candidates/trial/c++ prod-head - toolkit/production/candidates/production.HEAD/c++ frozen-head - toolkit/production/candidates/frozen.HEAD/c++ trial - toolkit/production/candidates/trial/c++ release - toolkit/release/public/current/c++ gbench - gbench/branches/1.1 gbench2 - gbench/trunk (See c++-branches.txt for an up-to-date list of branches.)
--date	Check out as at the start of the specified timestamp. If the --date flag is missing, today's date and current time are used.	A date in a format acceptable to the svn -r argument, for example --date="2013-03-29 19:49:48 +0000". (Do not include curly braces and quote the timestamp if it contains spaces.) See the Revision Dates section in the Subversion manual for details.
--export	Get a clean source tree without .svn directories.	n/a
--<platform>	Obtain sources for the specified platform(s).	--unix - Unix systems; --msvc - Microsoft Visual C++ environment; --mac - Macintosh systems; --cygwin - Cygwin UNIX environment for Windows; --all - all platforms. If no value is supplied, --all is used.
--with-ctools	Check out core projects responsible for working together with the NCBI C Toolkit (the ctools directory). This option is effective by default unless --without-ctools is used.	n/a
--without-ctools	Do not check out core projects responsible for working together with the NCBI C Toolkit (the ctools directory).	n/a
--with-gui	Check out core projects responsible for providing cross-platform graphic user interface capability (the gui directory). This option is effective by default unless --without-gui is used.	n/a
--without-gui	Do not check out core projects responsible for providing cross-platform graphic user interface capability (the gui directory).	n/a
--with-internal	Check out a selection of NCBI-internal core projects. See Table 4 for a detailed list of affected directories.	n/a
--without-internal	Do not check out NCBI-internal core projects.	n/a

--with-objects	Check out the objects, objmgr, and objtools directories and generate serialization code from the ASN.1 specifications. If this flag is not present, those directories are still checked out (unless overridden by the --without-objects flag) but no serialization code is generated.	n/a
--without-objects	Do not check out the objects, objmgr, and objtools directories or generate ASN.1 serialization code. (On Unix platforms the code generation can be done later, during the build.)	n/a

Table 2. List of the directories that are always checked out

Checked out directories	Recursive?
(include src)	no
(include src)/algo	yes
src/app	yes
src/build-system	yes
(include src)/cgi	yes
include/common	yes
(include src)/connect	no
(include src)/connect/ext	yes
include/connect/impl	yes
src/connect/test	yes
(include src)/connect/services	yes
(include src)/corelib	yes
(include src)/db	yes
(include src)/dbapi	yes
(include src)/html	yes
(include src)/misc	yes
(include src)/sample	yes
(include src)/serial	yes
include/test	yes
(include src)/util	yes
scripts	yes

Table 3. Directories that may be checked out depending on branch and platform options

Checked out directories	Recursive?	Options
compilers	yes	<platform> = all
compilers	<b>no</b>	<platform> != all
compilers/cygwin	yes	<platform> = cygwin
compilers/msvc1000_prj	yes	<platform> = msvc
compilers/unix	yes	<platform> = cygwin or mac or unix
compilers/xCode	yes	<platform> = max
compilers/xcode90_prj	yes	<platform> = mac
doc	yes	<branch> = development
include/connect/daemons	yes	<platform> = all or unix
src/check	yes	<platform> != mac
src/connect/daemons	yes	<platform> = all or unix
src/connect/mitsock	yes	<platform> = mac
src/dll	yes	<platform> = all or mac or msvc

Table 4. Directories that may be checked out depending on --with/--without options

Checked out directories	Recursive?	Options
(include src)/ctools	yes	--with-ctools or not --without-ctools
(include src)/gui	yes	--with-gui or not --without-gui
(include src)/internal	no	--with-internal
(include src)/internal/algo	no	--with-internal
(include src)/internal/algo/id_mapper	yes	--with-internal
(include src)/internal/align_model	yes	--with-internal
include/internal/asn_cache	yes	--with-internal
src/internal/asn_cache	no	--with-internal
src/internal/asn_cache/lib	yes	--with-internal
(include src)/internal/blast	no	--with-internal
(include src)/internal/blast/DistribDbSupport	yes	--with-internal
(include src)/internal/contigdb	no	--with-internal
src/internal/demo	yes	--with-internal
(include src)/internal/ID	no	--with-internal
(include src)/internal/ID/utlis	no	--with-internal
(include src)/internal/mapview	no	--with-internal
(include src)/internal/mapview/objects	yes	--with-internal
(include src)/internal/mapview/util	yes	--with-internal
(include src)/internal/myncbi	yes	--with-internal
include/internal/objects	no	--with-internal
(include src)/objects	yes	--with-objects or not --without-objects
(include src)/objmgr	yes	--with-objects or not --without-objects
(include src)/objtools	yes	--with-objects or not --without-objects
src/internal/objects	yes	--with-internal
(include src)/internal/sra	yes	--with-internal
src/internal/test	yes	--with-internal
(include src)/internal/txclient	yes	--with-internal
(include src)/internal/txserver	yes	--with-internal
(include src)/internal/txxml doc	yes	--with-internal