3: Retrieve the Source Code (FTP and Subversion)

Created: April 1, 2003.
Last Update: June 1, 2012.

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

- 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
- **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    The Unix distributions have been archived using the standard tar command and compressed using gzip.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

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 checkout 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 checkout 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

For usage info, run it without arguments:

```
svn_core
```

The arguments to svn_core are discussed in the svn_core Arguments subsection.

Default settings:

```
svn_core <dir> --date=<current> --all --development
```

A typical invocation for "core" source retrieval is shown below for the major three platforms:

```
# Unix:
svn_core <dir> --unix

# Windows:
svn_core <dir> --msvc

# Mac OS:
svn_core <dir> --mac
```

NCBI C++ Toolkit has many features and extensions beyond the core of portable functionality. Often one wants to obtain a set of the Toolkit 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 source by including (or excluding) portions of the Toolkit not checked out by default.

Table 1 describes the arguments of svn_core, along with their default values. Only the target directory is mandatory. The optional --with/without-<feature> argument pairs include or exclude portions of the Toolkit from the checked-out source. While both members of the pair may appear on the command line, only the last one influences the behavior of the script. Certain settings are meaningful only for certain <platform>s (most often with respect to MS Windows platforms).

In this section, all paths and filenames are in the source tree and unless otherwise specified, should be taken as relative to the following common directory within the repository: https://svn.ncbi.nlm.nih.gov/repos/toolkit/trunk/c++.

Common Source: Table 2 lists those being recursively checked out, regardless of the arguments to svn_core.

Platform-Specific Source: in addition to the above common source, the various <platform> options will populate the remainder of the checked-out source tree differently. A laundry-list for each platform option is not provided here, however users inside NCBI may view the source to examine the most current status. In general, the platform and argument-sensitive parts of the source reside in the directories shown in Table 3.

Feature-Specific Source: The --with/--without-<feature> options enable or inhibit portions of the source tree from being included in the checked-out out source tree. Consult Table 4 to identify where a certain option affects the source tree.

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 checkout 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 **required**.	Valid writable directory name (must not exist already); name cannot start with "-"
--<platform>	Obtain sources for the specified platform(s).	--*unix* - Unix systems; --*msvc* - Microsoft Visual C++ environment; --*mac* - Macintosh systems; --*cygwin* - Red Hat's Cygwin UNIX environment for MS Windows; --*all* - all systems. ***Default: --all***
--export	Get a clean source tree without .svn directories.	If present, it excludes Subversion-related directories from the resultant tree by executing Subversion "export" as of the date specified by the date parameter instead of checking out.
--date	Checkout as at the start of the specified timestamp.	If the --date flag is missing, today's date and current time are used.
--with-objects	Generate ASN.1 serialization code in the objects/ directory.	If the --with-objects flag is present, the objects, object manager and object tools are checked out and serialization code is generated from the ASN.1 specifications. If the --with-objects flag is not present, the objects, object manager and object tools are still checked out (unless overridden by the --without-objects flag) but no serialization code is generated. If the --without-objects flag is present, then the object components will not be checked out. (Unix platforms: the code generation can be done later, during the build)
--without-objects	Do not check out the objects, object manager or object tools directory or generate ASN.1 serialization code.	If not present, this flag has no affect and the behavior for --with-objects applies. That is, unless explicitly turned off by providing this argument the objects, object manager and object tools are always checked out. The main purpose of this flag is to ensure that the object components are not checked out.
--with-ctools	Checkout core projects responsible for working together with the NCBI C Toolkit (ctools/ directory).	If not present, it defaults to still checking out the ctools/ directory unless overridden by the --without-ctools flag.
--without-ctools	Do not checkout core projects responsible for working together with the NCBI C Toolkit.	If not present, this flag has no affect and the behavior for --with-ctools applies. That is, unless explicitly turned off by providing --without-ctools, the ctools are always checked out. The main purpose of this flag is to ensure that the ctools components are not checked out.
--with-gui	Checkout projects that depend on wxWindows.	If not present, it defaults to still checking out the gui components unless overridden by the --without-gui flag.
--without-gui	Do not checkout projects that depend on wxWindows.	If not present, this flag has no affect and the behavior for --with-gui applies. That is, unless explicitly turned off by providing --without-gui, the gui components are always checked out. The main purpose of this flag is to ensure that the gui components are not checked out.
--<srctree>	Whether to include internal NCBI components of the source tree.	The --<srctree> is replaced by either **--production** or **--**. If **--**is specified, the development source tree branch for NCBI users is checked out. If **--production** is specified, the production source tree branch for non-NCBI users is checked out. If not specified, this option defaults to **--**.

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

doc	
include/corelib	src/corelib
include/connect	src/connect/test
include/serial	src/serial
include/cgi	src/cgi
include/html	src/html
include/util	src/util
include/bdb	src/bdb
include/dbapi	src/dbapi
include/app	src/app

Table 3. Directories containing argument sensitive parts

compilers
connect/daemons
connect/mitsock
scripts

Table 4. Directories affected by enabling/disabling the --with/--without option

Feature	Affected Directories
ctools	include/ctools, src/ctools, compilers
gui	include/gui, src/gui, compilers
objects	include/objects/*, src/objects/*, compilers, scripts, src