The NCBI C++ Toolkit Book
The NCBI C++ Toolkit Book

The NCBI C++ Toolkit

Release Notes (February, 2005)

Created: February 1, 2005.
Last Update: February 1, 2005.

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

Download Location

ftp://ftp.ncbi.nih.gov/toolbox/ncbi_tools++/2005/Feb_17_2005/

Source Archive Contents

Source Code Archives

- ncbi_cxx_unix--Feb_17_2005.tar.gz -- for UNIX'es (see the list of UNIX flavors below) and MacOSX 10.[2-3] /GCC 3.3
- ncbi_cxx_unix--Feb_17_2005.gtar.gz -- for UNIX'es (see the list of UNIX flavors below) and MacOSX 10.[2-3] /GCC 3.3
- ncbi_cxx_win--Feb_17_2005.exe -- for MS-Windows / MSVC++ 7.1 (self-extracting)
- ncbi_cxx_win--Feb_17_2005.zip -- for MS-Windows / MSVC++ 7.1
- ncbi_cxx_mac_cw--Feb_17_2005.tgz -- for MacOS 10.3.X / CodeWarrior 9.2
- ncbi_cxx_mac_xcode--Feb_17_2005.tgz -- for MacOS 10.3.X / xCode 1.5

The sources correspond to the NCBI production tree sources from patch "RELEASE_FEB_2005", which in turn closely corresponds to the development tree sources from January 18-19, 2005.

New Development

Some of the new development that may introduce potentially backward-incompatible changes are marked with an asterisk(*).

CORELIB -- Portability and Application Framework

1. CNcbiRegistry -- refactored, with base classes split out at various levels. Along the way, it gained support for prioritized collections of sub-registries.
2. CMetaRegistry -- the default search path has changed, and become easier to tune via environment settings.
3. CNcbiRegistry -- added functions for easy retrieval of configuration parameters: GetConfigString, GetConfigInt, and GetConfigFlag.
4. CArgDescriptions -- new flag to allow multiple command-line arguments (EFlags::fAllowMultiple).
5. CDir::Create() -- return TRUE if creating directory already exists.
6. NStr::MatchesMask() -- added string version of this function.
7. Command line arguments processing, added constraint inversion (NOT).
8. CVersionInfo -- added support for version srtings (like "2.5.1")

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

9 Diagnostics -- now allows configuring message filtering by severity.

10 Removed operator bool() from CRef<> and CConstRef<> templates. Pointer conversion will be used instead. Added helper template and macro for easy implementation of boolean operator via pointer.

11 Moved obj_store and plugin_manager_store to CORELIB.

12 Plugin Manager -- added CDllResolver_Getter() to get default DLL resolver.

CONNECT -- Data streaming, Networking, and Dispatching

1 CNamedPipe -- added automatic generation of OS-specific pipe names.

2 FTP client connector (FTP_CreateDownloadConnector) and stream (CConn_FTPDownloadStream) have been implemented (only for download).

3 NetCache client -- implemented BLOB update, allow to engage NCBI load balancer.

UTIL -- Miscellaneous Low-Level APIs

1 CZipDecompressor -- fixed bug in Process() when fCheckFileHeader flag is set.

2 util/bitset -- integrated new version of BM library into NCBI toolkit.

SERIAL -- Data Serialization (ASN.1, XML)

1 XML output stream -- corrected writing namespace of NamedType.

2 XML input stream -- fixed the reading of empty sequence with all optional members represented by self-closed tag.

3 CTreeIterator -- implemented context-dependent filtering.

DATATOOL -- Code Generator and Data Converter Tool

1 Made it possible to process multiple objects in input stream when converting data from one format to another.

2 Implemented parsing of conditional sections in DTD, processing of entities in attribute definition, and processing of compound identifier names (those made of several entities).

3 Corrected generation of XML schema for "sequence of choice" types with attributes, and simple type with a default.

4 Corrected generation of links in DOXYGEN comments.

CGI -- CGI and Fast-CGI Application Framework

1 CCgiApplication -- new callback OnEvent() to allow one catch and handle a variety of states and events happening in the CGI and Fast-CGI applications.

2 CCgiEntry -- keep track of Content-Type headers from POST submissions. Added missing operators !=. Moved operators == and != into class.

3 Added CArgDescriptions class support for CGI. Now it is possible to describe CGI arguments as if they were command-line key arguments.

4 CCgiRequest -- new flag fCaseInsensitiveArgs to allow case-insensitive arguments in query string.

HTML -- HTML Generation Library

1 CHTML_map and CHTML_area -- new classes to support HTML MAP and AREA tags.

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

Berkeley DB API (bdb) -- Much Enriched C++ API Based On BerkeleyDB

1. CBDB_BlobFile -- added UpdateInsert().
2. Cursors -- implemented BLOB update, added support for Read-Modify-Write (RMW) locks.
3. CBDB_Cache -- implemented adaptive checkpoints and individual BLOB timeouts, improved performance. Code moved into a separate library, which is set up as a "standard NCBI plug-in", with registration entry point, fixed interface, class factory, driver name, versioning, etc.
4. Ported to work with Berkeley DB 4.3.

DBAPI -- Generic SQL Database Connectivity

1. FreeTDS database driver -- now works on MS Windows.
2. Refactoring of dbapi/driver/samples DBAPI sample applications -- refactored, now all are based on a common class CDbapiSampleApp.
3. CDBAPI_Cache -- code moved into a separate library, which is set up as a "standard NCBI plug-in", with registration entry point, fixed interface, class factory, driver name, versioning, etc.

ALGO/ALIGN -- Generic Alignment Algorithms

1. Reorganzation of the algo/align branch: XALGOALIGN moved under algo/align/nw and renamed to XALGOALIGNNW. XALGOSPLIGN renamed to XALGOALIGNSPLIGN.
2. CSplignFormatter -- Seq-align output now includes percent identity score.
3. CSplignFormatter -- was modified to use CSeq_id objects in place of strings.

BLAST

1. CBlastNucleotideOptionsHandle -- removed some member functions to simplify its usage.
2. Applications under src/algo/blast/api/demo and src/app/blast_client are NOT meant to provide full functionality of the BLAST C Toolkit binaries, please do not use them as such.
3. Improved error handling when initializing BlastSeqSrc implementations.
4. Removal of eSkipTbck from EBlastTbackExt enumeration, consolidation of means to avoiding the traceback stage of the algorithm.
5. Removal of total HSP limit option from low level structures and options API.
6. Made fetching sequences via the C++ Object Manager more robust and consistent with the C Toolkit.
7. BlastQueryInfo structure -- refactored.
8. RPS-BLAST now supports multiple queries. Streamlined the initialization of RPS-BLAST structures.
9. Reorganization of Blast*Options and Blast*Parameter structures.
10. Various file and functions renamed to improve clarity of the code and adherence to C++ Toolkit coding conventions.
11. Implemented reevaluation with ambiguities for translated ungapped searches.
12. Added methods to retrieve warnings/errors from CBl2Seq and CDbBlast.

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

BIO-OBJECTS -- Bio-Object Specific Utility Functions (Not Involving OM++)

1. CSeq_id -- support several new prefixes in IdentifyAccession() method.
2. CGen_code_table -- new LoadTransTable() method to support optionally replacing the built-in copy with an updated or customized version.
3. CSeq_id_Handle -- added helper functions to avoid getting intermediate instance of CSeq_id_Mapper: HaveMatchingHandles(), HaveReverseMatch(), GetMatchingHandles(), GetReverseMatchingHandles().

ALGO/PHY_TREE -- Phylogenetic and bio tree algorithms

1. Added new test-demo to algo/phy_tree/test/test_biotree.cpp.
2. New converters for phylogenetic trees. PhyTree serialization support.

OM++ -- Object Manager -- For Retrieving and Processing Bio-Objects

1. Added class CTSE_Handle.
2. Implemented auto-release of unused TSEs in scope. TSEs are locked from release by any object manager handle or iterator. Scope keeps last few unlocked TSEs from immediate release. Scope also maintains links from one TSE to indirectly used ones, like TSEs with far segments of segmented sequence. Number of unlocked TSEs can be changed by configuration variable SCOPE_AUTORELEASE_SIZE in OBJMGR section of configuration file. TSE auto-release can be turned off by setting SCOPE_AUTORELEASE boolean flag to FALSE.
3. Added CSeqVector constructor from CBioseq_Handle to allow used TSE linking.
4. Added CSeqMap::CanResolveRange() with SSeqMapSelector argument to allow used TSE linking.
5. Added SAnnotSelector::SetExcludeExternal().
6. Added helper class CBlobIdKey to use it as key, and to print BlobId.
7. Added method CSeq_annot_Handle GetAnnot() to various annotation iterators: CFeat_CI, CGraph_CI,and CAlign_CI.
8. Changed TSE suppression levels to blob state flags.
9. Removed many methods previously marked as deprecated.
10. Fixed sorting of circular features.
11. Redesigned CBioseq_CI not to collect all bioseqs in constructor.
12. Added CBioseq_Handle::GetRangeSeq_loc(): creates CSeq_loc to be used with annotation iterators instead of start/stop values.
13. Added proxy methods for CSeq_annot getters.
14. Reduced number of CSeqMap::FindResolved() methods, simplified BeginResolved () and EndResolved().
15. CTSE_Handle -- added new methods to get blob state IsSuppressed*() and IsDead().
16. Simplified resulting seq-loc in Add/Merge/Subtract. Moved seq-loc operations to CSeq_loc, modified flags. Removed old SeqLocMerge.

OM++ LOADERS/READERS -- Data Retrieval Libraries for OM++

1. LDS data loader now can read sequence entries out of binary ANS.1 RefSeq release files.
2. Added possibility to reload TSEs by their BlobId in GenBank loader.

3. Added conversion of BlobId to string in GenBank loader.
4. Merged "ID1" and "cached ID1" libraries. ID1 reader class factory now can create any of these two -- depending on parameters.
5. Loaders and readers are now set up as "standard NCBI plug-ins", with registration entry point, fixed interface, class factory, driver name, versioning, etc.

BUILD FRAMEWORK (UNIX)

1. CONFIGURE now allows for a wider range of Berkeley DB layouts by honoring settings of BERKELEYDB_{INCLUDE,LIBPATH,LIBS} from the environment when --with-berkeleydb=... has not been supplied.
2. Makefiles may specify DTD_PROJ, a la ASN_PROJ.

BUILD FRAMEWORK (MSVC++.NET)

1. Modified import_project.wsf and new_project.wsf to reduce dependency on CVS.
2. Changed new_project.wsf to generate new project in a separate folder, and to make it possible to enter the data interactively.
3. Improved diagnostics output.

PTB -- Project Tree Builder for MSVC++ .NET

1. Generate and use configuration-dependent site localization file - ncbiconf_msvc_site.$(ConfigurationName).h.
2. Configure libraries with choice and 3rd-party library dependencies and macros for each build configuration independently.
3. In 3rd-party library description (project_tree_builder.ini): allow macros, treat library INCLUDE as list, and made it possible to add standard libraries along with 3rd-party ones (STDLIB).
4. Corrected handling of project lists, and configurable macros. Correctly process MSVC project tuning files for DLLs (Makefile.*.msvc). In generated solution, correctly set dependency on user projects.
5. Process DTD_PROJ entry in Makefile.in - to generate library project by a DTD specification.
6. Improved diagnostic output.

3RD-PARTY PACKAGES

1. Upgraded code to work with newer versions of some 3rd-party packages.
2. The 3rd-party sources' bundle for MSVC++ has some packages upgraded to their newer version.

PYTHON -- Scripting Language Support

1. dbapi/lang/python -- the first draft of a Python database extension module based on the NCBI C++ Toolkit's DBAPI. It implements a big part of the Python Database API Specification v2.0 (http://www.python.org/peps/pep-0249.html), and allows Python developers to work with the databases supported by DBAPI (such Sybase and MS SQL Server.

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

APPLICATIONS

APP/NetCache -- General Purpose Network Cache

1 Fixed a number of bugs, improved performance, added logging. Some optimization changes to reduce server choking under heavy load.

Documentation

Document Location

The documentation is available online at http://www.ncbi.nlm.nih.gov/books/bv.fcgi?call=bv.View..ShowTOC&rid=toolkit.TOC&depth=2 as a book titled "The NCBI C++ Toolkit". This is an online searchable book.

The C++ Toolkit book also provides PDF version of the chapters; although these are not up to date in this release. The PDF version can be accessed by a link that appears on each page.

The older HTML documentation has been deprecated and is no longer being updated, and "The NCBI C++ Toolkit" online book at the previously listed URLs is the official documentation.

Document Content

Documentation has been grouped into chapters and sections that provide a more logical coherence and flow. New sections and paragraphs continue to be added to update and clarify the older documentation or provide new documentation. The chapter titled "Introduction to the C++ Toolkit" gives an overview of the C++ Toolkit. This chapter contains links to other chapters containing more details on a specific topic and is a good starting point for the new comer.

The DOXYGEN source browser is used to complement the traditional docs with structured DOXYGEN-generated "Library Reference" ones based on the in-source comments. You can access the DOXYGEN source browser for the NCBI code from:

http://www.ncbi.nlm.nih.gov/IEB/ToolBox/CPP_DOC/doxyhtml/

The above link is also available under the "Browsers" that appears on each page.

You can also access the CVS code repository via a Web interface. These links also appear in the sidebar box on each page.

A C/C++ Symbol Search query appears on each page of the online Toolkit documentation. You can use this to perform a symbol search on the public or in-house versions of LXR, Doxygen and Library. The Library search runs a CGI script that lists the library name where the symbol (such as function) is defined in.

The current release notes as well as past release notes are now in an appendix in the C++ Toolkit Book and a link to the current release notes appears on each page of the online Toolkit document.

Building on the MacOS

We now build all the libraries and most of the applications including the Genome Workbench (gbench), some test and a few demo applications. We also build the FLTK library's GUI editor, fluid.

All apps are built as application bundles except gbench_plugin_scan and datatool which are built as command line apps. Any of the applications can be built as command line apps by tweaking the build scripts or the CodeWarrior projects.

When building the toolkit with xCode, the latest version of xCode (at least 1.5) is required for the trouble-free build. Build procedure is as follows: open, build and run a project file in compilers/xCode. This is a GUI tool to generate a new NCBI C++ Toolkit xCode project. You'll have an option to specify third-party installation directories and choose which packages (libs, applications and tests) to include into the final project.

xCode build fully support all the latest Apple innovations: distributed builds, optional CPU specific optimization , pre-compiled headers, fix & continue, code cleanup and Zero Link (with few exceptions).

xCode builds all libraries as a Mach-O dynamically linked shared ones (.dylib) and all Genome Workbench plugins as Mach-O bundles (also .dylib extension). Note, that xCode will place Genome Workbench plugins inside Genome Workbench application bundle (Genome Workbench.app/Contents/MacOS/plugins).

To build the toolkit from a command line use an AppleScript editor to open and run the script files makeLibs.met and makeApps.met. You must use a script editor capable of opening a script file larger than 32K, such as Apple's Script Editor v2.0, or Smile. Script Editor v1.9 will not work since makeLibs.met just got too big. The command line tool osascript also works.

Projects include targets to compile with BSD/Apple headers and libraries and with MSL headers and libraries, but plugins (for gbench) built with MSL do not link properly, and so, because of lack of interest to keep up with changes, some source code does not currently compile with MSL. Hopefully this will become easier to work with and get fixed with CodeWarrior v.9.

Targets to be compiled can be controlled by including an empty file or folder in the compilers:mac_prj folder with the name 'Build' followed by the keywords of the targets you want built. The keywords are: MSL, BSD, Debug and Final. For example, to build only the BSD Debug targets use: "Build BSD Debug", to build both BSD debug and release (final) versions: "Build BSD". The default is to build everything.

If you install the C++ toolkit under a different name than "ncbi_cxx" or in a different location than your home directory, you can edit the script's properties, pRootFolderName and pRootFolderPath, to override these defaults. Note: these paths, and those mentioned below, must be entered in mac format (e.g. disk:Users:username:) not Unix format (e.g. /Users/username/). The disk name (and its following colon) may be omitted.

Certain third party libraries (see Table 1) are required to build some parts of the C++ toolkit. The scripts will try and find them if they are in your home directory, or you can specify where they were installed using properties at the beginning of the script.

Table 1. Third Party Libraries

Library	Property	Example
FLTK 1.1.5rc2 (w/ NCBI patches)	pFLTKRootFolder	"usr:local:src:fltk-1.1.5rc2:"
BerkeleyDB 4.3.21	pBdbRootFolder	"Users:myhome:mylibs:db-4.3.21"
SQLite 2.8.13	gSqliteFolder	

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

Library	Property	Example
dlcompat	pDLRootFolder	"usr:"

You do not have to build the FLTK or BDB libraries separately. This is done by the scripts and CodeWarrior along with the toolkit libraries. Just unpack the source bundles in your home directory or where ever you have specified in the appropriate properties. The root folders for FLTK and BDB do not have to have any particular names. If there is more than one version the scripts will grab whichever is last alphabetically (e.g. fltk-1.1.4r2 will get used instead of fltk-1.1.3).

The scripts normally halt on any CodeWarrior compilation errors. If you want them to continue and save errors, set the next script property, pSaveContinueOnErrors, to true. Compilation errors for a project will be saved in a file in the same folder as the project being built, with a name in the following format: projectName-targetNumber.errs (e.g. xncbi-2.errs).

The Genome Workbench's configuration file(s) is stored in the user's Library:Application Support:gbench folder.

CFM builds are not supported. OS 8 or 9 are not supported. We know 10.2 works. We think 10.1 still works, and 10.3 might work.

Platforms (OS's, compilers used inside NCBI)

This release was successfully tested on at least the following platforms -- but may also work on other platforms. Since the previous release, some platforms were dropped from this list, just because we do not use them here anymore, and some were added (these new platforms are highlighted using bold font). Also, it can happen that some projects would not work (or even compile) in the absence of 3rd-party packages, or with older or newer versions of such packages -- in these cases, just skipping such projects (e.g. using flag "-k" for make on UNIX), can get you through.

Unix

Table 2. Unix OS's and Supported Compilers

Operating System	Architecture	Compilers
Linux-2.4.23 (w/ LIBC 2**.3.2**)	INTEL	GCC 2.95.3, 3.0.4, 3.4.0
Linux-2.4.26 (w/ LIBC 2.3.2)	INTEL	GCC 3.4.0
Linux-2.6.7 (w/ LIBC 2.3.3)	INTEL/64	GCC 3.4**.3**
Linux-2.4.23 (w/ LIBC 2.2.5)	INTEL	ICC 8.0
Solaris-8	SPARC	WorkShop 6 update 2 C++ 5.3 Patch 111685-13 (64-bit, 32-bit)
Solaris-8	SPARC	GCC 3**.4.3**
Solaris-9	INTEL	WorkShop 6 update 2 C++ 5.3 Patch 111685-13
Solaris-9	INTEL	GCC 3**.4.3**
IRIX64-6.5	SGI-Mips	MIPSpro 7.3.1**.3**m (64-bit, 32-bit)
FreeBSD-4.10	INTEL	GCC 3.4.2
Tru64 (OSF1) V5.1	ALPHA	GCC 3.3.2

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

Operating System	Architecture	Compilers
Tru64 (OSF1) V5.1	ALPHA	Compaq C++ V6.5-014

MS Windows

Table 3. MS Windows and Supported Compilers

Operating System	Compilers
MS Windows	MSVC++ 7.1. See documentation on MS Visual C++.NET.

Mac OS X

Table 4. Mac OS, and Supported Compilers

Operating System	Compilers
MacOS 10.2, 10.3	GCC 3.3
MacOS 10.3	CodeWarrior 9.2

Caveats and Hints

MacOS 10.X / CodeWarrior 9.2

1. Not all of the test or demo applications are built.
2. The source code for the latest release of FLTK (1.1.x), BerkeleyDB (4.x) and SQLite (2.x) should be present. See the installation instructions for details.

MacOS 10.2/GCC 3.3

At least the GCC 3.3 update for Dec. 2002 Developers Tools required from Apple.

GCC 2.95

1. Poor MT-safety record.
2. Relatively incomplete/incorrect (comparing to modern compilers) STL implementation.
3. It is going to be deprecated in NCBI rather soon -- as soon as we have any significant trouble with its maintenance.

GCC 3.0.4

1. Destructor of constructed class member is not called when exception is thrown from a method called from class constructor body (fixed in GCC 3.3).
2. STL stream uses locale in thread unsafe way which may result to segmentation fault when run in multithread mode (fixed in GCC 3.3).
3. Long-file support for C++ streams is disabled/broken (first broken in 3.0, fixed in 3.4).

GCC 3.3

Other than the feature described below, GCC 3.3.2 had been very good to us; it had a lot of very ugly bugs finally fixed.

1. Painfully slow linking in debug mode on Linux with GCC-3.3 compiler. -- Starting with binutils 2.12 linker tries to merge constant/debug strings marked for merging in

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

object files. But it seems it does this job very inefficiently - I've seen messages about it in internet. GCC starting with version 3.2 marks section of string constants ready for merging, and also has an option to disable this flag in object files (-fno-merge-constants). Adding this flag to compilation stage allows to avoid slow linking. GCC 3.3 also sets merge flag for debug sections and unfortunately there is no option to disable this flag. As a result, linking of debug executables significatly slower than with gcc 3.0.4. The slowdown rate depends on size of debug strings section and it's non-linear, so bigger projects will suffer more of this bug (N^2). Binutils 2.15 fixes this. The link time still 2 times slower than without symbol merge, but the resultant executable is about two times smaller in size, and no compiler patching is necessary. We are still testing it in-house. We had to patch GCC 3.3 in-house with the fix described at http://lists.boost.org/MailArchives/boost/msg53004.php.

2. Long-file support still broken (first broken in 3.1).

GCC 3.4

1. The "Painfully slow linking..." (see GCC3.3 [1] above) was an issue, and we had to patch it in-house to speed up, a la GCC 3.3 -- until we finally upgraded to binutils 2.15.
2. At least on Linux, ifstream::readsome() does not always work for large files, as it calls an ioctl that doesn't work properly for large files.
3. At least on Linux, GCC 3.4.[0,1] optimizer (very rarely) generates incorrect code when comparing enumerated values in else-ifs. (Fixed in 3.4.2)

Last Updated

This section last updated on March 9, 2005.