The NCBI C++ Toolkit

Release Notes (April, 2005)

Created: April 1, 2005.
Last Update: April 27, 2005.

Download Location

ftp://ftp.ncbi.nih.gov/toolbox/ncbi_tools++/2005/Apr_22_2005/

Source Archive Contents

Source Code Archives

- ncbi_cxx_unix--Apr_22_2005.tar.gz -- for UNIX'es (see the list of UNIX flavors below) and MacOSX/GCC
- ncbi_cxx_unix-- Apr_22_2005.gtar.gz -- for UNIX'es (see the list of UNIX flavors below) and MacOSX/GCC
- ncbi_cxx_win-- Apr_22_2005.exe -- for MS-Windows / MSVC++ 7.1 (self-extracting)
- ncbi_cxx_win-- Apr_22_2005.zip -- for MS-Windows / MSVC++ 7.1
- ncbi_cxx_mac_cw-- Apr_22_2005.tgz -- for MacOS 10.3.4 / CodeWarrior DevStudio for MacOS 9.2
- ncbi_cxx_mac_xcode-- Apr_22_2005.tgz -- for MacOS 10.3.4 / xCode 1.[1-5]

The sources correspond to the NCBI production tree sources from patch "RELEASE_APR_2005", which in turn roughly corresponds to the development tree sources from the very end of March, 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. Modified CNcbiApplication class to add yet another command line flag - to distinguish between "short" and "detailed" help printout. The detailed help is printed only upon request.
2. CDll - added 'auto_unload' parameter to constructors to be able to unload DLL in a destructor.
3. CPluginManager_DllResolver - added 'unload_dll' parameter to a constructor to be able to unload driver DLLs in the destructor.
4. CPluginManager - CPluginManager won't unload its drivers by default. All drivers that used to be unloaded previously kept old behavior.
5. File API (CDirEntry, CFile, CDir) -- dropped native (non-UNIX) MacOS 9 support.

6 CDirEntry added methods:

- CreateObject() -- construct CDirEntry based object of specified type (file, dir, symlink).
- operator= -- assignment operator.
- Copy() -- copy one dir entry to another (should have equal type).
- CopyToDir() -- copy dir entry to to some directory.
- SetBackupSuffix() -- set suffix used to create backup copies of dir entries.
- GetBackupSuffix() -- get current backup suffix.
- Backup() -- create backup copy for dir entry.
- IsLink() -- check if directory entry a symbolic link.
- LookupLink() -- get entry name that link is pointed to.
- DereferenceLink() -- dereference symbolic link, replace current object entry name with entry name that link is pointed to
- IsNewer() -- check if current entry is newer than some other.
- GetOwner() -- get owner of dir entry.
- SetOwner() -- set new owner for dir entry.
- Get/SetTimeT() -- get/set dir entry time using time_t.

7 CDirEntry -- added CMask versions of MatchesMask(), GetEntries(), FindFiles() and FindFilesInDir() methods.

8 CDirEntry::Rename() -- added flags parameter.

9 CFile -- added new methods:

- Copy() -- copy one file to another.
- Compare() -- compare file's content.

10 CDir::Copy() -- new method to copy directories.

11 CDir::GetEntries(), CDirEntry::MatchesMask() - added parameter for case sensitive/insensitive matching.

12 CSymLink -- new class to work with symbolic links (UNIX specific).

13 SuppressSystemMessageBox() -- new function to suppress popup messages on execution errors in both runtime and in debug libraries, as well as all General Protection Fault messages. (Windows specific).

14 CDllResolver::AddExtraDllPath() -- added support of multiply paths in the NCBI runpath.

15 NStr::MatchesMask() -- added parameter for case sensitive/insensitive matching.

16 CTime::IsValid() -- added leapsecond support.

17 CTimeSpan::IsEmpty() -- new method to check if time span is empty.

18 CFastLocalTime -- new class for support quick-and-dirty getter of local time.

19 CNcbiRegistry -- support taking settings from the environment.

20 Added support for safe boolean operators to prevent their accidental use in non-boolean context.

21 CPluginManager now keeps track of registered entry points.

22 CPluginManager: add lib prefix to dll name mask.

23. Configuration files: added possibility to include sections. Merge subnodes and values having the same id.
24. New CProcess::GetParentPid() -- to retrieve parent's PID on UNIX/Windows.

CONNECT -- Data streaming, Networking, and Dispatching

1. New BASE64_{En|De}code API to perform BASE64 (RFC1521) encoding and decoding.
2. ConnNetInfo_Create() to always set SConnNetInfo::user_header from environment/ registry key CONN_HTTP_USER_HEADER.
3. New method CConn_IOStream::Close() for forced stream closure.
4. HTTP_CreateConnector() is now overriding additional user header instead of formerly setting it (thus the value is able to get preserved if it is not conflicting with the overriding one).
5. Major bug fix in "\r\n" parsing in both header and body.

UTIL -- Miscellaneous Low-Level APIs

1. CRequestRateControl -- new class to cap the rate of requests to a resource.
2. CTar -- new class for basic TAR archive support.
3. Implemented CIStreamBuffer::HasMore().

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

1. Implemented serialization of XML elements with mixed content.
2. CClassPrePostReadWrite -- implements PreRead and PostWrite serialization hooks.

DATATOOL -- Code Generator and Data Converter Tool

1. Implemented XML elements with mixed content in DTD parsing and writing.
2. Modified DATATOOL to produce correct output when converting DTD to DTD.
3. Corrected generation of XML schema: fixed syntax errors, corrected definition of default values, fixed "real" data type (changed "decimal" to "double").
4. Corrected C++ code generation to correctly handle custom data types for ASN data of ENUMERATED type.

CGI -- CGI and Fast-CGI Application Framework

1. Major fixes in the use of CArgs argument description mechanism for CGI applications.

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

1. Added cursor reopen operation.
2. Added PrintStat method to output BTREE statistics.

DBAPI -- Generic SQL Database Connectivity

1. Database driver manager revamped to use "core" CPluginManager.
2. Added a new method I_DriverMgr::AddDllSearchPath to be able to get I_DriverContext from a TPluginManagerParamTree.

3. Added a new method I_DriverMgr::GetDriverContextFromTree to allow adding a driver search path for the driver manager.
4. Developed a unit-test for DBAPI.
5. An initial version of a Python DBAPI module was released.
6. Renamed bdb_cache and dbapi_cache libs to ncbi_xcache_*.
7. CDB_Connection::Abort() -- new method to severe a connection to database on socket level (so there are no post-mortem communications with the database server and related risks of network hanging).

ALGO/ALIGN -- Generic Alignment Algorithms

1. CSeq_id -- support several new prefixes in IdentifyAccession() method.
2. FASTA reader -- fix handling of non-IUPAC residues, runs of hyphens, and semicolon-started comments.
3. The formatting classes are now using CSeq_id objects instead of plain strings to represent sequences IDs.
4. CNWAligner::SetTranscript() -- new method to simulate the alignment.

BLAST

1. Introduction of structures to represent the filtering string in CORE BLAST.
2. BlastSeqSrc implementations clean up (API BLAST).
3. Addition of composition-based statistics for PSI-BLAST (code to perform IMPALA-style scaling of PSSMs in blast_posit.[ch]).
4. Introduction of interval trees, a binary tree data structure that greatly speeds up the search for alignments that envelop/contain other alignments. Used in the preliminary gapped alignment phase of all BLAST programs and in the traceback phase for all programs except MEGABLAST.
5. Improvements in speed and sensitivity to DUST (blast_dust.c).
6. Changes to how score matrices are stored in CORE BLAST (SBlastScoreMatrix).
7. Major refactoring of greedy alignment routines, minor refactoring of in-frame and out-of-frame gapped alignments routines (in-frame and out-of-frame gapped alignment outines now allocate memory for auxiliary structures dynamically, instead of allocating the worst case amount of memory when initializing).
8. Change in tie-breakers for score comparisons.
9. Rewrite of uneven gap HSP linking.
10. Unification of traceback generation functions (memory usage has been greatly reduced).

ALNMGR -- Bio-sequence Alignment Manager

1. CAlnMap -- added a print class CAlnMapPrinter.
2. CAlnVec -- added a print class CAlnVecPrinter.
3. CAlnMix -- rearranged into a set of classes with a preserved CAlnMix front-end interface. The new classes are:
    - **a** CAlnMixMatches -- container for CAlnMixMatch plus methods,
    - **b** CAlnMixSegments -- container for CAlnMixSegment plus methods,

- **c** CAlnMixSequences -- container for CAlnMixSeq plus methods,
- **d** CAlnMixMerger -- the merging algorithm.

4 CAlnMix -- abstracted the CalculateScore method so that it could be delegated to the caller. Added handling for best reciprocal hits.

5 CAlnMrg -- added a sortseqbyscore flag, extended to reading multiple dense-segs, and added handling for best reciprocal hits.

6 CAlnVwr::GetAlnPosFromSeqPosDemo() -- new method.

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

1 CSeq_id -- support several new prefixes in IdentifyAccession() method.

2 FASTA reader -- fix handling of non-IUPAC residues, runs of hyphens, and semicolon-started comments.

3 Added possibility to use GetSeq_idByType() with containers of CSeq_id_Handle.

4 Preserve fuzz when merging/adding seq-locs in seq-loc operations.

5 Sequence::GetId() now returns CSeq_id_Handle.

ALGO/PHY_TREE -- Phylogenetic and bio tree algorithms

1 Minor bug fixes and improvements.

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

2 Implemented setters for 'descr' field of CBioseq_Handle.

3 Implemented CBioseq_EditHandle AddId() & RemoveId().

4 Added methods to get feature type and subtype.

5 Added getter for feature type and subtype to CMappedFeat.

6 SSNP_Info structure is defined in separate header to reduce dependencies.

7 Added "SNP" symbols to names of CSeq_feat_Handle methods used to access SNP table.

8 Added CScope::RemoveFromHistory(), CScope::RemoveTopLevelSeqEntry() and CScope::RemoveDataLoader().

9 Added state flags to CBioseq_Handle. Report conflicts using state flag, do not throw exception.

10 Added CBioseq_Handle::ContainsSegment with optional depth and limit object.

11 CBioseq_Handle::RemoveDesc() -- now returns CRef<CSeqdesc> (was bool).

12 Added SAnnotSelector(TFeatSubtype). Added flag to SAnnotSelector for skipping multiple SNPs from the same seq-annot.

13 CSeq_loc_Mapper -- added direction flag for mapping between top level sequence and segments.

14 Iterators: ignore e_Locs annotations with unknown format (used to fail before).

15 CBioseq_CI iterator with CSeq_inst::eMol_na now includes both dna and rna bioseqs. Added constructor accepting CBioseq_set_Handle.

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

1 Added auto cleaning of indexes with id resolution information.

2 Commented out obsolete classes CRefresher, CMutexPool, and CGBLGuard.

3. GenBank data loader now has new schema of readers/writers.
4. It's now possible to use GenBank loader cache by specifying it in registry, e.g. "[GENBANK] ReaderName = cache;id1", or via environment "GENBANK_LOADER_METHOD=cache;id1".

OM++ TEST programs

1. Added few tests for loading split data from ID2 server.

OM++ DEMO program (objmgr_demo)

1. Added flag -print_mapper to check CSeq_loc_Mapper.
2. Added scan of all features in TSE when -whole_tse is specified.
3. Added option to allow loading Seq-annot from file.

ID2

1. Added split-version to ID2S-Request-Get-Chunks structure of ID2 protocol.
2. Allow storing skeleton Seq-entry together with split info in ID2.

BUILD FRAMEWORK (MSVC++.NET)

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

1. Now can keep track of subproject types (expendable, potential) and propagate it down the project tree, and to generate MSVC projects for all subprojects, not only for those listed in Makefile.in.

3RD-PARTY PACKAGES

1. CPPUNIT was added to 3rd-party packages. To use CPPUNIT just add REQUIRES = CPPUNIT to your Makefile.in, $(CPPUNIT_INCLUDE) to CPPFLAGS and $ (CPPUNIT_LIBS) to LIBS respectively in your app file.

PYTHON -- Scripting Language Support

1. An initial version of a Python DBAPI module was released.

FRAMEWORKS

A beta version of grid computing components, including:

1. NetCache -- daemon for network-distributed data cache,
2. NetSchedule -- daemon to support network-distributed request queues,
3. GridWorker -- API to create grid computing nodes,
4. Cgi_Tunnel2Grid -- CGI to tunnel request to GridWorkers.

Documentation

Document Location

The documentation is available online at http://www.ncbi.nlm.nih.gov/books/bv.fcgi?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 with CodeWarrior 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.

On running the scripts you will be prompted as to which targets you want to build. Or if you always build the same targets they can be specified 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: Debug and Final. For example, to build only the Debug targets use: "Build Debug", to build both debug and release (final) versions: "Build".

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 1.1.6 (w/ NCBI patches)	pFLTKRootFolder	" home:mhome:fltk-1.1.6-ncbi3"
BerkeleyDB 4.2 (or 4.3)	pBdbRootFolder	"Users:myhome:mylibs:db-4.2.52" (or "... db-4.3.21")
SQLite 2.8.13	gSqliteFolder	

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 10.3 works. We think 10.1 still works, and 10.2 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 3.4.0, 3.0.4, 2.95.3
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-8,9	SPARC	WorkShop 6 update 2 C++ 5.3 Patch 111685-21 (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 7.3.1.3m (64-bit, 32-bit)
FreeBSD-4.10	INTEL	GCC 3.4.2
Tru64 (OSF1) V5.1	ALPHA	GCC 3.3.2
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 10.3	GCC 3.3
MacOS 10.3	CodeWarrior 9.2
MacOS 10.3	xCode 1.5

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 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.

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 April 27, 2005.