# The NCBI C++ Toolkit

## Release Notes (March, 2007)

Created: March 12, 2007.
Last Update: March 27, 2007.

---

- GCC 3.3
- GCC 3.4.x, 4.0.x

## Download

Download the source code archives at:

ftp://ftp.ncbi.nih.gov/toolbox/ncbi_tools++/2007/Mar_12_2007/

- ncbi_cxx--Mar_12_2007.tar.gz — for UNIX'es (see the list of UNIX flavors below) and MacOSX
- ncbi_cxx--Mar_12_2007.gtar.gz — for UNIX'es (see the list of UNIX flavors below) and MacOSX
- ncbi_cxx--Mar_12_2007.exe — for MS-Windows (32- and 64-bit) / MSVC++ (7.1, 8.0) — self-extracting
- ncbi_cxx--Mar_12_2007.zip — for MS-Windows (32- and 64-bit) / MSVC++ (7.1, 8.0)

The sources correspond to the NCBI production tree sources, which in turn roughly corresponds to the development tree sources from February 16, 2007.

There are also two sub-directories, containing easily buildable source distributives of the NCBI C Toolkit (for MS Windows and UNIX) and selected 3rd-party packages (for MS Windows only). These are the versions that the NCBI C++ Toolkit should build with. For build instructions, see README files there:

- NCBI_C_Toolkit
- ThirdParty

## Build

For guidelines to configure, build and install the Toolkit see here.

## New Developments

### CORELIB — Portability and Application Framework

1. Redesigned diagnostics setup and introduced new logging info and formatting (it is not turned on by default yet).
2. Enabled per-thread diagnostic properties (request status, client IP, session ID etc.).
3. Diag handlers now support atomic write to allow several applications to use the same log file.
4. Added CStackTrace class to get/store stack trace information.
5. The stack trace is automatically reported by exceptions.
6. FindFiles2<> — new algorithm for file search.
7. NStr::ReplaceInPlace(), NStr::HexChar() — new string functions.
8. CStringUTF8 — added character buffer validation methods.
9. XStr — added template-based string comparison, one that does not depend on the character type.
10. CTempString — provided with the clear() and assign() methods to better mimic "std::basic_string"

11. CNcbiApplication — now loads a global configuration file (.ncbirc or ncbi.ini) in addition to any application-specific configuration.
12. CSysLog — obtains the default facility to use from registry settings.
13. CDirEntry::CheckAccess() — to check effective access rights on directory entries.
14. CNcbiApplication — added option "-dryrun" to allow test runs of an application, just to verify if the command line is correct and all preconditions are met.
15. NcbiSystem::GetMemoryUsage — new function to determine the current process's approximate memory footprint.
16. CRWStreambuf — to allocate buffers per I/O direction; fixed write bug and implemented xsputn() to speed up writing; fixed showmanyc() not to loop with zero timeouts.
17. NcbiStreamCopy() — to directly copy contents of one stream into another.
18. Fix printable string and parse escapes to better match the C/C++ string representation standard.
19. "include/corelib/mswin_no_popup.h" — header file to disable popup message boxes on MS Windows. Also, SuppressSystemMessageBox() now can use new flag fSuppress_Exception (unhandled exceptions).

## CONNECT — Data streaming, Networking, and Dispatching

1. CServer — a replacement for the CThreadedServer. This multithreaded network server framework allows many permanent connections without the burden of having as many processing threads.
2. CPipe::ExecWait() — to run an external application, pass a stream with input data to it, wait and monitor its execution, and get back its standard output and error streams.
3. SOCK_isip(), SOCK_isipEx(), SOCK_GetLocalHostAddress().
4. HTTP connector — to make path and arguments non-inheritable in redirects; to allow relative redirects (not argument-only, however); and to take into account Content-Length (and fail if not enough data received).
5. FTP connector — to use minus sign in default password to try to turn off human-readable messages; also, fixed the handling of low-level status codes.
6. Reorganization of the heap manager API:
7. Faster heap with free blocks linked into a list; rename of HEAP_AttachEx() into HEAP_AttachFast(); new API calls HEAP_AllocFast(), HEAP_FreeFast(), and HEAP_Options().
8. CConnIniter (former CONNECT_InitInternal) — new helper class to simplify the initialization of the library (and to reduce the chance of that been
9. forgotten) without a necessity of any changes in the existing user code.
10. Most CONNECT related APIs have been modified to take advantage of the initer.

## UTIL — Miscellaneous Low-Level APIs

1. Compression API — Added automatic finalization for input streams.
2. Also, added an auto-detection and proper reading of uncompressed data (use flag CCompression::fAllowTransparentRead).
3. CCompression*Stream::GetStatus() — new, to get status of the last compression/ decompression stream operation.

4 CChunkStreamWriter, CChunkStreamReader — new classes for serialization and deserialization of interleaved sequences of control symbols and chunks of binary data. These two classes are primarily intended for use with asynchronous stream readers/ writers.

5 CMaskRegexp — new regexp based class to match string against set of masks.

6 CIStreamBuffer — now can be built directly on a memory buffer.

7 ILineReader — now can construct a line reader on top of a file thus utilizing more efficient (mmap-based) implementations.

8 Added CIReaderLineReader (line readed based on IReader interface)

9 CFormatGuess - improved format prediction, fixed stream positioning.

10 CResourcePool - added locking traits as a template parameter to provide syncronization where it is needed.

11 Library "xqueryparse" — a lexer/parser of query language, to parse PubMed queries and/or limited SQL SELECT subset into a query tree, for further interpretation.

12 CTimeLine — an approximate time tracker that can track millions of uniquely numbered objects. Used for timeout control and time based garbage collection.

13 CTar — fixed bug in the splitting of long filenames; added getters for the last access and creation; now can restore the original atime (and virtually ctime) for old GNU formats.

## SERIAL — Data Serialization (ASN.1, XML, XML Schema)

1 CObjectIStream — added option to skip unknown choice variants.

2 Restructured and documented SERIAL library headers.

3 "app/sample/asn/" — a sample project showing how to generate source code from an ASN.1 specification (see "sample_asn.asn") and then use that code to read, convert and write data matching that specification.

## DATATOOL — Code Generator and Data Converter Tool

1 Modified XML Schema parser to preserve comments found in the Schema specification. Enhanced parsing of comments in ASN.1 specification.

2 Enhanced datatool to use data specification comments in generated C++ code.

3 Added option to convert data specification into an easily parsable format (e.g. to be fed to the source browsers).

## CGI — CGI and Fast-CGI Application Framework

1 CCgiUserAgent::GetPlatform() — added.

2 CGI applications can produce multipart responses and suggest destination filenames.

3 Added a cookie to track the progress of Web sessions.

4 Throw an exception if the output CGI stream goes bad.

## HTML — HTML Generation Library

1 Added support for XHTML output format.

2 Added OPTGROUP tag support.

3 CPopupMenu — upgraded Sergey Kurdin's popup menu to v2.7.

4 CHTMLHelper — added HTMLDecode method to decode HTML entities and character references.

5 CHTMLHelper::HTMLAttributeEncode() — new method to encode HTML tags attributes.

## BDB — Yet Another C++ API Based On BerkeleyDB

1 Implemented multi-row fetch for cursors. Works much faster than regular fetch.

2 Added BLOB read into a resizable buffer (e.g. std::vector).

3 Split store now can free some unused memory (necessary for use in long uptime services).

## DBAPI — SQL Database Connectivity

1 FreeTDS v64 based drivers (CTLIB, ODBC and ODBCW flavors) added.

2 Authentication protocol NTLMv2 was implemented and enabled with

3 ftds64 and ftds64_odbc drivers; NTLMv1 was disabled.

4 CDB_Exception::{Get/Set}SybaseSeverity(), CDB_RPCCmd::GetProcName() — new.

5 Posting of a warning in case of a string truncation was added to DBAPI.

6 CWString and CDB_String — new classes, to help handle wide char strings.

7 ODBC based drivers — the timeout feature implemented.

8 CVariant — semantic changed so that CVariant constructed on an empty time assumes a "null" value.

9 Integrate a local fallback copy of UnixODBC for the sake of FreeTDS v64.

10 CTLIB driver — now uses TDS protocol 12.5 (was 11.0) by default.

## ALGO — Generic Algorithms

1 Added volume merge algorithm - a variant of merge sort for external storage. This particular implementation can use storage concurrency and work efficiently on multiple threads.

## ALGO/ALIGN/SPLIGN — Spliced and Generic Alignment Algorithms

1 CBandAligner (banded spliced aligner) — no longer requires the band to be centered at the main diagonal.

2 CCompartmentFinder (compartmentization algorithm) — the maximum intron length has been set to one megabase to support extra-long introns. A command line demo utility 'compart' has been added.

3 CHitFilter (greedy alignment reconciliation) — new method s_MergeAbutting() to merge hits abutting on either of the sequences.

4 CSplignFormatter (splign alignment formatter) — redesigned to produce a more readable and informative text alignment view featuring coding region translations.

## ALNMGR — Bio-sequence Alignment Manager

1 CAlnMerger: Fixed logic when fAllowTranslocation is used. Added early detection of failures to merge.

2 Introducing a new approach to alignment management. It comprises of a collection of classes and functions designed to provide flexibility, simplicity and efficiency. The components are independent of the original alignment manager and where possible, independent of each other. While this is still a work in progress in alpha stage, the new code is usable and already solves a number of problems.

3 For example of usage see: objtools/alnmgr/demo/aln_build_app.cpp.

4 List of new entities: CAlnAsnReader, MergeAlnRngColl, BuildAln, CAlnContainer, ConvertSeqAlignToPairwiseAln, ConvertDensegToPairwiseAln, ConvertStdsegToPairwiseAln, ConvertDendiagToPairwiseAln, ConvertSparseToPairwiseAl, CreateAnchoredAlnFromAln, CreateAnchoredAlnVec, IAlnExplorer, IAlnSegment, IAlnSegmentIterator, CNonDiagFilter, CreateSeqAlignFromAnchoredAln, CreateDensegFromAnchoredAln, CreateDensegFromPairwiseAln, SubtractAlnRngCollections, SubtractOnFirst, SubtractOnSecond, IAlnSeqId, TAlnSeqIdIR, CAlnSeqId, CAlnStats, CAlnIdMap, CAlnUserOptions, CAlnValidate

## BLAST

1 CDbBlast -deprecated, superceded by CLocalBlast class.

2 Major redesign of the low-level nucleotide BLAST routines. blastn and megablast should be significantly faster for smaller queries.

3 Lookup table improvements in CORE BLAST, optimized ScanSubject routines, compressed query support.

4 Added functionality to use the Smith-Waterman algorithm instead of the BLAST algorithm.

5 Added blastinput library to read FASTA files into BLAST input.

6 Added dbindex library to enable megablast search against a pre-indexed database. This can speed up searches in case of short (<100Kbases) queries against a large fixed database.

7 'makeindex' program located in dbindex/makeindex/ can be used to create indices from either fasta files or directly from BLAST databases.

8 Added HSP range support, an interface to fetch parts of nucleotide sequences which improves performance in the traceback stage.

9 Use CBlast4Field instead of hard coded strings in blast::CRemoteBlast.

10 Implemented feature to ignore strand blast::SSeqLoc::mask.

11 Preliminary alignment no longer computes traceback under any circumstances.

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

1 CSeq_align, CDense_diag, CStd_seg — allow negative offsets in OffsetRow

2 Sparse-seg, Sparse-align — use vector (instead of list) container

3 CSeq_id::IdentifyAccession() — recognizes more prefixes (AC_, EF-EL, plus internal genome pipeline prefixes) and (mixed-in) EMBL TPA protein accessions, and supports automatically loading the data it needs from an external file.

4 CFastaReader — handles excerpts as new local sequences with appropriate history rather than gappy versions of their parents.

5 CFastaReader — stores deflines in user descriptors for reference.

6 CBioseq::PackAsDeltaSeq(), CDelta_ext::AddAndSplit() — new methods (powered by new code in library sequtil) to pack nucleotide sequences with occasional ambiguities in multiple pieces for better overall efficiency.

7 Improved seq-id matching rules in seq-id mapper.

8 CSeq_loc_Mapper_Base, CSeq_align_Mapper_Base — new classes which can be used without creating ObjectManager.

## LDS — Local Data Storage

1 CLDS_Manager — major redesign.

2 lds_indexer — new utility to create or update index files for directories with FASTA or ASN sequences.

3 Added indexing of file names

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

New functionality:

1 CBioseq_EditHandle(CBioseq_Handle) made public and explicit to avoid unnecessary GetEditHandle() calls.

2 Introduced CSeq_feat_EditHandle.

3 Introduced CSeq_annot_ftable_CI and CSeq_annot_ftable_I.

4 Implemented TakeFeat() like methods.

5 Implemented user API to Update() annotation index.

6 Added RemoveTopLevelEntry(CSeq_entry_Handle).

7 AddXxx() and GetXxxHandle() accept extra control argument EMissing/EExist.

8 Added more exception types.

9 Added functions to calculate sequence map switch points.

10 Recognize Seq-data.gap.

11 Added feature ids index and retrieval.

12 Added information about local feat-ids to ID2 split specification.

13 Added generation of feature ids information in ID2 blob splitter.

14 Use feature lookup by feat-id in GetBestXxxForXxx().

15 Implemented seq-map switch editing.

16 Implemented SAnnotSelector::Reset*NamedAnnots().

17 Accept SNP quality data in OCTET STRING format.

18 Implemented full CSeqMap editing API.

Bug fixes:

1 Set name of Seq-annot only if desc.name is present.

2 MT-Safe management of the CSeqportUtil singleton.

3 Fixed removing top level entries.

4 Fixed killing threads of prefetch manager.

5 Fixed order of multi-id annotations.

6 Store annots indexing information to allow reindexing after modification.

7 Avoid feature mapping dependency on memory placement.

8. Disable multi-conversion of SNPs.
9. Added check for self-references in CSeqMap_CI.
10. Fixed GetRight() for single-strand CHandleRange.
11. PDB chain in Seq-id is case sensitive.
12. Fixed lost CSeq_data in split sequences.
13. Preserve original type of Seq-inst after editing if possible.
14. Fixed incorrect usage of CSeqMap::x_GetSegmentsCount().
15. Clean Seq-id cache on edit start.
16. Automatically reset 'annot' field if it's empty.

RemapAlignToLoc(), CSeq_align::RemapToLoc() — new.

## OM++ LOADERS/READERS — Data Retrieval Libraries for OM++

*(GenBank data loader)*

New functionality:

1. Added parsing of feature ids information.
2. Allow unknown members/variants in split info.

Bug fixes:

1. Properly mark withdrawn and confidential records.
2. Limit number of simultaneously requested blobs to avoid hangs with PIG ids.
3. Detect 'suppressed' state.
4. Fixed exclude-blobs list.

## BUILD FRAMEWORK (UNIX)

1. $(BLAST_LIBS), $(BLAST_FORMATTER_LIBS) — new macros to isolate users from BLAST's internal dependency changes.
2. $(DLL_UNDEF_FLAGS) — new variable, normally set to $(ALLOW_UNDEF); may alter to $(FORBID_UNDEF) to request strict dependency checking.

## PTB — Project Tree Builder for MSVC++ .NET

1. Added option to allow setting linker additional dependencies and additional library directries in MSVC tune-up files
2. LST-file filter now can recognize regular expressions in the directory names.
3. Enable configurations when:

3rd party library is absent, but not required

libraries with a choice where one of the choices is empty.

4. Added option to automate tweaking the toolkit tree for use with VTune.
5. Optimized to speed up the configuration on MSVC8.0.

## APPLICATIONS

1. NetCache
    - Added support of in-memory logs
2. NetSchedule

- Moved to new thread-per-request server framework (based on CServer)
- Implemented fast reply of idle worker node cluster. Important for the CGI request processing.
- Implemented a dynamic creation/deletion of queues.
- Implemented tags for job grouping and reporting.

3. ASN2ASN
   - Allow conversion of multiple entries in file.

### GRID (DISTRIBUTED COMPUTING) FRAMEWORK

#### *Grid Worker Node Implementation Framework*

1. Added "auto_shutdown_if_idle" parameter to worker nodes'
2. configuration files. It allows shutting down a worker node automatically if it is idle for some period of time.
3. Added option to perform a monitor script from jobs running through "remote_app" and "remote_cgi" utilities. This script allows getting a job's progress execution and making a decision if the job should be terminated.
4. Redesigned and re-implemented low-level NetSchedule client API. It allows making permanent connections to the load-balanced NetSchedule servers.
5. Moved all Grid Framework APIs and utilities to the new low-level NetSchedule API.
6. Added support from jobs' tags submission to "ns_submit_remote_job" utility.

## Documentation

### Location

The documentation is available online as a searchable book "The NCBI C++ Toolkit": .

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.

### 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/

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 up-to-date public or in-house versions using source browsers Entrez, LXR, Doxygen and Library.

**HEADS-UP:** We have switched our source control system from CVS to SVN (Subversion). Unfortunately, the SVN repository cannot (yet) be accessed from outside NCBI. The CVS code repository now contains older version of the source trees (before mid-January 2007) but it still can be accessed via a Web interface (see the sidebar box on each page of the C++ Toolkit Book).

## 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 inhouse 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-2.6.x (w/ LIBC 2.3.2 , 2.3.5 ) | x86-32 | GCC 3.4.0<br>ICC 8.0 (build 20040520Z, package ID l_cc_pc_8.0.066_pe067.1)<br>(GCC 3.0.4, 2.95.3, 3.4.2, 4.1.1- nominal support) |
| Linux-2.6.x (w/ LIBC 2.3.3 , 2.3.5 ) | x86-64 | GCC 4.0.1ICC 9.0 (build 20051201)<br>(GCC 4.1.1 - nominal support) |
| Solaris-8<br>**[Support will be discontinued in the next release!]** | SPARC | C++ 5.3 (WorkShop 6u2) patch 111685-**24**<br>(64-, 32-bit) (GCC 3.4.3 - nominal support) |
| Solaris-10 | SPARC | Sun C++ 5.5 (Studio8) patch 113817-**19**<br>GCC 4.0.1 (32-bit mode only) |
| Solaris-9 | x86-32 | C++ 5.3 (WorkShop 6u2) patch 111686-13<br>GCC 3.4.3 |
| Solaris-10 | x86-32 | Sun C++ 5.5 (Studio8) patch 113819-**19**<br>GCC 4.1.1 |
| IRIX64-6.5 | SGI-Mips | MIPSpro 7.3.1.3m (64-bit, 32-bit) |
| FreeBSD-4.10 | x86-32 | GCC 3.4.2 |
| FreeBSD-6.1 | x86-32 | **GCC 3.4.4** |
| Digital Tru64 Unix 5.1 (aka OSF1)<br>**[Support will be discontinued in the next release!]** | ALPHA | GCC 3.3.2 (limited support) |

### MS Windows

**Table 3. MS Windows and Supported Compilers**

| Operating System | Compilers |
|---|---|
| MS Windows-32 | MS Visual Studio .NET 2003 (C++ 7.1).NOTE: We also ship an easily buildable archive of 3rd-party packages (including NCBI C Toolkit) for this platform. |
| MS Windows-32 | MS Visual Studio .NET 2005 (C++ 8.0) |
| MS Windows-64 | MS Visual Studio .NET 2005 (C++ 8.0) |
| Cygwin 1.5.18 - 32 | GCC 3.4.4 |

### Mac OS X

**Table 4. Mac OS, and Supported Compilers**

| Operating System | Compilers |
|---|---|
| Darwin on MacOS X 10.4 ("Tiger") | GCC 4.0.1 |
| Darwin on MacOS X 10.4 ("Tiger") | Xcode 1.5 - 2.2.1 |

## Caveats and Hints

### 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 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 significantly 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.x, 4.0.x

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 (we didn't test whether 4.0.x fixed this).

**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)

**4** GCC 3.4.3, 3.4.4 (and maybe 3.4.5+) and 4.0 have a bug in the C++ stream library that affects some parts of our code, notably CGI framework. (Fixed in 4.0.1).

## Last Updated

This section last updated on March 27, 2007.