Release Notes (August, 2007)

Created: September 17, 2007.
Last Update: September 17, 2007.

Download

Download the source code archives at:

ftp://ftp.ncbi.nih.gov/toolbox/ncbi_tools++/2007/Aug_27_2007/

- ncbi_cxx--Aug_27_2007.tar.gz — for UNIX'es (see the list of UNIX flavors below) and MacOSX
- ncbi_cxx--Aug_27_2007.gtar.gz — for UNIX'es (see the list of UNIX flavors below) and MacOSX
- ncbi_cxx--Aug_27_2007.exe — for MS-Windows (32- and 64-bit) / MSVC++ (7.1, 8.0) — self-extracting
- ncbi_cxx--Aug_27_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 July 30, 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. NCBI_THROW*() macros — allow iostream-style message formatting.
2. AutoPtr<> — allow optional ownership of object pointer in AutoPtr<>; also, fixed possible double delete in AutoPtr::reset().
3. IReader — added implementations for the C++ istream and for the low level reading from file via system IO handle.
4. CDirEntry::CreateAbsolutePath() — new, to get an absoluite path from relative path.
5. CTime — new methods: MilliSecond(), GetMilliSecond(), Microsecond(), GetMicroSecond(), Round().
6. CTime — added more checks on incorrect using time objects stores empty dates.
7. CProcess::Wait() — now can return additional information about the waited process.
8. CExec — new methods: IsExecutable(), ResolvePath().
9. CProcess::Kill() — remove 'linger_timeout' parameter, used only on MS Windows. Now, use one timeout parameter as a sum of both timeouts.
10. CTime/CTimeSpan/CStopWatch — added support for different format types for the string conversions. Added new helper class CTimeFormat.
11. CTime::Truncate() — added precision parameter.
12. CFileLock — new, to lock a file (by file descriptor or file name) for reading or writing.

13 Added support for Byte Order Mark in plain text data streams.

14 It is now possible to check whether the stream has such mark or not (GetTextEncodingForm); or read all input text data from the stream and convert it into UTF8 string (ReadIntoUtf8).

15 CRWLock — now supports optionally blocking new readers when would-be writers are waiting, to avoid the possibility of starving them.

16 CException — now supports a polymorphic Throw() method that works for all classes derived via NCBI_EXCEPTION_DEFAULT and related macros.

17 CArgDescriptions — extended to allow dependencies between arguments (an argument can require or exclude another argument) and negated aliases for flag arguments (-flag vs -no-flag).

18 System mutex implementation on MS-Windows changed to use critical section rather than mutex.

19 CProcess::Fork() — new method, to update cached PID and other related variables after forking the process.

20 FindFiles() — added a glob-style functionality, which takes a string pattern and collects all matching files/directories.

21 StartTraceCollect() and StopTraceCollect() — new functions, to allow collecting trace messages without printing them immediately. The collected messages can be printed or discarded later (e.g. printed only if an error occurs).

22 Changed time format and added application state flag in the diagnostic messages.

23 ConvertRegToTree() — redesigned to synchronize path-style and SubNode-style definitions in INI-files.

24 CPushback_Streambuf::seekoff() — now supports tellg()

CONNECT — Data streaming, Networking, and Dispatching

1 CThreadedServer — now supports optionally reserving its port in advance.

2 CServer — added timer functionality, using IServer_ConnectionHandler::GetTimeout().

3 CSocketReaderWriter — implemented timeouts: {Get|Set}Timeout()

4 TRIGGER/CTrigger — a pollable object that can be used together with sockets

5 Implemented advanced (non-lingering) modes of close/abort of TCP sockets

6 CSocketAPI — added HostPortToString() and StringToHostPort()

UTIL — Miscellaneous Low-Level APIs

1 CIReaderLineReader — renamed to CBufferedLineReader, added buffering.

2 CMultiWriter — an implementation of IWriter interface which allows simultaneous writing to different streams

3 ILineReader::UngetLine() — allows for a full-line lookahead.

4 ICompression — new, interface class for CCompression.

5 Compression API — added LZO compression support.

6 CTimeLine — new class, for fast approximate time tracking.

7 CTar:

Better diagnostics

Handle hard links to the extent allowed by OS

Large file support added

Use flags to set case-sensitivity of the mask(s)

Relaxing requirements on native tar capabilities

Make sure normalized paths used throughout

Streaming extraction fully implemented

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

1. Corrected serialization of data in XML format to recognize UTF8 byte order mark, to read attributes of SET data type, to properly read XML containers with all optional data members. Corrected serialization of boolean attributes of data objects generated by XML specification.
2. SOAP library — implemented fault handling to comply with SOAP v1.1 specification.

DATATOOL — Code Generator and Data Converter Tool

1. Considerably enhanced XML schema parser by adding support for 'any' and 'all' element types, data type definitions with documentation, recursive type definitions, attribute groups and content model groups, mixed types, element defaults, type extensions, top level annotations and ability to import schemata.
2. Extended the C++ code generator to allow an UTF8 string as choice variant.

CGI — CGI and Fast-CGI Application Framework

1. CCgiUserAgent — added check on well-known search robots, bots, web checkers and link validators. Added IsBot() method.
2. CCgiRequest — now preserves the input stream for POST data with no Content-Type, which may be a BLOB rather than "application/x-www-form-urlencoded" data.
3. CCgiApplicationCached — new base class for CGI applications to allow for the caching of CGI results, which in some cases can help dramatically reduce CGI response time.
4. Add HTTP_X_FWD_IP_ADDR into tracking environment

BDB — Yet Another C++ API Based On BerkeleyDB

1. Added alternative location for transaction logs.

This offers parallel I/O option and better transaction performance.

2. Added more controls for the background write.
3. Reworked split store to use thread local transaction model.
4. Changed BDB cache to use split store to do non-locking parallel IO.
5. New garbage collection algorithm based on timeline (in-memory bit-index).

DBAPI — SQL Database Connectivity

1. Increased max. size of CDB_VarBinary and VarChar to 8000 bytes
2. Added attribute 'max_connect' to ctlib-based drivers. This attribute lets you set up max number of simultaneously opened connections (default value is 30)
3. C_DriverMgr::RegisterDriver(), C_DriverMgr::GetDriver() — deprecated

4 Deprecated the following functions:

DBAPI_RegisterDriver_CTLIB(I_DriverMgr& mgr),

DBAPI_RegisterDriver_DBLIB(I_DriverMgr& mgr)

DBAPI_RegisterDriver_FTDS(I_DriverMgr& mgr)

DBAPI_RegisterDriver_ODBC(I_DriverMgr& mgr)

DBAPI_RegisterDriver_MSDBLIB(I_DriverMgr& mgr)

DBAPI_RegisterDriver_MYSQL(I_DriverMgr& mgr) Please use functions with the same name but without an argument instead.

5 CDB_UserHandler_Exception and CDB_UserHandler_Exception_ODBC — new classes, to be used as database error message handlers which throw exception upon an error message;

6 CDriverManager::DestroyDs(const IDataSource* ds) — new method

7 Added version number to the FreeTDS library names (libsybdb_ftds.so —> libsybdb_ftds8.so; libtds_ftds.so —> libtds_ftds8.so)

8 Added DBLB_INSTALL_FACTORY macro in addition to DBLB_INSTALL_DEFAULT. New macro takes a factory name as a parameter.

9 Added macro DBAPI_TRANSACTION for RAII transaction support. Resource Acquisition Is Initialization (RAII) programming style is intended to revert a transaction automatically if any exception occurs in a code block

10 ctlib/bcp supports LongChar and LongBinary.data types now

11 Improved behavior of ctlib and ftds64 drivers in case of dead connection

ALGO/ALIGN/SPLIGN — Spliced and Generic Alignment Algorithms

1 CScoreBuilder — supplemented with the support for computing BLAST scores

2 CMultiAligner — added performance optimizations to the alignment of filler regions

3 CProSplign — spliced protein to genomic alignment algorithm

4 HFilter now supports a pairwise mode. In the pairwise mode, the uniquification algorithm is restricted to pairs of sequences. As a result, each sequence may have intervals with overlapping alignments as long as none of the overlapping alignments are between same two sequences.

5 An optional cross-filtering filtering has been added to the compartmentization algorithm, which allows to immediately use the compartment hits as splice refinement seeds.

6 Support for ASN.1 spliced segments was added. Spliced segment has been recently introduced to the NCBI data model to represent spliced alignments.

BLAST

1 Fixed a performance regression encountered when performing megablast with many queries and few database sequences

2 Added support for blastp or tblastn with a compressed alphabet and large word size

3 SeqDB (BLAST database access):

Add GetSequenceAsString() to fetch sequences as C++ strings.

Added DB filtering with Trace ID (TI) lists.

Added DB filtering with Seq-id lists.

Added DB filtering with negative GI or TI lists.

All SeqDB objects now share a global memory pool.

4 WriteDB (BLAST database creation):

WriteDB can now produce numerical ISAM files for trace IDs.

CBinaryListBuilder class for numerical list construction.

Reduced memory usage for ISAM construction.

Don't produce indexes or files that would be empty.

5 CBlastFastaReader — new class to read accessions, GIs, and Trace IDs.

6 CBlastScopeSource — to configure CScope objects to fetch sequence data from BLAST databases first, then from Genbank.

7 Introduced query splitting for blastx and blastn, users of C++ BLAST APIs should have transparent access to this feature.

8 Initial revisions of C++ BLAST command line binaries checked in src/app/blast (still under development). Previous command line binaries have been removed.

9 Discontinuous Seq-aligns are no longer produced by BLAST APIs.

10 CLocalDbAdapter — new class to provide BLAST database interfaces (BlastSeqSrc and

11 IBlastSeqInfoSrc) to the internal BLAST APIs.

12 Enabled the possibility to specify individual genetic codes for translated queries/ subjects.

13 Added BOOST based unit tests to xblast and blastinput libraries.

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

1 Added a low-level AGP line parser (CAgpRow) and a basic stream reader that detects scaffolds (CAgpReader)

2 CSeq_id::IdentifyAccession() — recognizes more prefixes (DI, EM-EV) and

3 (mixed-in) EMBL TPA protein accessions.

4 CFastaReader — now supports some new flags, all off by default: fParseRawID, fSkipCheck, and fNoSplit.

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

New functionality:

1. Allow getting CSeq_data object for gap segments.

2. Added CSeqMap::GetSegmentsCount().

3. Added CScope::UpdateAnnotIndex().

4. SSeq_align_Mapper — added a possibility to map a single alignment row rather than the whole alignment.

5. GetOverlappingFeatures() — new function to get all features that overlap the specified Seq-loc.

Bug fixes:

1. Fixed mapping of CSeq-align objects.

2. Fixed race condition in CSeq_id_Handle destructor.

3. Smart detection of when to warn about conflict in scope history.

4. Fixed loss of external annotations on edited Bioseq objects.

5. Fixed loss of Seq-ids from split-info.

6. Throw an exception instead of abort in case of duplicated blob-id.

7. GetBestXXX() and other overlap-related functions — fixed overlap types.

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

(GenBank data loader)

New functionality:

1. Separated ID2 logic from transport protocol (ID2 or PubSeqOS).

2. Implemented PubSeqOS2 reader - ID2 functionality via PubSeqOS connection.

3. Added description of ID connection to exception message.

4. Re-compress uncompressed ID2 data before storing it in cache.

5. Added optional dumping of request in case of error.

Bug fixes:

1. Fixed loading of SNP table with octet strings.

PTB — Project Tree Builder for MSVC++ .NET

1. Added option to add build configurations for VTune.
2. Modified CONFIGURE project on MS Visual Studio 8 platform so that it always uses project tree builder built in Release configuration, which makes it run faster.
3. Fixed makefile macro resolution on Unix platform, where PTB can be used to generate "flat" makefile for the whole Toolkit tree.

APPLICATIONS

1. NetSchedule

a) Run time config reload implemented correctly

b) Parameters and statistics for locks and mutexes added

c) Robustness of restart enhanced (private environment, or automatic recovery of shared environment)

d) Queries for the job tags and status added

e) Changes in the authorization system for easy migration - logging of invalid access added, access granted

GRID (DISTRIBUTED COMPUTING) FRAMEWORK

NetSchedule client API

1. Allow to get queue configuration parameters from NetSchedule server.
2. Added QUERY and SELECT NetSchedule commands (to get job tags and status from the server)

Grid Worker Node Implementation Framework

1. Added support for fast job status check, automatic control port discovery from a given range, and an 'auto' value (which sets the # of threads according to the # of CPUs available on the host) for 'max_threads' parameter

remote_app and remote_cgi utilites

1. 'kill_timeout' — a new configuration parameter, to specify the maximum execution time for the launched executables (working processes).

Documentation

Location

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

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 ICC 8.0 (build 20040520Z, package ID l_cc_pc_8.0.066_pe067.1) (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 4.0.1 ICC 9.0 (build 20051201) (GCC 4.1.1 - nominal support)
Solaris-10	SPARC	Sun C++ 5.5 (Studio8) patch 113817-19 GCC 4.0.1 (32-bit mode only)
Solaris-9 **(Support will be discontinued in the next release)**	x86-32	C++ 5.3 (WorkShop 6u2) patch 111686-13 GCC 3.4.3
Solaris-10	x86-32	Sun C++ 5.5 (Studio8) patch 113819-19 GCC 4.1.1
Solaris-10	x86-64	**Sun Studio 11 (C++ 5 .8 Patch 121017-08)**
IRIX64-6.5	SGI-Mips	MIPSpro 7.3.1.3m (64-bit, 32-bit)
FreeBSD-6.1	x86-32	GCC 3.4.4

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	Architecture	Compilers
Darwin on MacOS X 10.4 ("Tiger")	Native (PowerPC or x86-32)	GCC 4.0.1
Darwin on MacOS X 10.4 **- or newer**	**Universal (PowerPC and x86-32)**	GCC 4.0.1
Darwin on MacOS X 10.4 ("Tiger")	Native (PowerPC or x86-32)	Xcode 1.5 - 2.2.1

Discontinued

Operating System	Architecture	Compilers
Solaris-8	SPARC	C++ 5.3 (WorkShop 6u2) GCC 3.4.3
Digital Tru64 Unix 5.1 (aka OSF1)	ALPHA	GCC 3.3.2
FreeBSD-4.10	x86-32	GCC 3.4.2

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 September 17, 2007.