Release Notes (March, 2008)

Created: April 11, 2008.
Last Update: April 11, 2008.

Download

Download the source code archives at:

ftp://ftp.ncbi.nih.gov/toolbox/ncbi_tools++/2008/Mar_17_2008/

- ncbi_cxx-- Mar_17_2008.tar.gz — for UNIX'es (see the list of UNIX flavors below) and MacOSX
- ncbi_cxx-- Mar_17_2008.gtar.gz — for UNIX'es (see the list of UNIX flavors below) and MacOSX
- ncbi_cxx-- Mar_17_2008.exe — for MS-Windows (32- and 64-bit) / MSVC++ (7.1, 8.0) — self-extracting
- ncbi_cxx-- Mar_17_2008.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 19, 2008.

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. CPluginManager -- now looks for libraries in $LD_LIBRARY_PATH first, hardcoded RPATH second (reversed the lookup order)
2. CProcess -- added method KillGroup()
3. CExec -- added new modes for Spawn*() methods (eWaitGroup, eNoWaitGroup)
4. CFileIO -- new class, implementing low-level I/O API for files
5. IWriter, IReaderWriter -- added implementations for the low level reading from file via the bare system IO handle
6. CTmpFile -- new class, to generate temporary name for file (it can also use user specified name) that can be automatically deleted on the object's destruction
7. CSignal -- new class, to handle OS signals
8. CFastLocalTime -- added support for nanoseconds
9. CStopWatch -- new method IsRunning()
10. CNcbiApplication -- added option to print description of application arguments in XML format
11. CArgDescriptions -- added possibility to take default value for an argument from an environment variable
12. CDiagCollectGuard -- new RAII class, to replace StartTraceCollect() and StopTraceCollect()

13. <corelib/stream_utils.hpp> -- improved pushback performance; add light stepback method; make use of them in UTIL classes

CONNECT — Data streaming, Networking, and Dispatching

1. CPipe -- added possibility to run child process in new process group (UNIX) and to kill the whole tree of processes
2. <connect/ncbi_socket.h[pp]>:
    - SOCK_GetLocalPort -- new
    - CSocket::GetLocalPort -- new
    - SOCK_isip[Ex]() -- to allow all documented IP formats
    - LSOCK_Create() -- to actually heed CloseOnExec (it was silently ignored before)
3. Define STimeout/ms conversions (including infinite timeouts); make use of it
4. CConn_IOStream and its buffer -- added a fast initial get segment

XCONNSERV -- NCBI Grid Client API Library

1. CNetCacheAPI::HasBlob() -- new, to check if the BLOB with the specified key exists

UTIL — Miscellaneous Low-Level APIs

Modified methods that translate UTF-8 strings into ASCII ones -- added default translation, to be used when the translation is not possible

1. CTar -- implemented PAX extensions for TAR (read/extract only)

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

1. CObjectOStreamJson new class that implements JavaScript Object Notation formatted output
2. Added support for the base64Binary encoded data
3. Improved compatibility between classes generated from XML schema and from ASN.1 specification

DATATOOL — Code Generator and Data Converter Tool

1. Added option to convert data into JSON format
2. Improved support for XML schema, including circular includes, recursive type definitions, definitions with multiple levels of inheritance, handling of comments
3. Improved support for DTD, including skipping general entities declarations and using nmtokens in enumerated values
4. Added support for base64Binary encoded data
5. Improved compatibility between classes generated from XML schema and from ASN.1 specification.

CGI — CGI and Fast-CGI Application Framework

1. CCgiUserAgent -- new method GetBrowserName()
2. Fixed a bug resulted in repeated creation of the session object each time the CCgiRequest::GetSession() method was called

DBAPI -- Generic SQL Database Connectivity

1 Made ftds64 (instead of ftds8) the default FreeTDS driver, and renamed drivers correspondingly:

- ftds to ftds8
- ftds64 to ftds
- ftds64_odbc to ftds_odbc
- ftds64_dblib to ftds_dblib
- ftds64 driver now supports UTF-8 client character encoding

2 CTL_Connection::SetTimeout() -- implemented for the ftds64 driver

3 Discontinued support of msdblib and ftds63 drivers

4 Introduced a uniform driver-independent control of the total amount of concurrent database connections. Maximum amount is set to 100 by default.

5 Reusable connections will not be closed after call to method Close() anymore, they will return to the pool.

6 Sybase CTLIB driver -- cancellation of bulk-insert operation is not available now (to avoid crash due to a bug in at least some versions of Sybase client)

7 Enable BCP by default with all database drivers.

8 Removed methods:

9 DBAPI_RegisterDriver_[CTLIB | DBLIB | FTDS | ODBC | MSDBLIB | MYSQL] (I_DriverMgr& mgr)

10 DBAPI_RegisterDriver_MSDBLIB (void)

11 Deprecated methods:

12 I_LangCmd::More(const string&)

13 CDB_LangCmd::More(const string&)

14 CDB_Exception::Severity()

15 CDB_Exception::SeverityString(EDB_Severity)

16 Added methods:

17 I_DriverContext::MakeConnection()

18 CDBUDRandomMapper::Add() and CDBUDPriorityMapper::Add()

19 CTrivialConnValidator::GetAttr()

20 CTrivialConnValidator::GetName() and CConnValidatorCoR::GetName()

21 CDBHandlerStack::GetSize()

22 CDriverContext::ResetEnvSybase()

23 IResultSetMetaData::GetDirection()

24 Changed signature of Get_I_DriverContext()

25 Describe SQL statement input/output parameters

26 Lock SQL statement parameter binding after they have been used for the first time

27 Report extra debug information (such as SQL statement, name of stored procedure, etc) with exceptions

28 Put server name, user name, and extra-message to the database error message in CDB_UserHandler_Exception::HandleIt()

Python DBAPI module

1. Fixed 'callproc' to return modified copy of the input sequence.

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

1. Added support for spliced and sparse alignments in mappers and Object Manager.
2. Added functions ConvertSeqLocsToPairwiseAln() and SeqLocMapperToPairwiseAligns() to convert between seq-locs and seq-loc mappers and pairwise alignments.
3. CSeq_id::IdentifyAccession recognizes more prefixes (DJ, EW-FE, FM-FR) and (mixed-in) EMBL TPA protein accessions
4. CSeq_id::EAccessionInfo uses standardized values for all divisions, even those associated only with a single accession type, and incorporates one bit of that to mark definite genomic sequences
5. CSeq_id's FASTA converters use tr| rather than sp| as the tag for unreviewed UniProt [formerly Swiss-Prot] IDs in both directions
6. CSeq_id's FASTA and raw parsers are generally more forgiving, and in particular now recognize bare PRF/SEQDB loci
7. CFastaReader supports two new flags for more comprehensive input validation, both off by default: fValidate and fUniqueIDs

ALGO/ALIGN -- Generic Alignment Algorithms

1. NW_ALIGNER -- the aligner moved to under the "src/app"
2. CBandAligner -- space requirements reduced by storing two dynprog cells per byte
3. Splign:
    - the minimum singleton identity can now be specified in absolute units
    - the maximum intron length has been upwardly revised
    - 'auto' query direction mode introduced, in which the query is first aligned in the direction of its maximum ORF, and then re-aligned in the opposite direction if a non-consensus splice was found
4. COMPART -- in addition to using external BLAST hits, the application is now able to go on its own by utilizing a novel hit search engine relying on use of participation vector and index-to-index comparison. To utilize this mode, a user must prepare BLAST databases (using formatdb) of cDNA and genomic sequences then specify them in -qdb and -sdb arguments. The new engine does not require any external repeat filtering, is faster and more sensitive than MEGABLAST. Its application scope is same-species comparison. For cross-species, discontiguous MEGABLAST is advised.

ALNMGR -- Bio-sequence Alignment Manager

1. Old-style alignment manager:
    - Added implementation of IAlnSegmentIterator for CAlnVec: CAlnVecIterator
    - New segment types: fUnalignedOn{Left,Right}OnAnchor
2. New-style alignment manager:
3. Added alignment serialization

4 Added support for spliced seg alignments

5 Use a more specific {E,T}SegTypeFlags type instead of {EType,int}

6 Renamed MergeAlnRngColl to MergePairwiseAlns and moved it's definition to aln_builders.cpp

7 Added conditional splitting of disc alignments when they are inserted in the container

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

1 Table features (seqtable.asn) are now supported in the object manager. Fast access to table features is implemented via CSeq_feat_Handle.

2 Added CScope::GetLabel(seq_id), which could retrieve only label information if possible

3 CObjectManager singleton now lives until the very end of application

4 CSeqVector and CSeqVector_CI modifications:

- CSecVector can now be created faster directly from CBioseq object.
- Class CNcbi2naRandomizer is renamed to interface INcbi2naRendomizer.
- CSeqVector now implements GetPackedSeqData() to get NCBI2na and NCBI4na encodings in packed form (more than one base per char).

5 CSeqMap Several modifications:

6 Implemented CSeqMap_CI::IsSeqData()

7 Added CBioseq_EditHandle::SetSeqMap() to allow editing of sequence map

BLAST

1 WriteDB -- now supports building optional ISAM files mapping "sequence hash" values to a list of OIDs

2 Use Int8 for ISAM numeric values internally, to support TI list filtering when TIs exceed 32 bit range

3 Introduction of XML formatting for BLAST command line binaries

4 Implemented importing/exporting of search strategy in BLAST command line

5 binaries

6 Added segmasker application to filter protein sequences

BIO-TOOLS

1 objtools/eutils -- new API to access [http://eutils.ncbi.nlm.nih.gov/entrez/query/static/eutils_help.html eUtils tools]

2 CFastaOstream -- now supports setting hard and soft masks, and sports virtual

3 Write* methods to ease subclassing

4 GetProteinWeights -- now supports proteins containing pyrrolysine (O) and leucine/isoleucine ambiguities (J), and properly skips proteins containing Xs rather than spuriously returning weight 0 in some cases

5 GenBank data loader -- no longer depends on the PubSeqOS backend, which is only useful within NCBI and is therefore not part of the release

APPLICATIONS

1 NetCache:

BUILD FRAMEWORK (UNIX)

1. A number of scripts and auxiliary files have moved into subdirectories to reduce clutter. In particular, the configure frontends for various compilers are moving from the "compilers" directory to its new "unix" subdirectory. This release continues to include copies in the old location for convenience, but future releases will not.

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

1. Improved performance, redesigned analysis of project dependencies, enhanced reporting
2. Modified project filter to use regular expressions
3. Redesigned description of DLL projects - to use special makefiles instead of single dll_info.ini file (see the new "src/dll" subtree)

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 ICC 8.0 (build 20040520Z, package ID l_cc_pc_8.0.066_pe067.1) (GCC 3.0.4, 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)
Linux	x86-32 and x86-64	**GCC 4.1.2** **GCC 4.2.3** (nominal support only)
Linux-2.6.x	x86-32	GCC 2.95.3 (support to be discontinued in the next release)
Solaris-10	SPARC	Sun C++ 5.5 (Studio8) patch 113817-19 GCC 4.0.1 (32-bit mode only)
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) (nominal support only)
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 C++ 2005 (C++ 8.0)
MS Windows-64	MS Visual C++ 2005 (C++ 8.0)
Cygwin 1.5.1832	GCC 3.4.4 (nominal support only)

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-9	x86-32	C++ 5.3 (WorkShop 6u2) patch 111686-13 GCC 3.4.3

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 avoiding 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 April 15, 2008.