Xerces C++ Documentation

Table of Contents

1. Xerces C++ Parser 5

- Xerces C++ Version 1.5.1 5
- Applications of the Xerces Parser 5
- Features 5
- Platforms with Binaries 5
- Other ports... 6

2. Installation 7

- Window NT/98 7
- UNIX 7

3. Build Instructions 9

- Building on Windows and UNIX 9
- Building on Other Platforms 14
- Other Build Instructions 21

4. API Documentation 25

- API Docs for SAX and DOM 25

5. Xerces C++ Samples 26

- Building the Samples 26
- Building the Samples for OS2 26
- Running the Samples 27
- Xerces C++ Sample 1: SAXCount 29
- Xerces C++ Sample 2: SAXPrint 31
- Xerces C++ Sample 3: DOMCount 34
- Xerces C++ Sample 4: DOMPrint 36
- Xerces C++ Sample 5: MemParse 39
- Xerces C++ Sample 6: Redirect 41
- Xerces C++ Sample 7: PParse 42
- Xerces C++ Sample 8: StdInParse 44
- Xerces C++ Sample 9: EnumVal 45
- Xerces C++ Sample 10: CreateDOMDocument 47
- Xerces C++ Sample 11: SAX2Count 48
- Xerces C++ Sample 12: SAX2Print 50
- Xerces C++ Sample 13: IDOMCount 53
- Xerces C++ Sample 14: IDOMPrint 55

6. Schema 58

- Disclaimer 58
- Introduction 58
- Limitations 58
- Features/Datatypes Supported 58
- Other Limitations 59
- Usage 59

7. Frequently Asked Questions 61

- Distributing Xerces C++ 61
- Parsing with Xerces C++ 63

Other Xerces C++ Questions 70

8. Programming Guide 72

SAX1 Programming Guide 72
SAX2 Programming Guide 74
DOM Programming Guide 77
Experimental IDOM Programming Guide 80

9. Migration 85

Migrating from Xerces C++ 1.4.0 to Xerces C++ 1.5.1 85
General Improvements 85
Changes required to migrate to Xerces C++ 1.5.1 86
New features in Xerces C++ 1.5.1 86
Migration Archive 86

10. Migration Archive 87

Migrating from XML4C 2.x to Xerces C++ 1.4.0 87
General Improvements 87
Summary of changes required to migrate from XML4C 2.x to Xerces C++ 1.4.0 88
The Samples 89
Parser Classes 89
DOM Level 2 support 90
Progressive Parsing 90
Namespace support 91
Moved Classes to src/framework 91
Loadable Message Text 91
Pluggable Validators 92
Pluggable Transcoders 92
Util directory Reorganization 92

11. Releases 94

Xerces C++ Version 1.5.1: July 18, 2001 94
Xerces C++ Version 1.5.0: June 15, 2001 96
Release Archive 100

12. Releases Archive 101

Xerces C++ Version 1.4.0: January 31, 2001 101
Xerces C++ Version 1.3.0: Sept 21, 2000 104
Xerces C++ Version 1.2.0: June 22, 2000 108
Xerces C++ Version 1.1.0: Feb 28, 2000 111
Xerces C++ Version 1.0.1: December 15, 1999 112
Xerces C++ Parser Version 1.0.0: December 7, 1999 113
Xerces C++ BETA November 5, 1999 113

13. Bug Reporting 114

How to report bugs 114
Search frist 114
Write good bug report 114

14. Feedback Procedures 116

Questions or Comments 116

Acknowledgements 116

15. Y2K Compliance 119

Apache Xerces Parser Year-2000 Readiness 119

16. PDF Documentation 120

PDF Documentation 120

Appendix A: Links Reference 121

1
Xerces C++ Parser

Xerces C++ Version 1.5.1

Xerces C++ is a validating XML parser written in a portable subset of C++. Xerces C++ makes it easy to give your application the ability to read and write XML [1] data. A shared library is provided for parsing, generating, manipulating, and validating XML documents. Xerces C++ is faithful to the XML 1.0 [2] recommendation and associated standards (DOM 1.0 [3] , DOM 2.0 [4] . SAX 1.0 [5] , SAX 2.0 [6] , Namespaces [7]). Xerces C++ 1.5.1 also provides an implementation of a subset of the Schema [8] . The parser provides high performance, modularity, and scalability. Source code, samples and API documentation are provided with the parser. For portability, care has been taken to make minimal use of templates, no RTTI, no C++ namespaces and minimal use of #ifdefs.

Applications of the Xerces Parser

Xerces has rich generating and validating capabilities. The parser is used for:

- Building XML-savvy Web servers
- Building next generation of vertical applications that use XML as their data format
- On-the-fly validation for creating XML editors
- Ensuring the integrity of e-business data expressed in XML
- Building truly internationalized XML applications

Features

- Conforms to XML Spec 1.0 [2]
- Tracking of latest DOM (Level 1.0) [3] , DOM (Level 2.0) [4] , SAX/SAX2 [6] , Namespace [7] specifications.
- Experimental Schema [8] subset support
- Source code, samples, and documentation is provided.
- Programmatic generation and validation of XML
- Pluggable catalogs, validators and encodings
- High performance
- Customizable error handling

Platforms with Binaries

- Win32 using MSVC 6.0 SP3
- Linux (RedHat 6.1) using egcs-2.91.66 and glibc-2.1.2-11
- Solaris 2.6 using Sun Workshop 4.2
- AIX 4.3 using xlC 3.6.4
- HP-UX 11 using aCC A.03.13 with pthreads

Other ports...

- OS/390
- AS/400
- SGI IRIX
- Macintosh
- OS/2
- PTX
- and more!

2 Installation

Window NT/98

Install the binary Xerces C++ release by using `unzip` on the *file*-win32.zip archive in the Windows environment. You can use WinZip, or any other UnZip utility.

```
unzip xerces-c1_5_1-win32.zip
```

This creates a 'xerces-c1_5_1-win32' sub-directory containing the Xerces C++ distribution.

You need to add the 'xerces-c1_5_1-win32\bin' directory to your path:

To do this under Windows NT, go to the start menu, click the settings menu and select control panel. When the control panel opens, double click on System and select the 'Environment' tab. Locate the PATH variable under system variables and add <full_path_to_xerces-c1_5_1 >\bin to the PATH variable. To do this under Windows 95/98 add this line to your AUTOEXEC.BAT file:

```
SET PATH=<full_path_to_xerces-c1_5_1>\bin;%PATH%
```

or run the `SET PATH` command in your shell window.

UNIX

Binary installation of this release is to extract the files from the compressed .tar archive (using 'tar').

```
cd $HOME
gunzip xerces-c1_5_1-linux.tar.gz
tar -xvf xerces-c1_5_1-linux.tar
```

This will create an 'xerces-c1_5_1-linux' sub-directory (in the home directory) which contains the Xerces C++ distribution. You will need to add the xerces-c1_5_1-linux/bin directory to your PATH environment variable:

For Bourne Shell, K Shell or Bash, type:

```
export PATH="$PATH:$HOME/xerces-c1_5_1-linux/bin"
```

For C Shell, type:

```
setenv PATH "$PATH:$HOME/xerces-c1_5_1-linux/bin"
```

If you wish to make this setting permanent, you need to change your profile by changing your setup files which can be either .profile or .kshrc.

In addition, you will also need to set the environment variables XERCESCROOT, ICUROOT and the library search path. (LIBPATH on AIX, LD_LIBRARY_PATH on Solaris and Linux, SHLIB_PATH on HP-UX).

Note: *XERCESCROOT and ICUROOT are needed only if you intend to recompile the samples or build your own applications. The library path is necessary to link the shared libraries at runtime.*

For Bourne Shell, K Shell or Bash, type:

```
export XERCESCROOT=<wherever you installed Xerces C++>
export ICUROOT=<wherever you installed ICU>
export LIBPATH=$XERCESCROOT/lib:$LIBPATH (on AIX)
export LD_LIBRARY_PATH=$XERCESCROOT/lib:$LD_LIBRARY_PATH (on Solaris, Linux)
export SHLIB_PATH=$XERCESCROOT/lib:$SHLIB_PATH (on HP-UX)
```

For C Shell, type:

```
setenv XERCESCROOT "<wherever you installed Xerces C++>"
setenv ICUROOT "<wherever you installed ICU>"
setenv LIBPATH "$XERCESCROOT/lib:$LIBPATH" (on AIX)
setenv LD_LIBRARY_PATH "$XERCESCROOT/lib:$LD_LIBRARY_PATH" (on Solaris, Linux)
setenv SHLIB_PATH "$XERCESCROOT/lib:$SHLIB_PATH" (on HP-UX)
```

Note: *If you need to build the samples after installation, make sure you read and follow the build instructions given in the FAQ.*

3
Build Instructions

Building on Windows and UNIX

Building Xerces C++ on Windows NT/98

Xerces C++ comes with Microsoft Visual C++ projects and workspaces to help you build Xerces C++. The following describes the steps you need to build Xerces C++.

Building Xerces C++ library

To build Xerces C++ from it source (using MSVC), you will need to open the workspace containing the project. If you are building your application, you may want to add the Xerces C++ project inside your applications's workspace.

The workspace containing the Xerces C++ project file and all other samples is:

```
xerces-c-src1_5_1\Projects\Win32\VC6\xerces-all\xerces-all.dsw
```

Once you are inside MSVC, you need to build the project marked **XercesLib**.

If you want to include the Xerces C++ project separately, you need to pick up:

```
xerces-c-src1_5_1\Projects\Win32\VC6\xerces-all\XercesLib\XercesLib.dsp
```

You must make sure that you are linking your application with the xerces-c_1.lib library and also make sure that the associated DLL is somewhere in your path.

> ***Note:*** *If you are working on the AlphaWorks version which uses ICU, you must have the ICU data DLL named* `icudata.dll` *available from your path setting. For finding out where you can get ICU from and build it, look at the How to Build ICU.*

Building samples

Inside the same workspace (xerces-all.dsw), you'll find several other projects. These are for the samples. Select all the samples and right click on the selection. Then choose "Build (selection only)" to build all the samples in one shot.

Building Xerces C++ on Windows using Visual Age C++

A few unsupported projects are also packaged with Xerces C++. Due to origins of Xerces C++ inside IBM labs, we do have projects for IBM's Visual Age C++ compiler [9] on Windows. The following describes the steps you need to build Xerces C++ using Visual Age C++.

Building Xerces C++ library

Requirements:

- VisualAge C++ Version 4.0 with Fixpak 1:

Download the Fixpak [10] from the IBM VisualAge C++ Corrective Services web page.

To include the ICU library:

- ICU Build:

  You should have the ICU Library [11] in the same directory as the Xerces C++ library. For example if Xerces C++ is at the top level of the d drive, put the ICU library at the top level of d e.g. d:/xerces-c1_5_1 d:/icu.

Instructions:

1. Change the directory to d:\xerces-c1_5_1\Projects\Win32
2. If a d:\xerces-c1_5_1\Project\Win32\VACPP40 directory does not exist, create it.
3. Copy the IBM VisualAge project file, `XML4C2X.icc`, to the VACPP40 directory.
4. From the VisualAge main menu enter the project file name and path.
5. When the build finishes the status bar displays this message: Last Compile completed Successfully with warnings on date.

Note: *These instructions assume that you install in drive d:\. Replace d with the appropriate drive letter.*

Building Xerces C++ on UNIX platforms

Xerces C++ uses GNU [12] tools like Autoconf [13] and GNU Make [14] to build the system. You must first make sure you have these tools installed on your system before proceeding. If you don not have required tools, ask your system administrator to get them for you. These tools are free under the GNU Public Licence and may be obtained from the Free Software Foundation [12] .

Do not jump into the build directly before reading this.

Spending some time reading the following instructions will save you a lot of wasted time and support-related e-mail communication. The Xerces C++ build instructions are a little different from normal product builds. Specifically, there are some wrapper-scripts that have been written to make life easier for you. You are free not to use these scripts and use Autoconf [13] and GNU Make [14] directly, but we want to make sure you know what you are by-passing and what risks you are taking. So read the following instructions carefully before attempting to build it yourself.

Besides having all necessary build tools, you also need to know what compilers we have tested Xerces C++ on. The following table lists the relevant platforms and compilers.

Operating System	C++, C Compilers
Redhat Linux 6.1	g++, gcc (egcs)
AIX 4.2.1 and higher	xlC_r, xlc_r
Solaris 2.6	CC, cc
HP-UX 11	aCC, cc

If you are not using any of these compilers, you are taking a calculated risk by exploring new grounds. Your effort in making Xerces C++ work on this new compiler is greatly appreciated and any problems you face can be addressed on the Xerces-C mailing list [15] .

Differences between the UNIX platforms: The description below is generic, but as every programmer is aware, there are minor differences within the various UNIX flavors the world has been bestowed with. The one difference that you need to watch out in the discussion below, pertains to the system environment variable for finding libraries. On **Linux and Solaris**, the environment variable name is called `LD_LIBRARY_PATH`, on **AIX** it is `LIBPATH`, while on **HP-UX** it is `SHLIB_PATH`. The following discussion assumes you are working on Linux, but it is with subtle understanding that you know how to interpret it for the other UNIX flavors.

> ***Note:*** *If you wish to build Xerces C++ with ICU, look at the Building ICU. It tells you where you can get ICU and how to build Xerces C++ with it.*

Setting build environment variables

Before doing the build, you must first set your environment variables to pick-up the compiler and also specify where you extracted Xerces C++ on your machine. While the first one is probably set for you by the system administrator, just make sure you can invoke the compiler. You may do so by typing the compiler invocation command without any parameters (e.g. xlc_r, or g++, or cc) and check if you get a proper response back.

Next set your Xerces C++ root path as follows:

```
export XERCESCROOT=<full path to xerces-c-src1_5_1>
```

This should be the full path of the directory where you extracted Xerces C++.

Building Xerces C++ library

As mentioned earlier, you must be ready with the GNU tools like autoconf [13] and gmake [14] before you attempt the build.

The autoconf tool is required on only one platform and produces a set of portable scripts (configure) that you can run on all other platforms without actually having the autoconf tool installed everywhere. In all probability the autoconf-generated script (called `configure`) is already in your `src` directory. If not, type:

```
cd $XERCESCROOT/src
autoconf
```

This generates a shell-script called `configure`. It is tempting to run this script directly as is normally the case, but wait a minute. If you are using the default compilers like gcc [16] and g++ [16] you do not have a problem. But if you are not on the standard GNU compilers, you need to export a few more environment variables before you can invoke configure.

Rather than make you to figure out what strange environment variables you need to use, we have provided you with a wrapper script that does the job for you. All you need to tell the script is what your compiler is, and what options you are going to use inside your build, and the script does everything for you. Here is what the script takes as input:

```
runConfigure
runConfigure: Helper script to run "configure" for one of the
              supported platforms.
Usage: runConfigure "options"
       where options may be any of the following:
       -p <platform> (accepts 'aix', 'linux', 'solaris',
                           'hp-10', 'hp-11', 'irix', 'unixware')
       -c <C compiler name> (e.g. xlc_r, gcc, cc)
       -x <C++ compiler name> (e.g. xlC_r, g++, CC, aCC)
       -d (specifies that you want to build debug version)
       -m <message loader> can be 'inmem', 'icu', 'iconv'
       -n <net accessor> can be 'fileonly', 'libwww'
       -t <transcoder> can be 'icu' or 'native'
       -r <thread option> can be 'pthread' or 'dce' (only used on HP-11)
       -l <extra linker options>
       -z <extra compiler options>
```

```
          -h (to get help on the above commands)
```

Note: *Xerces C++ can be built as either a standalone library or as a library dependent on International Components for Unicode (ICU). For simplicity, the following discussion only explains standalone builds.*

One of the common ways to build Xerces C++ is as follows:

```
    runConfigure -plinux -cgcc -xg++ -minmem -nfileonly -tnative
```

The response will be something like this:

```
    Generating makefiles with the following options ...
    Platform: linux
    C Compiler: gcc
    C++ Compiler: g++
    Extra compile options:
    Extra link options:
    Message Loader: inmem
    Net Accessor: fileonly
    Transcoder: native
    Thread option:
    Debug is OFF

    creating cache ./config.cache
    checking for gcc... gcc
    checking whether the C compiler (gcc -O -DXML_USE_NATIVE_TRANSCODER
   -DXML_USE_INMEM_MESSAGELOADER   ) works... yes
    checking whether the C compiler (gcc -O -DXML_USE_NATIVE_TRANSCODER
   -DXML_USE_INMEM_MESSAGELOADER   ) is a cross-compiler... no
    checking whether we are using GNU C... yes
    checking whether gcc accepts -g... yes
    checking for c++... g++
    checking whether the C++ compiler (g++ -O -DXML_USE_NATIVE_TRANSCODER
   -DXML_USE_INMEM_MESSAGELOADER   ) works... yes
    checking whether the C++ compiler (g++ -O -DXML_USE_NATIVE_TRANSCODER
   -DXML_USE_INMEM_MESSAGELOADER   ) is a cross-compiler... no
    checking whether we are using GNU C++... yes
    checking whether g++ accepts -g... yes
    checking for a BSD compatible install... /usr/bin/install -c
    checking for autoconf... autoconf
    checking for floor in -lm... yes
    checking how to run the C preprocessor... gcc -E
    checking for ANSI C header files... yes
    checking for XMLByte... no
    checking host system type... i686-pc-linux-gnu
    updating cache ./config.cache
    creating ./config.status
    creating Makefile
    creating util/Makefile
    creating util/Transcoders/ICU/Makefile
    creating util/Transcoders/Iconv/Makefile
    creating util/Transcoders/Iconv390/Makefile
    creating util/Transcoders/Iconv400/Makefile
```

```
 creating util/Platforms/Makefile
 creating util/Compilers/Makefile
 creating util/MsgLoaders/InMemory/Makefile
 creating util/MsgLoaders/ICU/Makefile
 creating util/MsgLoaders/MsgCatalog/Makefile
 creating util/MsgLoaders/MsgFile/Makefile
 creating validators/DTD/Makefile
 creating framework/Makefile
 creating dom/Makefile
 creating parsers/Makefile
 creating internal/Makefile
 creating sax/Makefile
 creating ../obj/Makefile
 creating conf.h
 cat: ./conf.h.in: No such file or directory
 conf.h is unchanged

 Having build problems? Read instructions at
http://xml.apache.org/xerces-c/build.html
 Still cannot resolve it? Find out if someone else had the same problem before.
 Check the mailing list archives at http://archive.covalent.net.

 In future, you may also directly type the following commands to create the
Makefiles.

 export TRANSCODER=NATIVE
 export MESSAGELOADER=INMEM
 export USELIBWWW=0
 export CC=gcc
 export CXX=g++
 export CXXFLAGS=-O -DXML_USE_NATIVE_TRANSCODER -DXML_USE_INMEM_MESSAGELOADER
 export CFLAGS=-O -DXML_USE_NATIVE_TRANSCODER -DXML_USE_INMEM_MESSAGELOADER
 export LIBS= -lpthread
 configure

 If the result of the above commands look OK to you, go to the directory
 XERCESCROOT and type "gmake" to make the Xerces C++ system.
```

> ***Note:*** *The error message concerning* `conf.h` *is NOT an indication of a problem. This code has been inserted to make it work on AS/400, but it gives this message which appears to be an error. The problem will be fixed in future.*

So now you see what the wrapper script has actually been doing! It has invoked `configure` to create the Makefiles in the individual sub-directories, but in addition to that, it has set a few environment variables to correctly configure your compiler and compiler flags too.

Now that the Makefiles are all created, you are ready to do the actual build.

```
 gmake
```

Is that it? Yes, that's all you need to build Xerces C++.

Building samples

Similarly, you can build the samples by giving the same commands in the `samples` directory.

```
cd $XERCESCROOT/samples
runConfigure -plinux -cgcc -xg++
gmake
```

The samples get built in the `bin` directory. Before you run the samples, you must make sure that your library path is set to pick up libraries from `$XERCESCROOT/lib`. If not, type the following to set your library path properly.

```
export LD_LIBRARY_PATH=$XERCESCROOT/lib:$LD_LIBRARY_PATH
```

You are now set to run the sample applications.

Building Xerces C++ as a single-threaded library on Unix platforms

To build a single-threaded library on Unix platforms you have to update one or more of the following files `Makefile.incl, Makefile.in, runConfigure`. The following steps guide you to create a single-threaded library for each platform:

For Aix -

- Replace `xlc_r` and `xlC_r` libraries with `xlc` and `xlC` respectively
- Replace `makeC++SharedLib_r` with `makeC++SharedLib`
- Remove the flag `-D_THREAD_SAFE`
- Remove inclusion of any threaded library directories from the `LIBPATH`
- Remove inclusion of `-lpthreads` and `-lpthread_compat`
- Add `-DAPP_NO_THREADS` to define the variable under AIX specific options in `Makefile.incl`

For Solaris -

- Add `-DAPP_NO_THREADS` to define the variable under SOLARIS specific options in `Makefile.incl`
- Remove compiler switch `-mt`
- Remove `-D_REENTRANT` flag from the 'compile' options
- Remove inclusion of `-lpthread`

For Linux -

- Add `-DAPP_NO_THREADS` to define the variable under LINUX specific options in `Makefile.incl`
- Remove `-D_REENTRANT` flag from the 'compile' options
- Remove inclusion of `-lpthread`

For HPUX -

- Add `-DAPP_NO_THREADS` to define the variable under HP specific options in `Makefile.incl`
- Remove inclusion of `-lpthread` and `-lcma`
- Remove threading defines like `-D_PTHREADS_DRAFT4 , -DXML_USE_DCE`

Building on Other Platforms

Building Xerces C++ on OS/2 using Visual Age C++

OS/2 is a favourite IBM PC platforms. The only option in this platform is to use Visual Age C++ compiler [9] . Here are the steps you need to build Xerces C++ using Visual Age C++ on OS/2.

Building Xerces C++ library

Requirements:

- VisualAge C++ Version 4.0 with Fixpak 1:

Download the Fixpak [10] from the IBM VisualAge C++ Corrective Services web page.

There are two ways to build Xerces C++. The "From Existing" method only requires VAC++. The "From Scratch" method requires both Object Rexx and VAC++ installed.

The "From Existing" Method

1. In the `xerces-c-src1_5_1\Projects\OS2\VACPP40` directory, find and edit the VAC++ configuration file `project_options.icc`.
2. Change the directory on the first line `'BASE_DIR = "..."'` to match the base directory of the Xerces C++ sources on your system. Note that the directory path must use double backslashes `"\\"`!
3. Save `project_options.icc`
4. Start the Command Line in the VAC++ folder.
5. Navigate to the `xerces-c-src1_5_1\Projects\OS2\VACPP40` directory.
6. Run `build.cmd`. This does a migration build.
7. When `build.cmd` finishes, review the file `compiler.errors`. This file should contain only informational messages, almost all complaining about constant values in comparisons.
8. You should now have a `xerces-c.dll` and `xerces-c.lib`. The library file is an import library for the DLL.

The "From Scratch" Method

1. If you are not currently running `Object Rexx`, run the `SWITCHRX` command from a command line, answer `"yes"` to switching to `Object Rexx`, and follow the instructions to reboot. You can switch back to `"Classic Rexx"` by running `SWITCHRX` again. But you probably won't need to switch back since `Object Rexx` runs almost 100% of Classic Rexx programs.
2. In the `xerces-c-src1_5_1\Projects\OS2\VACPP40` directory, run `genICC.cmd`. This builds the VAC++ configuration files for the sources you have on your system.
3. Check the generated `ICC` files to ensure that they didn't pick up some non-OS/2 platform stuff. This happens when new platform-specific directories are added to Xerces. If they did pick up new non-OS/2 stuff, either edit it out of the `ICC` file or add them to the "ignore" array in `genICC.cmd` and re-run `genICC`.
4. Start the Command Line in the VAC++ folder.
5. Navigate to the `xerces-c-src1_5_1\Projects\OS2\VACPP40` directory.
6. Run `build.cmd` This does a migration build.
7. When `build.cmd` finishes, review the file `compiler.errors`. This file should contain only informational messages, almost all complaining about constant values in comparisons.
8. You should now have a `xerces-c.dll` and `xerces-c.lib`. The library file is an import library for the DLL.)

Packaging the Binaries

There is an `Object Rexx` program that will package the binaries and headers. (See step 1 of the "From scratch" method on how to switch to `Object Rexx`.) The `packageBinaries.cmd` file is in the `xerces-c-src1_5_1\Projects\OS2\VACPP40` directory. Run `packageBinaries`, giving the source and target directories like this:

```
packageBinaries -s x:\xerces-c-src1_5_1 -o x:\temp\xerces-c1_5_1-os2
```

(Match the source directory to your system; the target directory can be anything you want.)

> ***Note:*** *If you don't want to use the `Object Rexx` program, you'll need to manually copy the "*.hpp" and "*.c" files to an include directory. (Be sure to maintain the same directory structure that you find under `xerces-c-src1_5_1`.)*

Building Xerces C++ on AS/400

The following addresses the requirements and build of Xerces C++ natively on the AS/400.

Building Xerces C++ library

Requirements:

- `QSHELL` interpreter installed (install base option 30, operating system)
- QShell Utilities, PRPQ 5799-XEH
- ILE C++ for AS/400, PRPQ 5799-GDW
- GNU facilities (the gnu facilities are currently available by request only. Send e-mail to rchgo400@us.ibm.com [17])

Recommendations:

- There are a couple of options when building the Xerces C++ parser on AS/400. For messaging support, you can use the in memory message option or the message file support. For code page translation, you can use the AS/400 native `Iconv400` support or ICU. If you choose ICU, follow the instructions to build the ICU service program with the ICU download. Those instructions are not included here.
- Currently we recommend that you take the options of `MsgFile` and `Iconv400` (see below)

Setup Instructions:

- Make sure that you have the requirements installed on your AS/400. We highly recommend that you read the writeup that accompanies the gnu facilities download. There are install instructions as well as information about how modules, programs and service programs can be created in Unix-like fashion using gnu utilities. Note that symbolic links are use in the file system to point to actual AS/400 `*module`, `*pgm` and `*srvpgm` objects in libraries.
- Download the tar file (unix version) to the AS/400 (using a mapped drive), and decompress and `untar` the source. We have had difficulty with the tar command on AS/400. This is under investigation. If you have trouble, we recommend the following work around:

```
qsh:
gunzip -d <tar file.gz>
pax -r -f <uncompressed tar file>
```

- Create AS400 target library. This library will be the target for the resulting modules and Xerces C++ service program. You will specify this library on the `OUTPUTDIR` environment variable in step 4
- Set up the following environment variables in your build process (use `ADDENVVAR` or `WRKENVVAR CL` commands):

```
XERCESCROOT - <the full path to your Xerces C++ sources>
PLATFORM  - 'OS400'
MAKE   - '/usr/bin/gmake'
OUTPUTDIR  - <identifies target as400 library for *module, *pgm and *srvpgm
objects>
ICUROOT - (optional if using ICU)  <the path of your ICU includes>
```

- Add `QCXXN`, to your build process library list. This results in the resolution of `CRTCPPMOD` used by the `icc` compiler.
- The runConfigure instruction below uses `'egrep'`. This is not on the AS/400 but you can create it by doing the following: `edtf '/usr/bin/egrep'` with the following source:

```
#!/usr/bin/sh
/usr/bin/grep -e "$@"
```

You may want to put the environment variables and library list setup instructions in a `CL` program so you will not forget these steps during your build.

Configure

To configure the make files for an AS/400 build do the following:

```
qsh
cd <full path to Xerces C++>/src
runConfigure -p os400 -x icc -c icc -m MsgFile -t Iconv400
```

Troubleshooting:

```
error: configure: error: installation or configuration problem:
C compiler cannot create executables.
```

If during `runConfigure` you see the above error message, it can mean one of two things. Either `QCXXN` is not on your library list **OR** the `runConfigure` cannot create the temporary modules `CONFTest1`, etc) it uses to test out the compiler options. The second reason happens because the test modules already exist from a previous run of `runConfigure`. To correct the problem, do the following:

```
DLTMOD <your OUTPUTDIR library>/CONFT* and
DLTPGM your <OUTPUTDIR library>/CONFT*
```

Build

```
qsh
gmake -e
```

The above gmake will result in a service program being created in your specified library and a symbolic link to that service program placed in <path to Xerces C++/lib>. You can either bind your XML application programs directly to the parser's service program via the `BNDSRVPGM` option on the `CRTPGM` or `CRTSRVPGM` command or you can specify a binding directory on your `icc` command. To specify an archive file to bind to, use the `-L, -l` binding options on icc. An archive file on AS/400 is a binding directory. To create an archive file, use `qar` command. (see the gnu facilities write up).

After building the Xerces C++ service program, create a binding directory by doing the following (note, this binding directory is used when building the samples):

```
qsh
cd <full path to Xerces C++>/lib>
qar -cuv libxercesc1_1.a *.o
command = CRTBNDDIR BNDDIR(yourlib/libxercesc) TEXT('/yourlib/Xerces
C++/lib/libxercesc1_1.a')
command = ADDBNDDIRE BNDDIR(yourlib/libxercesc) OBJ((yourlib/LIBXERCESC *SRVPGM)
)
```

Troubleshooting:

If you are on a V4R3 system, you will get a bind problem `'descriptor QlgCvtTextDescToDesc not found'` using Iconv400. On V4R3 the system doesn't automatically pick up the `QSYS/QLGUSR` service program for you when resolving this function. This is not the case on V4R4. To fix this, you can either manually create the service program after creating all the resulting modules in your <OUTPUTDIR> library or you can create a symbolic link to a binding directory that points to the `QLGUSR` service program and then specify an additional `-L, -l` on the `EXTRA_LINK_OPTIONS` in `Makefile.incl`. See the `ln` and `qar` function in the gnu utilities.

To build for transcoder ICU:

1. Make sure you have an `ICUROOT` path set up so that you can find the ICU header files (usually `/usr/local`)
2. Make sure you have created a binding directory (symbolic link) in the file system so that you can

bind the Xerces C++ service program to the ICU service program and specify that on the `EXTRA_LINK_OPTIONS` in `src/Makefile.incl` (usually the default is a link in `/usr/local/lib`).

Creating AS400 XML parser message file:

As specified earlier, the `-m` MsgFile support on the `runConfigure` enable the parser messages to be pulled from an AS/400 message file. To view the source for creating the message file and the XML parser messages, see the following stream file:

```
EDTF <full path to Xerces C++>/src/util/MsgLoaders/MsgFile/CrtXMLMsgs
```

In the prolog of `CrtXMLMsgs` there are instructions to create the message file:

1. Use the `CPYFRMSTMF` to copy the CL source to an AS/400 source physical file. Note that the target source file needs to have record length of about 200 bytes to avoid any truncation.
2. Create the CL program to create the message file and add the various message descriptions
3. Call the CL program, providing the name of the message file (use `QXMLMSG` as default) and a library (this can be any library, including any product library in which you wish to embed the xml parser)

Note that the Xerces C++ source code for resolving parser messages is using by default message file `QXMLMSG, *LIBL`. If you want to change either the message file name or explicitly qualify the library to match your product needs, you must edit the following `.cpp` files prior to your build.

```
<full path to Xerces C++>/src/util/MsgLoaders/MsgFile/MsgLoader.cpp
<full path to Xerces C++>/src/util/Platforms/OS400/OS400PlatformUtils.cpp
```

Troubleshooting:

If you are using the parser and are failing to get any message text for error codes, it may be because of the `*LIBL` resolution of the message file.

Building Samples on AS/400

```
qsh
cd <full path to Xerces C++>/samples
runConfigure -p os400 -x icc -c icc
gmake -e
```

Troubleshooting:

If you take a `'sed'` error, while trying to make the samples. This is an AS400 anomaly having to do with certain new line character and the `sed` function. A temporary work around is to use `EDTF` on the configure stream file (`../samples/configure`) and delete the following line near the bottom: `s%@DEFS@%$DEFS%g`.

Building Xerces C++ on Macintosh

The Xerces C++ Mac port has the key following attributes:

1. Built atop CoreServices APIs and a limited number of Carbon APIs; supports builds for both Mac OS Classic, Carbon, and Mac OS X systems.
2. Has a Mac OS native transcoder that utilizes the built-in Mac OS Unicode converter [MacOSUnicodeConverter].
3. Has a Mac OS native netaccessor that utilizes the built-in Mac OS URLAccess routines [MacOSURLAccess].
4. Supports builds from Metroworks CodeWarrior, Apple Project Builder, and Mac OS X shell.

Using Xerces C++ with CodeWarrior

Xerces C++ and CodeWarrior:

Xerces C++ may be built with CodeWarrior under Mac OS Classic or Mac OS X. Since the Xerces C++ code contains some files with very long names, and CodeWarrior does not yet support use of files with such long names, the installation in this case is somewhat involved.

Installing Xerces C++ for use with CodeWarrior:

For compatibility with CodeWarrior, it is necessary to adjust some of the file names (and referencing include statements). To do this, it is necessary to perform the following steps on a unix (or Mac OS X) machine that has support for long file names (a Windows machine may also work):

- Retrieve Xerces C++ from CVS, or untar a packaged build. Note that these steps should not be performed in a Classic Mac OS environment, as file names would then be mangled at this point!
- Xerces C++ comes with a tool that will shorten file names as appropriate, and fix up referencing include statements. Duplicate the file Projects/MacOS/ShortenFiles.pl to the xercesc main directory (the same directory that contains the Projects directory). Executing this perl script from this location will create a new directory MacSrc that contains patched up versions of files from the src directory.

```
cd <xercescroot>
cp Projects/MacOS/ShortenFiles.pl .
perl ShortenFiles.pl
```

- The source files will likely not now have proper Mac OS type/creator attribution. CodeWarrior badly wants this to be correct. So set the type/creator of these files somehow. The following should work from Mac OS X (but if you're not going to keep building on a Mac OS X machine, you may well need to perform this step in some other way once you get the files onto your classic machine).

```
find . \( -name "*.c" -or -name "*.h" -or -name "*.cpp" -or -name "*.hpp" -or \
-name "*.xml" \) -print0 | xargs -0 /Developer/Tools/SetFile -c CWIE -t TEXT
```

- Move the entire directory structure to your Mac OS machine.

Building Xerces C++ with CodeWarrior:

- Run CodeWarrior (tested with latest CW Pro 6.2).
- Import the project Projects/MacOS/CodeWarrior/XercesLib/XercesLib.mcp.xml, saving it back out to the same directory as XercesLib.mcp.
- This project contains five build targets that build all combinations of classic, carbon, debug, and release versions, with an all target that builds all of these. Build any or all of these.
- Note that the Carbon targets contain an access path for a Carbon Support folder in the compiler folder. Up-to-date Apple headers and libraries are required. Either create a Carbon Support folder with recent headers and libraries or, if your MacOS Support folder is up to date, point the access path to this, or make an alias to it called "Carbon Support".

Building Xerces C++ Samples with CodeWarrior:

A CodeWarrior project is included that builds the DOMPrint sample. This may be used as an example from which to build additional sample projects. Please read the following important notes:

- Once again, it is required that you import the .xml version of the project file, and save it back out.
- The Xerces C++ sample programs are written to assume a command line interface. To avoid making Macintosh-specific changes to these command line programs, we have opted to instead require that you make a small extension to your CodeWarrior runtime that supports such command line programs. Please read and follow the usage notes in XercesSampleSupport/XercesSampleStartupFragment.c.

Building Xerces C++ with Project Builder

Projects are included to build the Xerces C++ library and DOMPrint sample under Apple's Project Builder for Mac OS X. The following notes apply:

- Since you are running under Mac OS X, and if you are not also performing CodeWarrior builds, it is not necessary to shorten file names or set the type/creator codes as required for CodeWarrior.
- The Project Builder project builds XercesLib as the framework Xerces.framework. This framework, however, does not currently include a correct set of public headers. Any referencing code must have an include path directive that points into the Xerces C++ src directory.
- The DOMPrint project illustrates one such usage of the Xerces.framework.

Building Xerces C++ from the Mac OS X command line

Support for Mac OS X command line builds is now included in the standard "unix" Xerces C++ build infrastructure.

- In general, the Mac OS X command line build follows the generic unix build instructions. You need to set your XERCESCROOT environment variable, `./runConfigure`, and `make`.

```
setenv XERCESCROOT "<directory>"
cd src
./runConfigure -p macosx -n native
make
```

- Similar instructions apply to build the samples and tests, though the `-n` flag is not used in these cases:

```
cd samples
./runConfigure -p macosx
make
```

Special usage information for Xerces C++ on the Macintosh

File Path Specification

Apart from the build instructions, above, the most important note about use of Xerces C++ on the Macintosh is that Xerces C++ expects all filename paths to be specified in unix syntax. If running natively under a Mac OS X system, this path will be the standard posix path as expected by the shell. The easiest means of creating and interpreting these paths will be through the routines `XMLCreateFullPathFromFSRef` and `XMLParsePathToFSRef` as declared in the file `MacOSPlatformUtils.hpp`. FSSpec variants of these routines are also supplied.

Mac OS Version Compatibility

Xerces C++ requires that several key components of the Mac OS be relatively up to date. It should be readily compatible with any system above Mac OS 9.0. Compatibility with earlier systems may perhaps be achieved if you can install appropriate components.

Required components are:

- Unicode Converter and Text Encoding Converter. These provide the base transcoding service used to support Xerces C++ transcoding requirements.

Optional components are:

- URLAccess. Provides NetAccessor support to Xerces C++ for use in fetching network referenced entities. If URLAccess is not installed, any such references will fail; the absence of URLAccess, however, will not in itself prevent Xerces C++ from running.
- Multiprocessing library. Provides mutual exclusion support. Once again, the routines will back down gracefully if Multiprocessing support is not available.
- HFS+ APIs. If HFS+ APIs are available, all file access is performed using the HFS+ fork APIs to support long file access, and to support long unicode compliant file names. In the absence of HFS+ APIs, classic HFS APIs are used instead.

Other Build Instructions

Building Xerces C++ with ICU using bundled Perl scripts on Windows

As mentioned earlier, Xerces C++ may be built in stand-alone mode using native encoding support and also using ICU where you get support over 180 different encodings. ICU stands for International Components for Unicode and is an open source distribution from IBM. You can get ICU libraries [11] from IBM's developerWorks site [18] or go to the ICU download page [19] directly.

> ***Note: Important:*** *Please remember that* ***ICU and Xerces C++ must be built with the same compiler****, preferably with the same version. You cannot for example, build ICU with a threaded version of the xlC compiler and build Xerces C++ with a non-threaded one.*

There are two options to build Xerces C++ with ICU. One is to use the MSDEV GUI environment, and the other is to invoke the compiler from the command line.

Using, the GUI environment, requires one to edit the project files. Here, we will describe only the second option. It involves using the perl script '`packageBinaries.pl`'.

Prerequisites:

- Perl 5.004 or higher
- Cygwin tools or MKS Toolkit
- zip.exe

Extract Xerces C++ source files from the .zip archive using WinZip, say in the root directory (an arbitrary drive x:). It should create a directory like '`x:\xerces-c-src1_5_1`'.

Extract the ICU files, using WinZip, in root directory of the disk where you have installed Xerces C++, sources. After extraction, there should be a new directory '`x:\icu`' which contains all the ICU source files.

Start a command prompt to get a new shell window. Make sure you have perl, cygwin tools (`uname`, `rm`, `cp`, ...), and `zip.exe` somewhere in the path. Next setup the environment for MSVC using `VCVARS32.BAT`' or a similar file. Then at the prompt enter:

```
    set XERCESCROOT=x:\xerces-c-src1_5_1
    set ICUROOT=x:\icu
    cd x:\xerces-c-src1_5_1\scripts
    perl packageBinaries.pl -s x:\xerces-c-src1_5_1 -o x:\temp\xerces-c1_5_1-win32
   -t icu
```

(Match the source directory to your system; the target directory can be anything you want.)

If everything is setup right and works right, then you should see a binary drop created in the target directory specified above. This script will build both ICU and Xerces C++, copy the files (relevant to the binary drop) to the target directory.

For a description of options available, you can enter:

```
    perl packageBinaries.pl
```

Building Xerces C++ COM Wrapper on Windows

To build the COM module for use with XML on Windows platforms, you must first set up your machine appropriately with necessary tools and software modules and then try to compile it. The end result is an additional library that you can use along with the standard Xerces C++ for writing VB templates or for use with IE 5.0 using JavaScript.

Setting up your machine for COM

To build the COM project you will need to install the MS PlatformSDK. Some of the header files we use don't come with Visual C++ 6.0. You may download it from Microsoft's Website at http://www.microsoft.com/msdownload/platformsdk/setuplauncher.htm or directly FTP it from ftp://ftp.microsoft.com/developr/PlatformSDK/April2000/Msi/WinNT/x86/InstMsi.exe.

The installation is huge, but you don't need most of it. So you may do a **custom install** by just selecting "Build Environment" and choosing the required components. First select the top level Platform SDK. Then click the down arrow and make all of the components unavailable. Next open the "Build Environment" branch and select only the following items:

- Win32 API
- Component Services
- Web Services - Internet Explorer

Important: When the installation is complete you need to update VC6's include path to include `..\platformsdk\include\atl30`. You do this by choosing "Tools - > Options - > Directories". This path should be placed *second* after the normal PlatformSDK include. You change the order of the paths by clicking the up and down arrows.

> ***Note:*** *The order in which the directories appear on your path is important. Your first include path should be* `..\platformsdk\include`*. The second one should be* `..\platformsdk\include\atl30`*.*

Building COM module for Xerces C++

Once you have set up your machine, build Xerces C++ COM module by choosing the project named 'xml4com' inside the workspace. Then select your build mode to be **xml4com - Win32 Release MinDependency**. Finally build the project. This will produce a DLL named `xerces-com.dll` which needs to be present in your path (on local machine) before you can use it.

Testing the COM module

There are some sample test programs in the `test/COMTest` directory which show examples of navigating and searching an XML tree using DOM. You need to browse the HTML files in this directory using IE 5.0. Make sure that your build has worked properly, specially the registration of the ActiveX controls that happens in the final step.

You may also want to check out the NIST DOM test suite at http://xw2k.sdct.itl.nist.gov/BRADY/DOM/. You will have to modify the documents in the NIST suite to load the Xerces COM object instead of the MSIE COM object.

Building User Documentation

The user documentation (this very page that you are reading on the browser right now), was generated using an XML application called StyleBook. This application makes use of Xerces-J and Xalan to create the HTML file from the XML source files. The XML source files for the documentation are part of the Xerces C++ module. These files reside in the `doc` directory.

Pre-requisites for building the user documentation are:

- JDK 1.2.2 (or later).
- Xerces-J 1.0.1.**bundled**
- Xalan-J 0.19.2.**bundled**
- Stylebook 1.0-b2. **bundled**
- The Apache Style files (dtd's and .xsl files).**bundled**

Invoke a command window and setup PATH to include the JDK 1.2.2 bin directory

Next, cd to the Xerces C++ source drop root directory, and enter

- Under Windows:

```
createDocs
```

- Under Unix's:

```
sh createDocs.bat
```

This should generate the .html files in the 'doc/html' directory.

I wish to port Xerces to my favourite platform. Do you have any suggestions?

All platform dependent code in Xerces has been isolated to a couple of files, which should ease the porting effort. Here are the basic steps that should be followed to port Xerces.

1. The directory `src/util/Platforms` contains the platform sensitive files while `src/util/Compilers` contains all development environment sensitive files. Each operating system has a file of its own and each development environment has another one of its own too.

   As an example, the Win32 platform as a `Win32Defs.hpp` file and the Visual C++ environment has a `VCPPDefs.hpp` file. These files set up certain define tokens, typedefs, constants, etc... that will drive the rest of the code to do the right thing for that platform and development environment. AIX/CSet have their own `AIXDefs.hpp` and `CSetDefs.hpp` files, and so on. You should create new versions of these files for your platform and environment and follow the comments in them to set up your own. Probably the comments in the Win32 and Visual C++ will be the best to follow, since that is where the main development is done.
2. Next, edit the file `XercesDefs.hpp`, which is where all of the fundamental stuff comes into the system. You will see conditional sections in there where the above per-platform and per-environment headers are brought in. Add the new ones for your platform under the appropriate conditionals.
3. Now edit `AutoSense.hpp`. Here we set canonical Xerces internal `#define` tokens which indicate the platform and compiler. These definitions are based on known platform and compiler defines.

   `AutoSense.hpp` is included in `XercesDefs.hpp` and the canonical platform and compiler settings thus defined will make the particular platform and compiler headers to be the included at compilation.

   It might be a little tricky to decipher this file so be careful. If you are using say another compiler on Win32, probably it will use similar tokens so that the platform will get picked up already using what is already there.
4. Once this is done, you will then need to implement a version of the *platform utilities* for your platform. Each operating system has a file which implements some methods of the XMLPlatformUtils class, specific to that operating system. These are not terribly complex, so it should not be a lot of work. The Win32 verions is called `Win32PlatformUtils.cpp`, the AIX version is `AIXPlatformUtils.cpp` and so on. Create one for your platform, with the correct name, and empty out all of the implementation so that just the empty shells of the methods are there (with dummy returns where needed to make the compiler happy.) Once you've done that, you can start to get it to build without any real implementation.
5. Once you have the system building, then start implementing your own platform utilties methods. Follow the comments in the Win32 version as to what they do, the comments will be improved in subsequent versions, but they should be fairly obvious now. Once you have these implementations done, you should be able to start debugging the system using the demo programs.

Other concerns are:

- Does ICU compile on your platform? If not, then you'll need to create a transcoder implementation that uses your local transcoding services. The Iconv transcoder should work for you, though perhaps with some modifications.

- What message loader will you use? To get started, you can use the "in memory" one, which is very simple and easy. Then, once you get going, you may want to adapt the message catalog message loader, or write one of your own that uses local services.

That is the work required in a nutshell!

What should I define XMLCh to be?

XMLCh should be defined to be a type suitable for holding a utf-16 encoded (16 bit) value, usually an `unsigned short`.

All XML data is handled within Xerces C++ as strings of XMLCh characters. Regardless of the size of the type chosen, the data stored in variables of type XMLCh will always be utf-16 encoded values.

Unlike XMLCh, the encoding of wchar_t is platform dependent. Sometimes it is utf-16 (AIX, Windows), sometimes ucs-4 (Solaris, Linux), sometimes it is not based on Unicode at all (HP/UX, AS/400, system 390).

Some earlier releases of xerce-c defined XMLCh to be the same type as wchar_t on most platforms, with the goal of making it possible to pass XMLCh strings to library or system functions that were expecting wchar_t paramters. This approach has been abandonded because of

- Portability problems with any code that assumes that the types of XMLCh and wchar_t are compatible
- Excessive memory usage, especially in the DOM, on platforms with 32 bit wchar_t.
- utf-16 encoded XMLCh is not always compatible with ucs-4 encoded wchar_t on Solaris and Linux. The problem occurs with Unicode characters with values greater than 64k; in ucs-4 the value is stored as a single 32 bit quatity. With utf-16, the value will be stored as a "surrogate pair" of two 16 bit values. Even with XMLCh equated to wchar_t, xerces will still create the utf-16 encoded surrogate pairs, which are illegal in ucs-4 encoded wchar_t strings.

Where can I look for more help?

If you have read this page, followed the instructions, and still cannot resolve your problem(s), there is more help. You can find out if others have solved this same problem before you, by checking the Apache XML mailing list archives at http://archive.covalent.net [20] and the Bugzilla [21] Apache bug database.

4
API Documentation

API Docs for SAX and DOM

Xerces C++ is packaged with the API documentation for SAX and DOM, the two most common programming interfaces for XML. The most common framework classes have also been documented.

Xerces C++ DOM is an implementation of the Document Object Model (Core) Level 1 [3] as defined in the W3C Recommendation of 1 October, 1998; and Document Object Model (Core) Level 2 [4] as defined in the W3C Recommendation of 13 November, 2000. For a complete understanding of how the Xerces C++ APIs work, we recommend you to read these documents.

Xerces C++ SAX is an implementation of the SAX 1.0/2.0 [6] specification. You are encouraged to read this document for a better understanding of the SAX API in Xerces C++.

See the **Xerces C++ API documentation.** for more details.

> ***Note:*** *The API documentation is automatically generated using doxygen [22] and GraphViz [23] .*

5
Xerces C++ Samples

Building the Samples

Xerces C++ comes packaged with ten sample applications that demonstrate salient features of the parser using simple applications written on top of the SAX and DOM APIs provided by the parser.

Once you have set up your PATH variable, you can run the samples by opening a command window (or your shell prompt for UNIX environments). Sample XML data files are provided in the samples/data directory.

The installation process for the samples is same on all UNIX platforms. Note that **runConfigure** is just a helper script and you are free to use **./configure** with the correct parameters to make it work on any platform-compiler combination of your choice. The script needs the following parameters:

```
Usage: runConfigure "options"
       where options may be any of the following:
       -p <platform> (accepts 'aix', 'linux', 'solaris', 'hp-10', 'hp-11')
       -c <C compiler name> (e.g. gcc, xlc_r, cc or aCC)
       -x <C++ compiler name> (e.g. g++, xlC_r, CC or aCC)
       -d (specifies that you want to build debug version)
       -h (get help on the above commands)
```

Note: NOTE:*The code samples in this section assume that you are are working on the Linux binary drop. If you are using some other UNIX flavor, please replace '-linux' with the appropriate platform name in the code samples.*

Building the Samples for OS2

Building the Xerces C++ samples using IBM Visual Age C++ Professional 4.0 for OS/2 (VAC++).

- In the `XercesCSrcInstallDir;\samples\Projects\OS2\VACPP40` directory, find and edit the VAC++ configuration file `basedir.icc`.
- All of the directories used to build the samples are defined in `basedir.icc`. You need to edit the directories to match your system. Here are the directories you need to assign: SRC_DIR -- `XercesCSrcInstallDir;` This is where VAC++ should look to find the samples directories containing the source files. BASE_DIR -- The install directory `XercesCSrcInstallDir;`. VAC++ will store the compiled samples in the `bin` directory under BASE_DIR. It will also look for the `xerces-c.lib` file in the `lib` directory under BASE_DIR. Other directories are set based on these two. You can choose to override them if you wish.
- Save `basedir.icc`
- Start the Command Line in the VAC++ folder.
- Navigate to the `XercesCSrcInstallDir;\samples\Projects\OS2\VACPP40` directory.

- Run `bldsamples.cmd`
- When `build.cmd` finishes, review the file `compiler.errors`. This file should contain only informational messages, almost all complaining about constant values in comparisons.
- You should now have several executable files.

Rebuilding the Configuration Files

Although it shouldn't be necessary, if you want to rebuild the VAC++ configuration files, you'll need to have Object Rexx running on your system:

- If you are not currently running Object Rexx, run the SWITCHRX command from a command line, answer "yes" to switching to Object Rexx, and follow the instructions to reboot. (Note: You can switch back to "Classic Rexx" by running SWITCHRX again. But you probably won't need to switch back since Object Rexx runs almost 100% of Classic Rexx programs.)
- In the Projects\OS2\VACPP40 directory, run genICC.cmd. This builds the VAC++ configuration files for the samples you have on your system.
- Go to the first step above in the "Building asmples for OS/2" section.

Running the Samples

The sample applications are dependent on the Xerces C++ shared library (and could also depend on the ICU library if you built Xerces C++ with ICU). Therefore, on Windows platforms you must make sure that your `PATH` environment variable is set properly to pick up these shared libraries at runtime.

On UNIX platforms you must ensure that *LIBPATH* environment variable is set properly to pick up the shared libraries at runtime. (UNIX gurus will understand here that *LIBPATH* actually translates to **LD_LIBRARY_PATH** on Solaris and Linux, **SHLIB_PATH** on HP-UX and stays as **LIBPATH** on AIX).

To set you LIBPATH (on AIX for example), you would type:

```
    export LIBPATH=xerces-c1_5_1/lib:$LIBPATH
```

Xerces C++ Samples

- SAXCount

  SAXCount counts the elements, attributes, spaces and characters in an XML file.
- SAXPrint

  SAXPrint parses an XML file and prints it out.
- DOMCount

  DOMCount counts the elements in a XML file.
- DOMPrint

  DOMPrint parses an XML file and prints it out.
- MemParse

  MemParse parses XML in a memory buffer, outputing the number of elements and attributes.
- Redirect

  Redirect redirects the input stream for external entities.
- PParse

  PParse demonstrates progressive parsing.
- StdInParse

  StdInParse demonstrates streaming XML data from standard input.
- EnumVal

  EnumVal shows how to enumerate the markup decls in a DTD Validator.
- CreateDOMDocument

CreateDOMDocument creates a DOM tree in memory from scratch.

- SAX2Count

  SAX2Count counts the elements, attributes, spaces and characters in an XML file.

- SAX2Print

  SAX2Print parses an XML file and prints it out.

- IDOMCount

  IDOMCount counts the elements in a XML file.

- IDOMPrint

  IDOMPrint parses an XML file and prints it out.

Xerces C++ Sample 1: SAXCount

SAXCount

SAXCount is the simplest application that counts the elements and characters of a given XML file using the (event based) SAX API.

Building on Windows

Load the xerces-c1_5_1-win32\samples\Projects\Win32\VC6\samples.dsw Microsoft Visual C++ workspace inside your MSVC IDE. Then build the project marked SAXCount.

Building on UNIX

```
cd xerces-c1_5_1-linux/samples
./runConfigure -p<platform> -c<C_compiler> -x<C++_compiler>
cd SAXCount
gmake
```

This will create the object files in the current directory and the executable named SAXCount in 'xerces-c1_5_1-linux/bin' directory.

To delete all the generated object files and executables, type

```
gmake clean
```

Running SAXCount

The SAXCount sample parses an XML file and prints out a count of the number of elements in the file. To run SAXCount, enter the following

```
SAXCount <XML File>
```

The following parameters may be set from the command line

```
Usage:
    SAXCount [options] <XML file>

Options:
    -v=xxx      Validation scheme [always | never | auto*]
    -n          Enable namespace processing. Defaults to off.
    -s          Enable schema processing. Defaults to off.

This program prints the number of elements, attributes,
white spaces and other non-white space characters in the input file.

  * = Default if not provided explicitly
```

-v=always will force validation

-v=never will not use any validation

-v=auto will validate if a DOCTYPE declaration is present in the XML document

Here is a sample output from SAXCount

```
cd xerces-c1_5_1-linux/samples/data
SAXCount -v=always personal.xml
personal.xml: 60 ms (37 elems, 12 attrs, 134 spaces, 134 chars)
```

Running SAXCount with the validating parser gives a different result because ignorable white-space is counted separately from regular characters.

```
SAXCount -v=never personal.xml
personal.xml: 10 ms (37 elems, 12 attrs, 0 spaces, 268 chars)
```

Note that the sum of spaces and chracters in both versions is the same.

> ***Note:*** *The time reported by the program may be different depending on your machine processor.*

Xerces C++ Sample 2: SAXPrint

SAXPrint

SAXPrint uses the SAX APIs to parse an XML file and print it back. Do note that the output of this sample is not exactly the same as the input (in terms of whitespaces, first line), but the output has the same information content as the input.

Building on Windows

Load the xerces-c1_5_1-win32\samples\Projects\Win32\VC6\samples.dsw Microsoft Visual C++ workspace inside your MSVC IDE. Then build the project marked SAXPrint.

Building on UNIX

```
cd xerces-c1_5_1-linux/samples
./runConfigure -p<platform> -c<C_compiler> -x<C++_compiler>
cd SAXPrint
gmake
```

This will create the object files in the current directory and the executable named SAXPrint in 'xerces-c1_5_1-linux/bin' directory.

To delete all the generated object files and executables, type

```
gmake clean
```

Running SAXPrint

The SAXPrint sample parses an XML file and prints out the contents again in XML (some loss occurs). To run SAXPrint, enter the following

```
SAXPrint <XML file>
```

The following parameters may be set from the command line

```
Usage: SAXPrint [options] file
This program prints the data returned by the various SAX
handlers for the specified input file. Options are NOT case
sensitive.

Options:
    -u=xxx      Handle unrepresentable chars [fail | rep | ref*]
    -v=xxx      Validation scheme [always | never | auto*]
    -n          Enable namespace processing.
    -s          Enable schema processing.
    -x=XXX      Use a particular encoding for output (LATIN1*).
    -?          Show this help

  * = Default if not provided explicitly

The parser has intrinsic support for the following encodings:
    UTF-8, USASCII, ISO8859-1, UTF-16[BL]E, UCS-4[BL]E,
    WINDOWS-1252, IBM1140, IBM037
```

-u=fail will fail when unrepresentable characters are encountered

-u=rep will replace with the substitution character for that codepage

-u=ref will report the character as a reference

-v=always will force validation

-v=never will not use any validation

-v=auto will validate if a DOCTYPE declaration is present in the XML document

Here is a sample output from SAXPrint

```
cd xerces-c1_5_1-linux/samples/data
SAXPrint -v=always personal.xml

<?xml version="1.0" encoding="LATIN1"?>
<personnel>

  <person id="Big.Boss">
    <name><family>Boss</family> <given>Big</given></name>
    <email>chief@foo.com</email>
    <link subordinates="one.worker two.worker three.worker
                        four.worker five.worker"></link>
  </person>

  <person id="one.worker">
    <name><family>Worker</family> <given>One</given></name>
    <email>one@foo.com</email>
    <link manager="Big.Boss"></link>
  </person>

  <person id="two.worker">
    <name><family>Worker</family> <given>Two</given></name>
    <email>two@foo.com</email>
    <link manager="Big.Boss"></link>
  </person>

  <person id="three.worker">
    <name><family>Worker</family> <given>Three</given></name>
    <email>three@foo.com</email>
    <link manager="Big.Boss"></link>
  </person>

  <person id="four.worker">
    <name><family>Worker</family> <given>Four</given></name>
    <email>four@foo.com</email>
    <link manager="Big.Boss"></link>
  </person>

  <person id="five.worker">
    <name><family>Worker</family> <given>Five</given></name>
    <email>five@foo.com</email>
    <link manager="Big.Boss"></link>
  </person>

</personnel>
```

Note: *SAXPrint does not reproduce the original XML file. SAXPrint and DOMPrint produce different results because of the way the two APIs store data and capture events.*

Xerces C++ Sample 3: DOMCount

DOMCount

DOMCount uses the provided DOM API to parse an XML file, constructs the DOM tree and walks through the tree counting the elements (using just one API call).

Building on Windows

Load the xerces-c1_5_1-win32\samples\Projects\Win32\VC6\samples.dsw Microsoft Visual C++ workspace inside your MSVC IDE. Then build the project marked DOMCount.

Building on UNIX

```
cd xerces-c1_5_1-linux/samples
./runConfigure -p<platform> -c<C_compiler> -x<C++_compiler>
cd DOMCount
gmake
```

This will create the object files in the current directory and the executable named DOMCount in ' xerces-c1_5_1-linux/bin' directory.

To delete all the generated object files and executables, type

```
gmake clean
```

Running DOMCount

The DOMCount sample parses an XML file and prints out a count of the number of elements in the file. To run DOMCount, enter the following

```
DOMCount <XML file>
```

The following parameters may be set from the command line

```
Usage:
    DOMCount [-v -n] {XML file}

This program invokes the XML4C DOM parser, builds
the DOM tree, and then prints the number of elements
found in the input XML file.

Options:
    -v=xxx      Validation scheme [always | never | auto*]
    -n          Enable namespace processing. Defaults to off.
    -s          Enable schema processing. Defaults to off.

  * = Default if not provided explicitly
```

-v=always will force validation

-v=never will not use any validation

-v=auto will validate if a DOCTYPE declaration is present in the XML document

Here is a sample output from DOMCount

```
cd xerces-c1_5_1-linux/samples/data
DOMCount -v=always personal.xml
```

```
    personal.xml: 20 ms (37 elems)
```

The output of both versions should be same.

> ***Note:*** *The time reported by the system may be different, depending on your processor type.*

Xerces C++ Sample 4: DOMPrint

DOMPrint

DOMPrint parses an XML file, constructs the DOM tree, and walks through the tree printing each element. It thus dumps the XML back (output same as SAXPrint).

Building on Windows

Load the xerces-c1_5_1-win32\samples\Projects\Win32\VC6\samples.dsw Microsoft Visual C++ workspace inside your MSVC IDE. Then build the project marked DOMPrint.

Building on UNIX

```
cd xerces-c1_5_1-linux/samples
./runConfigure -p<platform> -c<C_compiler> -x<C++_compiler>
cd DOMPrint
gmake
```

This will create the object files in the current directory and the executable named DOMPrint in 'xerces-c1_5_1-linux/bin' directory.

To delete all the generated object files and executables, type

```
gmake clean
```

Running DOMPrint

The DOMPrint sample parses an XML file, using either a validating or non-validating DOM parser configuration, builds a DOM tree, and then walks the tree and outputs the contents of the nodes in a 'canonical' format. To run DOMPrint, enter the following:

```
DOMPrint <XML file>
```

The following parameters may be set from the command line

```
Usage: DOMPrint [options] file

This program invokes the Xerces C++ DOM parser and builds the DOM
tree. It then traverses the DOM tree and prints the contents
of the tree. Options are NOT case sensitive.

Options:
    -e          create entity reference nodes. Default is no expansion.
    -u=xxx      Handle unrepresentable chars [fail | rep | ref*]
    -v=xxx      Validation scheme [always | never | auto*]
    -n          Enable namespace processing. Default is off.
    -s          Enable schema processing. Default is off.
    -x=XXX      Use a particular encoding for output. Default is
                the same encoding as the input XML file. UTF-8 if
                input XML file has not XML declaration.
    -?          Show this help (must be the only parameter)

  * = Default if not provided explicitly

The parser has intrinsic support for the following encodings:
```

```
        UTF-8, USASCII, ISO8859-1, UTF-16[BL]E, UCS-4[BL]E,
        WINDOWS-1252, IBM1140, IBM037
```

-u=fail will fail when unrepresentable characters are encountered

-u=rep will replace with the substitution character for that codepage

-u=ref will report the character as a reference

-v=always will force validation

-v=never will not use any validation

-v=auto will validate if a DOCTYPE declaration is present in the XML document

Here is a sample output from DOMPrint

```
    cd xerces-c1_5_1-linux/samples/data
    DOMPrint -v personal.xml

    <?xml version="1.0" encoding="iso-8859-1"?>

    <!DOCTYPE personnel SYSTEM "personal.dtd">
    <!-- @version: -->
    <personnel>

    <person id="Big.Boss">
      <name><family>Boss</family> <given>Big</given></name>
      <email>chief@foo.com</email>
      <link subordinates="one.worker two.worker three.worker
                          four.worker five.worker"></link>
    </person>

    <person id="one.worker">
      <name><family>Worker</family> <given>One</given></name>
      <email>one@foo.com</email>
      <link manager="Big.Boss"></link>
    </person>

    <person id="two.worker">
      <name><family>Worker</family> <given>Two</given></name>
      <email>two@foo.com</email>
      <link manager="Big.Boss"></link>
    </person>

    <person id="three.worker">
      <name><family>Worker</family> <given>Three</given></name>
      <email>three@foo.com</email>
      <link manager="Big.Boss"></link>
    </person>

    <person id="four.worker">
      <name><family>Worker</family> <given>Four</given></name>
      <email>four@foo.com</email>
      <link manager="Big.Boss"></link>
    </person>
```

```
<person id="five.worker">
  <name><family>Worker</family> <given>Five</given></name>
  <email>five@foo.com</email>
  <link manager="Big.Boss"></link>
</person>

</personnel>
```

Note that DOMPrint does not reproduce the original XML file. DOMPrint and SAXPrint produce different results because of the way the two APIs store data and capture events.

Xerces C++ Sample 5: MemParse

MemParse

MemParse uses the Validating SAX Parser to parse a memory buffer containing XML statements, and reports the number of elements and attributes found.

Building on Windows

Load the xerces-c1_5_1-win32\samples\Projects\Win32\VC6\samples.dsw Microsoft Visual C++ workspace inside your MSVC IDE. Then build the project marked MemParse.

Building on UNIX

```
cd xerces-c1_5_1-linux/samples
./runConfigure -p<platform> -c<C_compiler> -x<C++_compiler>
cd MemParse
gmake
```

This will create the object files in the current directory and the executable named `MemParse` in 'xerces-c1_5_1-linux/bin' directory.

To delete all the generated object files and executables, type

```
gmake clean
```

Running MemParse

This program uses the SAX Parser to parse a memory buffer containing XML statements, and reports the number of elements and attributes found.

```
MemParse [-v]
```

The -v option is used to invoke the Validating SAX Parser instead. When invoked with a validating parser:

```
cd xerces-c1_5_1-linux/samples/data
MemParse -v
```

The output is the following:

```
Finished parsing the memory buffer containing the following XML statements:

<?xml version='1.0' encoding='ascii'?>
<!DOCTYPE company [
<!ELEMENT company     (product,category,developedAt)>
<!ELEMENT product     (#PCDATA)>
<!ELEMENT category    (#PCDATA)>
<!ATTLIST category idea CDATA #IMPLIED>
<!ELEMENT developedAt (#PCDATA)>
]>

<company>
  <product>XML4C</product>
  <category idea='great'>XML Parsing Tools</category>
  <developedAt>
    IBM Center for Java Technology, Silicon Valley, Cupertino, CA
```

```
      </developedAt>
    </company>

    Parsing took 0 ms (4 elements, 1 attributes, 16 spaces, 95 characters).
```

Xerces C++ Sample 6: Redirect

Redirect

Redirect uses the SAX EntityResolver handler to redirect the input stream for external entities. It installs an entity resolver, traps the call to the external DTD file and redirects it to another specific file which contains the actual DTD.

Building on Windows

Load the xerces-c1_5_1-win32\samples\Projects\Win32\VC6\samples.dsw Microsoft Visual C++ workspace inside your MSVC IDE. Then build the project marked Redirect.

Building on UNIX

```
cd xerces-c1_5_1-linux/samples
./runConfigure -p<platform> -c<C_compiler> -x<C++_compiler>
cd Redirect
gmake
```

This will create the object files in the current directory and the executable named Redirect in 'xerces-c1_5_1-linux/bin' directory.

To delete all the generated object files and executables, type

```
gmake clean
```

Running Redirect

This program illustrates how a XML application can use the SAX EntityResolver handler to redirect the input stream for external entities. It installs an entity resolver, traps the call to the external DTD file and redirects it to another specific file which contains the actual DTD.

The program then counts and reports the number of elements and attributes in the given XML file.

```
Redirect <XML file>
```

Redirect is invoked as follows:

```
cd xerces-c1_5_1-linux/samples/data
Redirect personal.xml
```

The output is the following:

```
cd xerces-c1_5_1-linux/samples/data
Redirect personal.xml
personal.xml: 30 ms (37 elems, 12 attrs, 0 spaces, 268 chars)
```

External files required to run this sample are 'personal.xml', 'personal.dtd' and 'redirect.dtd', which are all present in the 'samples/data' directory. Make sure that you run redirect in the samples/data directory.

The 'resolveEntity' callback in this sample looks for an external entity with system id as 'personal.dtd'. When it is asked to resolve this particular external entity, it creates and returns a new InputSource for the file 'redirect.dtd'.

A real-world XML application can similarly do application specific processing when encountering external entities. For example, an application might want to redirect all references to entities outside of its domain to local cached copies.

Xerces C++ Sample 7: PParse

PParse

PParse demonstrates progressive parsing.

In this example, the programmer doesn't have to depend upon throwing an exception to terminate the parsing operation. Calling parseFirst() will cause the DTD to be parsed (both internal and external subsets) and any pre-content, i.e. everything up to but not including the root element. Subsequent calls to parseNext() will cause one more piece of markup to be parsed, and spit out from the core scanning code to the parser. You can quit the parse any time by just not calling parseNext() anymore and breaking out of the loop. When you call parseNext() and the end of the root element is the next piece of markup, the parser will continue on to the end of the file and return false, to let you know that the parse is done.

Building on Windows

Load the xerces-c1_5_1win32\samples\Projects\Win32\VC6\samples.dsw Microsoft Visual C++ workspace inside your MSVC IDE. Then build the project marked PParse.

Building on UNIX

```
cd xerces-c1_5_1-linux/samples
./runConfigure -p<platform> -c<C_compiler> -x<C++_compiler>
cd PParse
gmake
```

This will create the object files in the current directory and the executable named PParse in ' xerces-c1_5_1-linux/bin' directory.

To delete all the generated object files and executables, type:

```
gmake clean
```

Running PParse

The program looks for the first 16 elements of the XML file, and reports if successful.

```
PParse <XML file>
```

```
Usage: PParse [options] <file>

This sample program demonstrates the progressive parse capabilities of
the parser system. It allows you to do a scanFirst() call followed by
a loop which calls scanNext(). You can drop out when you've found what
ever it is you want. In our little test, our event handler looks for
16 new elements then sets a flag to indicate its found what it wants.
At that point, our progressive parse loop exits.

Options:
      -v=xxx        - Validation scheme [always | never | auto*]
      -n            - Enable namespace processing [default is off]
      -s            - Enable schema processing [default is off]
      -?            - Show this help (must be the only parameter)

  * = Default if not provided explicitly
```

The output is the following:

```
cd xerces-c1_5_1-linux/samples/data
PParse personal.xml

Got the required 16 elements.
```

Xerces C++ Sample 8: StdInParse

StdInParse

StdInParse demonstrates streaming XML data from standard input.

Building on Windows

Load the xerces-c1_5_1-win32\samples\Projects\Win32\VC6\samples.dsw Microsoft Visual C++ workspace inside your MSVC IDE. Then build the project marked StdInParse.

Building on UNIX

```
cd xerces-c1_5_1-linux/samples
./runConfigure -p<platform> -c<C_compiler> -x<C++_compiler>
cd StdInParse
gmake
```

This will create the object files in the current directory and the executable named StdInParse in ' xerces-c1_5_1-linux/bin' directory.

To delete all the generated object files and executables, type:

```
gmake clean
```

Running StdInParse

The StdInParse sample parses an XML file and prints out a count of the number of elements in the file. To run StdInParse, enter the following:

```
StdInParse < <XML file>
```

The following parameters may be set from the command line

```
Usage:
    StdInParse [options]
    -v=xxx      Validation scheme [always | never | auto]
    -n          Enable namespace processing. [default is off]
    -s          Enable schema processing. [default is off]
    -?          Show this help

  * = Default if not provided explicitly

This program allows you to redirect a file into the program
to be parsed. It will count the elements, characters, and
spaces and display these stats at the end
```

Here is a sample output from StdInParse:

```
cd xerces-c1_5_1-linux/samples/data
StdInParse < personal.xml
stdin: 60 ms (37 elems, 12 attrs, 0 spaces, 268 chars)
```

Note: *The time reported by the program may be different depending on your machine processor.*

Xerces C++ Sample 9: EnumVal

EnumVal

EnumVal shows how to enumerate the markup decls in a DTD Validator.

Building on Windows

Load the xerces-c1_5_1-win32\samples\Projects\Win32\VC6\samples.dsw Microsoft Visual C++ workspace inside your MSVC IDE. Then build the project marked EnumVal.

Building on UNIX

```
cd xerces-c1_5_1-linux/samples
./runConfigure -p<platform> -c<C_compiler> -x<C++_compiler>
cd EnumVal
gmake
```

This will create the object files in the current directory and the executable named EnumVal in ' xerces-c1_5_1-linux/bin' directory.

To delete all the generated object files and executables, type

```
gmake clean
```

Running EnumVal

This program parses a file, then shows how to enumerate the contents of the validator pools. To run EnumVal, enter the following

```
EnumVal <XML file>
```

Here is a sample output from EnumVal

```
cd xerces-c1_5_1-linux/samples/data
EnumVal personal.xml

ELEMENTS:
---------------------------
  Name: personnel
  Content Model: (person)+

  Name: person
  Content Model: (name,email*,url*,link?)
  Attributes:
    Name:id, Type: ID

  Name: name
  Content Model: (#PCDATA|family|given)*

  Name: email
  Content Model: (#PCDATA)*

  Name: url
  Content Model: EMPTY
  Attributes:
```

```
        Name:href, Type: CDATA

      Name: link
      Content Model: EMPTY
      Attributes:
        Name:subordinates, Type: IDREF(S)
        Name:manager, Type: IDREF(S)

      Name: family
      Content Model: (#PCDATA)*

      Name: given
      Content Model: (#PCDATA)*
```

Xerces C++ Sample 10: CreateDOMDocument

CreateDOMDocument

CreateDOMDocument, illustrates how you can create a DOM tree in memory from scratch. It then reports the elements in the tree that was just created.

Building on Windows

Load the xerces-c1_5_1-win32\samples\Projects\Win32\VC6\samples.dsw Microsoft Visual C++ workspace inside your MSVC IDE. Then build the project marked CreateDOMDocument.

Building on UNIX

```
cd xerces-c1_5_1-linux/samples
./runConfigure -p<platform> -c<C_compiler> -x<C++_compiler>
cd CreateDOMDocument
gmake
```

This will create the object files in the current directory and the executable named CreateDOMDocument in ' xerces-c1_5_1-linux/bin' directory.

To delete all the generated object files and executables, type

```
gmake clean
```

Running CreateDOMDocument

The CreateDOMDocument sample illustrates how you can create a DOM tree in memory from scratch. To run CreateDOMDocument, enter the following

```
CreateDOMDocument
```

Here is a sample output from CreateDOMDocument

```
cd xerces-c1_5_1-linux/samples/data
CreateDOMDocument
The tree just created contains: 4 elements.
```

Xerces C++ Sample 11: SAX2Count

SAX2Count

SAX2Count is the simplest application that counts the elements and characters of a given XML file using the (event based) SAX2 API.

Building on Windows

Load the xerces-c1_5_1-win32\samples\Projects\Win32\VC6\samples.dsw Microsoft Visual C++ workspace inside your MSVC IDE. Then build the project marked SAX2Count.

Building on UNIX

```
cd xerces-c1_5_1-linux/samples
./runConfigure -p<platform> -c<C_compiler> -x<C++_compiler>
cd SAX2Count
gmake
```

This will create the object files in the current directory and the executable named SAX2Count in 'xerces-c1_5_1-linux/bin' directory.

To delete all the generated object files and executables, type

```
gmake clean
```

Running SAX2Count

The SAX2Count sample parses an XML file and prints out a count of the number of elements in the file. To run SAX2Count, enter the following

```
SAX2Count <XML File>
```

The following parameters may be set from the command line

```
Usage:
    SAX2Count [options] <XML file>

Options:
    -v=xxx      Validation scheme [always | never | auto*]
    -n          Enable namespace processing. Defaults to off.
    -s          Disable schema processing. Defaults to on.

This program prints the number of elements, attributes,
white spaces and other non-white space characters in the input file.

  * = Default if not provided explicitly
```

-v=always will force validation

-v=never will not use any validation

-v=auto will validate if a DOCTYPE declaration is present in the XML document

Here is a sample output from SAX2Count

```
cd xerces-c1_5_1-linux/samples/data
SAX2Count -v=always personal.xml
personal.xml: 60 ms (37 elems, 12 attrs, 134 spaces, 134 chars)
```

Running SAX2Count with the validating parser gives a different result because ignorable white-space is counted separately from regular characters.

```
SAX2Count -v=never personal.xml
personal.xml: 10 ms (37 elems, 12 attrs, 0 spaces, 268 chars)
```

Note that the sum of spaces and chracters in both versions is the same.

> ***Note:*** *The time reported by the program may be different depending on your machine processor.*

Xerces C++ Sample 12: SAX2Print

SAX2Print

SAX2Print uses the SAX2 APIs to parse an XML file and print it back. Do note that the output of this sample is not exactly the same as the input (in terms of whitespaces, first line), but the output has the same information content as the input.

Building on Windows

Load the xerces-c1_5_1-win32\samples\Projects\Win32\VC6\samples.dsw Microsoft Visual C++ workspace inside your MSVC IDE. Then build the project marked SAX2Print.

Building on UNIX

```
cd xerces-c1_5_1-linux/samples
./runConfigure -p<platform> -c<C_compiler> -x<C++_compiler>
cd SAX2Print
gmake
```

This will create the object files in the current directory and the executable named SAX2Print in 'xerces-c1_5_1-linux/bin' directory.

To delete all the generated object files and executables, type

```
gmake clean
```

Running SAX2Print

The SAX2Print sample parses an XML file and prints out the contents again in XML (some loss occurs). To run SAX2Print, enter the following

```
SAX2Print <XML file>
```

The following parameters may be set from the command line

```
Usage: SAX2Print [options] file
This program prints the data returned by the various SAX2
handlers for the specified input file. Options are NOT case
sensitive.

Options:
    -u=xxx      Handle unrepresentable chars [fail | rep | ref*]
    -v=xxx      Validation scheme [always | never | auto*]
    -e          Expand Namespace Alias with URI's.
    -x=XXX      Use a particular encoding for output (LATIN1*).
    -?          Show this help

  * = Default if not provided explicitly

The parser has intrinsic support for the following encodings:
    UTF-8, USASCII, ISO8859-1, UTF-16[BL]E, UCS-4[BL]E,
    WINDOWS-1252, IBM1140, IBM037
```

-u=fail will fail when unrepresentable characters are encountered

-u=rep will replace with the substitution character for that codepage

-u=ref will report the character as a reference

-v=always will force validation

-v=never will not use any validation

-v=auto will validate if a DOCTYPE declaration is present in the XML document

Here is a sample output from SAX2Print

```
cd xerces-c1_5_1-linux/samples/data
SAX2Print -v=always personal.xml

<?xml version="1.0" encoding="LATIN1"?>
<personnel>

  <person id="Big.Boss">
    <name><family>Boss</family> <given>Big</given></name>
    <email>chief@foo.com</email>
    <link subordinates="one.worker two.worker three.worker
                        four.worker five.worker"></link>
  </person>

  <person id="one.worker">
    <name><family>Worker</family> <given>One</given></name>
    <email>one@foo.com</email>
    <link manager="Big.Boss"></link>
  </person>

  <person id="two.worker">
    <name><family>Worker</family> <given>Two</given></name>
    <email>two@foo.com</email>
    <link manager="Big.Boss"></link>
  </person>

  <person id="three.worker">
    <name><family>Worker</family> <given>Three</given></name>
    <email>three@foo.com</email>
    <link manager="Big.Boss"></link>
  </person>

  <person id="four.worker">
    <name><family>Worker</family> <given>Four</given></name>
    <email>four@foo.com</email>
    <link manager="Big.Boss"></link>
  </person>

  <person id="five.worker">
    <name><family>Worker</family> <given>Five</given></name>
    <email>five@foo.com</email>
    <link manager="Big.Boss"></link>
  </person>

</personnel>
```

Note: *SAX2Print does not reproduce the original XML file. SAX2Print and DOMPrint produce different results because of the way the two APIs store data and capture events.*

Xerces C++ Sample 13: IDOMCount

IDOMCount

IDOMCount uses the provided IDOM API to parse an XML file, constructs the DOM tree and walks through the tree counting the elements (using just one API call).

Building on Windows

Load the xerces-c1_5_1-win32\samples\Projects\Win32\VC6\samples.dsw Microsoft Visual C++ workspace inside your MSVC IDE. Then build the project marked IDOMCount.

Building on UNIX

```
cd xerces-c1_5_1-linux/samples
./runConfigure -p<platform> -c<C_compiler> -x<C++_compiler>
cd IDOMCount
gmake
```

This will create the object files in the current directory and the executable named IDOMCount in ' xerces-c1_5_1-linux/bin' directory.

To delete all the generated object files and executables, type

```
gmake clean
```

Running IDOMCount

The IDOMCount sample parses an XML file and prints out a count of the number of elements in the file. To run IDOMCount, enter the following

```
IDOMCount <XML file>
```

The following parameters may be set from the command line

```
Usage:
    IDOMCount [-v -n] {XML file}

This program invokes the XML4C IDOM parser, builds
the DOM tree, and then prints the number of elements
found in the input XML file.

Options:
    -v=xxx      Validation scheme [always | never | auto*]
    -n          Enable namespace processing. Defaults to off.

  * = Default if not provided explicitly
```

-v=always will force validation

-v=never will not use any validation

-v=auto will validate if a DOCTYPE declaration is present in the XML document

Here is a sample output from IDOMCount

```
cd xerces-c1_5_1-linux/samples/data
IDOMCount -v=always personal.xml
personal.xml: 20 ms (37 elems)
```

The output of both versions should be same.

> ***Note:*** *The time reported by the system may be different, depending on your processor type.*

Xerces C++ Sample 14: IDOMPrint

IDOMPrint

IDOMPrint parses an XML file, constructs the DOM tree, and walks through the tree printing each element. It thus dumps the XML back (output same as SAXPrint).

Building on Windows

Load the xerces-c1_5_1-win32\samples\Projects\Win32\VC6\samples.dsw Microsoft Visual C++ workspace inside your MSVC IDE. Then build the project marked IDOMPrint.

Building on UNIX

```
cd xerces-c1_5_1-linux/samples
./runConfigure -p<platform> -c<C_compiler> -x<C++_compiler>
cd IDOMPrint
gmake
```

This will create the object files in the current directory and the executable named IDOMPrint in 'xerces-c1_5_1-linux/bin' directory.

To delete all the generated object files and executables, type

```
gmake clean
```

Running IDOMPrint

The IDOMPrint sample parses an XML file, using either a validating or non-validating IDOM parser configuration, builds a DOM tree, and then walks the tree and outputs the contents of the nodes in a 'canonical' format. To run IDOMPrint, enter the following:

```
IDOMPrint <XML file>
```

The following parameters may be set from the command line

```
Usage: IDOMPrint [options] file

This program invokes the Xerces C++ IDOM parser and builds the DOM
tree. It then traverses the DOM tree and prints the contents
of the tree. Options are NOT case sensitive.

Options:
    -e          Expand entity references. Default is no expansion.
    -u=xxx      Handle unrepresentable chars [fail | rep | ref*]
    -v=xxx      Validation scheme [always | never | auto*]
    -n          Enable namespace processing. Default is off.
    -x=XXX      Use a particular encoding for output. Default is
                the same encoding as the input XML file. UTF-8 if
                input XML file has not XML declaration.
    -?          Show this help (must be the only parameter)

  * = Default if not provided explicitly

The parser has intrinsic support for the following encodings:
    UTF-8, USASCII, ISO8859-1, UTF-16[BL]E, UCS-4[BL]E,
```

```
        WINDOWS-1252, IBM1140, IBM037
```

-u=fail will fail when unrepresentable characters are encountered

-u=rep will replace with the substitution character for that codepage

-u=ref will report the character as a reference

-v=always will force validation

-v=never will not use any validation

-v=auto will validate if a DOCTYPE declaration is present in the XML document

Here is a sample output from IDOMPrint

```
    cd xerces-c1_5_1-linux/samples/data
    IDOMPrint -v personal.xml

    <?xml version="1.0" encoding="iso-8859-1"?>

    <!DOCTYPE personnel SYSTEM "personal.dtd">
    <!-- @version: -->
    <personnel>

    <person id="Big.Boss">
      <name><family>Boss</family> <given>Big</given></name>
      <email>chief@foo.com</email>
      <link subordinates="one.worker two.worker three.worker
                          four.worker five.worker"></link>
    </person>

    <person id="one.worker">
      <name><family>Worker</family> <given>One</given></name>
      <email>one@foo.com</email>
      <link manager="Big.Boss"></link>
    </person>

    <person id="two.worker">
      <name><family>Worker</family> <given>Two</given></name>
      <email>two@foo.com</email>
      <link manager="Big.Boss"></link>
    </person>

    <person id="three.worker">
      <name><family>Worker</family> <given>Three</given></name>
      <email>three@foo.com</email>
      <link manager="Big.Boss"></link>
    </person>

    <person id="four.worker">
      <name><family>Worker</family> <given>Four</given></name>
      <email>four@foo.com</email>
      <link manager="Big.Boss"></link>
    </person>

    <person id="five.worker">
```

```
      <name><family>Worker</family> <given>Five</given></name>
      <email>five@foo.com</email>
      <link manager="Big.Boss"></link>
    </person>

    </personnel>
```

Note that IDOMPrint does not reproduce the original XML file. IDOMPrint and SAXPrint produce different results because of the way the two APIs store data and capture events.

6
Schema

Disclaimer

Schema is not fully supported in Xerces C++ yet. But an experimental implementation of a subset of the W3C XML Schema language is now available for review in Xerces C++ 1.5.1. You should not consider this implementation complete or correct. The limitations of this implementation are detailed below. Please read this document before using Xerces C++ 1.5.1.

Introduction

The Xerces C++ 1.5.1 contains an implementation of a subset of the W3C XML Schema Language as specified in the 2 May 2001 Recommendation for Structures [24] and Datatypes [25] . The parsers contained in this package are able to read and validate XML documents with the grammar specified in either DTD or XML Schema format.

We intend to update this package until it implements all the functionality of the current XML Schema Recommendation. If you are interested in a particular unimplemented feature, or if you have any feedback on the implementation design, we welcome your input to the Xerces-C mailing list xerces-c-dev@xml.apache.org [15] .

Limitations

The XML Schema implementation in the Xerces C++ 1.5.1 is a subset of the features defined in the 2 May 2001 XML Schema Recommendation.

Features/Datatypes Supported

- Partial Simple type support
  - Yes: atomic simple type
  - No: union and list
- Partial Complex type suppport
  - Yes: choice, sequence
  - No: group, all
- Element and Attribute Declaration
  - No: any/anyAttribute
- SubsitutionGroup
- Subset of Built-in Datatypes
  - Primitive Datatypes
  - Derived Datatypes
- xsi Markup
  - Yes: xsi:nil

- Yes: xsi:schemaLocation and xsi:noNamespaceSchemaLocation
  - No: xsi:type

Additional Experimental Features (not tested and subject to change, use as is)

- Complex type derivation support (simpleContent and complexContent).
- Element and attribute re-use using "ref".
- Include support
- Import Support
- Element declaration <any>
- Subset of Built-in Datatypes
  - Derived Datatypes

Other features in the Schema recommendation such as "redefine", "identity constraints" and others which are not mentioned above, are not supported yet. Also, particle and model group constraint checking is not yet fully implemented. But development is continuing and we target to implement all the features of the current XML Schema Recommendation before end of this year. Please note that the date is tentative and subject to change.

Other Limitations

The schema must be specified by the xsi:schemaLocation or xsi:noNamespaceSchemaLocation attribute on the root element of the document. The xsi prefix must be bound to the Schema document instance namespace, as specified by the Recommendation. See the sample provided in the Usage section.

Usage

XML document specifies the XML Schema grammar location in the xsi:schemaLocation attribute attached to the root / top-level element. Here is an example with no target namspace:

```
<?xml version="1.0" encoding="UTF-8"?>
<personnel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:noNamespaceSchemaLocation='personal.xsd'>
...
</personnel>
```

Please see the sample file, 'samples/data/personal-schema.xml' for further detail. And review the sample file 'samples/data/personal.xsd' for an example of an XML Schema grammar.

Here is an example how to turn on schema processing in DOMParser (default is off). Note that you must also turn on namespace support (default is off) for schema processing.

```
// Instantiate the DOM parser.
DOMParser parser;
parser.setDoNamespaces(true);
parser.setDoSchema(true);
parser.parse(xmlFile);
```

Usage in SAXParser is similar, please refer to the sample program 'samples/SAXCount/SAXCount.cpp' for further reference.

Here is an example how to turn on schema processing in SAX2XMLReader (default is on). Note that namespace must be on (default is on) as well.

```
SAX2XMLReader* parser = XMLReaderFactory::createXMLReader();
parser->setFeature(XMLString::transcode("http://xml.org/sax/features/namespaces"),
```

```
    true);
     parser->setFeature(XMLString::transcode("http://apache.org/xml/features/validation/schem
    true);
     parser->parse(xmlFile);
```

7
Frequently Asked Questions

Distributing Xerces C++

What compilers are being used on the supported platforms?

Xerces binaries has been built on the following platforms with these compilers

Operating System	Compiler
Windows NT 4.0 SP5/98	MSVC 6.0 SP3
Redhat Linux 6.1	egcs-2.91.66 and glibc-2.1.2-11
AIX 4.2.1	xlC 3.6.4
Solaris 2.6	CC Workshop 4.2
HP-UX 11.0	aCC A.03.13 with pthreads

What are the differences between Xerces-C and XML4C?

Xerces-C has intrinsic support for ASCII, UTF-8, UTF-16 (Big/Small Endian), UCS4 (Big/Small Endian), EBCDIC code pages IBM037 and IBM1140 encodings, ISO-8859-1 (aka Latin1) and Windows-1252. This means that it can parse input XML files in these above mentioned encodings.

However, if you wish to parse XML files in any other encodings, say in Shift-JIS, Big5 etc., then you cannot use Xerces-C. XML4C addresses this need. It combines Xerces-C and International Components for Unicode (ICU) [11] and provides support for over 100 different encodings.

ICU is also an open source project but is licensed under the X License [26] . XML4C is published by IBM and can be downloaded from their Alphaworks [27] site. The license to use XML4C is simply to comply with the Apache license (because of Xerces-C) and X License (because of ICU).

XML4C binaries are published for Solaris using SunWorkshop compiler, HPUX 10.20 and 11.0 using CC and aCC, Redhat Linux using gcc, Windows NT using MSVC, AIX using xlC.

Which DLL's do I need to distribute with my application?

As mentioned above, there are two configurations in which Xerces-C binaries are shipped. One is from the Apache site [28] , while the other is from IBM published at IBM's Alphaworks Site [27] .

If you are using the binaries from the Apache download site [29] site, then you only need to distribute **one** file:

xerces-c_1_5_1.dll for Windows NT/95/98, or

libxerces-c1_5_1.a for AIX, or

libxerces-c1_5_1.so for Solaris/Linux, or

libxerces-c1_5_1.sl for HP-UX.

However, if you are using the XML4C binaries then in **addition** to the library file mentioned above, you also need to ship:

1. **ICU shared library file**:
   icuuc.dll for Windows NT/95/98, or
   libicuuc.a for AIX, or
   libicuuc.so for Solaris/Linux, or
   libicuuc.sl for HP-UX.
2. **ICU converter data shared library file:**
   icudata.dll for Windows NT/95/98, or
   libicudata.a for AIX, or
   libicudata.so for Solaris/Linux, or
   libicudata.sl for HP-UX.

How do I package the sources to create a binary drop?

You have to first compile the sources inside your IDE to create the required DLLs and EXEs. Then you need to copy over the binaries to another directory for the binary drop. A perl script has been provided to give you a jump start. You need to install perl on your machine for the script to work. If you have changed your source tree, you have to modify the script to suit your current directory structure. To invoke the script, go to the \<Xerces>\scripts directory, and type:

```
    perl packageBinaries.pl
```

You will get a message that somewhat looks like this (changes always happpen, we are evolving you see!):

```
    Usage is: packageBinaries <options>
    options are:  -s <source_directory>
                  -o <target_directory>
                  -c <C compiler name> (e.g. gcc or xlc_r)
                  -x <C++ compiler name> (e.g. g++ or xlC_r)
                  -m <message loader> can be 'inmem', 'icu' or 'iconv'
                  -n <net accessor> can be 'fileonly' or 'libwww'
                  -t <transcoder> can be 'icu' or 'native'
                  -r <thread option> can be 'pthread' or 'dce' (only used on HP-11)
                  -h to get help on these commands
    Example: perl packageBinaries.pl -s$HOME/xerces-c_1_0_0
                                     -o$HOME/xerces-c_1_0_0
                                     -cgcc -xg++ -minmem
                                     -nfileonly -tnative
```

Make sure that your compiler can be invoked from the command line and follow the instructions to produce a binary drop.

I do not see binaries for my platform. When will they be available?">

The reason why you see binaries only for some specific platforms is that we have had the maximum requests for them. Moreover, we have limited resources and hence cannot publish binaries for every platform. If you wish to contribute your time and effort in building binaries for a specific platform/environment then please send a mail to the Xerces-C mailing list [15] . We can definitely use any extra help in this open source project

When will a port to my platform be available?

We would like to see Xerces ported to as many platforms as there are. Again, due to limited resources we cannot do all the ports. We will help you make this port happen. Here are some Porting Guidelines.

We strongly encourage you to submit the changes that are required to make it work on another platform. We will incorporate these changes in the source code base and make them available in the future releases.

All porting changes may be sent to the Xerces-C mailing list [15] .

How can I port Xerces to my favourite platform?

Some porting information is mentioned on the build page.

What application did you used to create the documentation?

We have used an internal XML based application to create the documentation. The documentation files are all written in XML and the application, internally codenamed StyleBook, makes use of XSL to transform it into an HTML document that you are seeing right now. It is currently available on the Apache [30] open source website as Cocoon [31] .

The API documentation is automatically generated using doxygen [22] and GraphViz [23] .

Can I use Xerces in my product?

Yes! Read the license agreement first and if you still have further questions, then please address them to the Xerces-C mailing list [15] .

How do I uninstall Xerces C++?

Xerces C++ only installs itself in a single directory and does not set any registry entries. Thus, to uninstall, you only need to remove the directory where you installed it, and all Xerces C++ related files will be removed.

I am getting a tar checksum error on Solaris. What's the problem?

The problem is caused by a limitation in the original tar spec, which prevented it from archiving files with long pathnames. Unfortunately, various current versions of tar use different extensions for eliminating this restriction which are incompatible with each other (or they do not remove the restriction at all). Rather than altering the pathnames for the Xerces C++ package, which would make them compatible with the original tar spec but make it more difficult to know what was where, it was decided to use GNU tar (gtar), which handles arbitrarily long pathnames and is freely available on every platform on which Xerces C++ is supported. If you don't already have GNU tar installed on your system, you can obtain it from the Free Software Foundation http://www.gnu.org/software/tar/tar.html [32] . For additional background information on this problem, see the online manual GNU tar and POSIX tar [33] for the utility.

Parsing with Xerces C++

Does Xerces C++ support Schema?

See the Schema page.

Why Xerces C++ does not support this particular Schema feature?

See supported schema features in Xerces C++ 1.5.1

Why does my application crash on AIX when I run it under a multi-threaded environment?

AIX maintains two kinds of libraries on the system, thread-safe and non-thread safe. Multi-threaded libraries on AIX follow a different naming convention, Usually the multi-threaded library names are followed with "_r". For example, libc.a is single threaded whereas libc_r.a is multi-threaded.

To make your multi-threaded application run on AIX, you **must** ensure that you do not have a "system library path" in your `LIBPATH` environment variable when you run the application. The appropriate

libraries (threaded or non-threaded) are automatically picked up at runtime. An application usually crashes when you build your application for multi-threaded operation but don't point to the thread-safe version of the system libraries. For example, LIBPATH can be simply set as:

```
    LIBPATH=$HOME/<Xerces>/lib
```

Where <Xerces> points to the directory where the Xerces application resides.

If, for any reason unrelated to Xerces, you need to keep a "system library path" in your LIBPATH environment variable, you must make sure that you have placed the thread-safe path before you specify the normal system path. For example, you must place */lib/threads* before */lib* in your LIBPATH variable. That is to say your LIBPATH may look like this:

```
    export LIBPATH=$HOME/<Xerces>/lib:/usr/lib/threads:/usr/lib
```

Where /usr/lib is where your system libraries are.

I cannot run the sample applications. What is wrong?

In order to run an application built using Xerces you must set up your path and library search path properly. In the stand-alone version from Apache, you must have the Xerces C++ runtime library available from your path settings. On Windows this library is called `xerces-c_1_5_1.dll` which must be available from your `PATH` settings. (Note that now there are separate debug and release dlls for Windows. If the release dll is named `xerces-c_1_5_1.dll` then the debug dll is named `xerces-c_1_5_1d.dll`). On UNIX platforms the library is called `libxerces-c1_5_1.so` (or `.a` or `.sl`) which must be available from your `LD_LIBRARY_PATH` (or `LIBPATH` or `SHLIB_PATH`) environment variable.

Thus, if you installed your binaries under `$HOME/fastxmlparser`, you need to point your library path to that directory.

```
    export LIBPATH=$LIBPATH:$HOME/fastxmlparser/lib # (AIX)
    export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HOME/fastxmlparser/lib # (Solaris,
   Linux)
    export SHLIB_PATH=$SHLIB_PATH:$HOME/fastxmlparser/lib # (HP-UX)
```

If you are using the enhanced version of this parser from IBM, you will need to put in two additional DLLs. In the Windows build these are `icuuc.dll` and `icudata.dll` which must be available from your PATH settings. On UNIX, these libraries are called `libicuuc.so` and `libicudata.so` (or `.sl` for HP-UX or `.a` for AIX) which must be available from your library search path.

I just built my own application using the Xerces C++ parser. Why does it crash?

In order to work with the Xerces C++ parser, you have to first initialize the XML subsystem. The most common mistake is to forget this initialization. Before you make any calls to Xerces C++ APIs, you must call:

```
    XMLPlatformUtils::Initialize():
    try {
       XMLPlatformUtils::Initialize();
    }
    catch (const XMLException& toCatch) {
       // Do your failure processing here
    }
```

This initializes the Xerces system and sets its internal variables. Note that you must the include `util/PlatformUtils.hpp` file for this to work.

Is Xerces C++ thread-safe?

This is not a question that has a simple yes/no answer. Here are the rules for using Xerces C++ in a multi-threaded environment:

Within an address space, an instance of the parser may be used without restriction from a single thread, or an instance of the parser can be accessed from multiple threads, provided the application guarantees that only one thread has entered a method of the parser at any one time.

When two or more parser instances exist in a process, the instances can be used concurrently, without external synchronization. That is, in an application containing two parsers and two threads, one parser can be running within the first thread concurrently with the second parser running within the second thread.

The same rules apply to Xerces C++ DOM documents. Multiple document instances may be concurrently accessed from different threads, but any given document instance can only be accessed by one thread at a time.

DOMStrings allow multiple concurrent readers. All DOMString const methods are thread safe, and can be concurrently entered by multiple threads. Non-const DOMString methods, such as `appendData()`, are not thread safe and the application must guarantee that no other methods (including const methods) are executed concurrently with them.

The libs/dll's I downloaded keep me from using the debugger in VC6.0. I am using the 'D', debug versions of them. "no symbolic information found" is what it says. Do I have to compile everything from source to make it work?

Unless you have the .pdb files, all you are getting with the debug library is that it uses the debug heap manager, so that you can compile your stuff in debug mode and not be dangerous. If you want full symbolic info for the Xerces C++ library, you'll need the .pdb files, and to get those, you'll need to rebuild the Xerces C++ library.

"First-chance exception in DOMPrint.exe (KERNEL32.DLL): 0xE06D7363: Microsoft C++ Exception." I am always getting this message when I am using the parser. My programs are terminating abnormally. Even the samples are giving this exception. I am using Visual C++ 6.0 with latest service pack installed.

Xerces C++ uses C++ exceptions internally, as part of its normal operation. By default, the MSVC debugger will stop on each of these with the "First-chance exception ..." message.

To stop this from happening do this:

- start debugging (so the debug menu appears)
- from the debug menu select "Exceptions"
- from the box that opens select "Microsoft C++ Exception" and set it to "Stop if not handled" instead of "stop always".

You'll still land in the debugger if your program is terminating abnormally, but it will be at your problem, not from the internal Xerces C++ exceptions.

I am seeing memory leaks in Xerces C++. Are they real?

The Xerces C++ library allocates and caches some commonly reused items. The storage for these may be reported as memory leaks by some heap analysis tools; to avoid the problem, call the function `XMLPlatformUtils::Terminate()` before your application exits. This will free all memory that was being held by the library.

For most applications, the use of `Terminate()` is optional. The system will recover all memory when the application process shuts down. The exception to this is the use of Xerces C++ from DLLs that will be repeatedly loaded and unloaded from within the same process. To avoid memory leaks with this kind of use, `Terminate()` must be called before unloading the Xerces C++ library

Is there a facility in Xerces C++ to validate the data contained in a DOM tree? That is, without saving and re-parsing the source document?

No. This is a frequently requested feature, but at this time it is not possible to feed XML data from the DOM directly back to the DTD validator. The best option for now is to generate XML source from the DOM and feed that back into the parser.

Can I use Xerces to perform "write validation" (which is having an appropriate DTD and being able to add elements to the DOM whilst validating against the DTD)? Is there a function that I have totally missed that creates an XML file from a DTD, (obviously with the values missing, a skeleton, as it were.)

The answers are: "No" and "No." Write Validation is a commonly requested feature, but Xerces C++ does not have it yet.

The best you can do for now is to create the DOM document, write it back as XML and re-parse it.

Why does my multi-threaded application crash on Solaris?

The problem appears because the throw call on Solaris 2.6 is not multi-thread safe. Sun Microsystems provides a patch to solve this problem. To get the latest patch for solving this problem, go to SunSolve.sun.com [34] and get the appropriate patch for your operating system. For Intel machines running Solaris, you need to get Patch ID 104678. For SPARC machines you need to get Patch ID #105591.

Why does my application gives unresolved linking errors on Solaris?

On Solaris there are a few things that need to be done before you execute your application using Xerces C++. In case you're using the binary build of Xerces C++ make sure that the OS and compiler are the same version as the ones used to build the binary. Different OS and compiler versions might cause unresolved linking problems or compilation errors. If the versions are different, rebuild the Xerces C++ library on your system before building your application. If you're using ICU (which is packaged with XML4C) you need to rebuild the compatible version of ICU first.

Also check that the library path is set properly and that the correct versions of `gmake` and `autoconf` are on your system.

Why do I get Internal Compiler Error when compiling Xerces C++ for a 64bit target with gcc?

This is a compiler problem. Try turning off optimization to bypass the problem.

How are entity reference nodes handled in DOM?

If you are using the native DOM classes, the function `setExpandEntityReferences` controls how entities appear in the DOM tree. When setExpandEntityReferences is set to false (the default), an occurrence of an entity reference in the XML document will be represented by a subtree with an EntityReference node at the root whose children represent the entity expansion. Entity expansion will be a DOM tree representing the structure of the entity expansion, not a text node containing the entity expansion as text.

If setExpandEntityReferences is true, an entity reference in the XML document is represented by only the nodes that represent the entity expansion. The DOM tree will not contain any entityReference nodes.

What kinds of URLs are currently supported in Xerces C++?

The `XMLURL` class provides for limited URL support. It understands the `file://, http://`, and `ftp://` URL types, and is capable or parsing them into their constituent components, and normalizing them. It also supports the commonly required action of conglomerating a base and relative URL into a single URL. In other words, it performs the limited set of functions required by an XML parser.

Another thing that URLs commonly do are to create an input stream that provides access to the entity

referenced. The parser, as shipped, only supports this functionality on URLs in the form `file:///` and `file://localhost/`, i.e. only when the URL refers to a local file.

You may enable support for HTTP and FTP URLs by implementing and installing a NetAccessor object. When a NetAccessor object is installed, the URL class will use it to create input streams for the remote entities referred to by such URLs.

How can I add support for URLs with HTTP/FTP protocols?

Support for the http: protocol is now included by default on all platforms.

To address the need to make remote connections to resources specified using additional protocols, ftp for example, Xerces C++ provides the `NetAccessor` interface. The header file is `src/util/XMLNetAccessor.hpp`. This interface allows you to plug in your own implementation of URL networking code into the Xerces C++ parser.

Can I use Xerces C++ to parse HTML?

Yes, but only if the HTML follows the rules given in the XML specification [2] . Most HTML, however, does not follow the XML rules, and will generate XML well-formedness errors.

I keep getting an error: "invalid UTF-8 character". What's wrong?

Most commonly, the XML `encoding =` declaration is either incorrect or missing. Without a declaration, XML defaults to the use utf-8 character encoding, which is not compatible with the default text file encoding on most systems.

The XML declaration should look something like this:

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

Make sure to specify the encoding that is actually used by file. The encoding for "plain" text files depends both on the operating system and the locale (country and language) in use.

Another common source of problems is that some characters are not allowed in XML documents, according to the XML spec. Typical disallowed characters are control characters, even if you escape them using the Character Reference form. See the XML spec [35] , sections 2.2 and 4.1 for details. If the parser is generating an `Invalid character (Unicode: 0x???)` error, it is very likely that there's a character in there that you can't see. You can generally use a UNIX command like "od -hc" to find it.

What encodings are supported by Xerces-C / XML4C?

Xerces-C has intrinsic support for ASCII, UTF-8, UTF-16 (Big/Small Endian), UCS4 (Big/Small Endian), EBCDIC code pages IBM037 and IBM1140 encodings, ISO-8859-1 (aka Latin1) and Windows-1252. This means that it can parse input XML files in these above mentioned encodings.

XML4C -- the version of Xerces-C available from IBM -- extends this set to include the encodings listed in the table below.

Common Name	Use this name in XML
8 bit Unicode	UTF-8
ISO Latin 1	ISO-8859-1
ISO Latin 2	ISO-8859-2
ISO Latin 3	ISO-8859-3
ISO Latin 4	ISO-8859-4
ISO Latin Cyrillic	ISO-8859-5
ISO Latin Arabic	ISO-8859-6
ISO Latin Greek	ISO-8859-7

ISO Latin Hebrew	ISO-8859-8
ISO Latin 5	ISO-8859-9
EBCDIC US	ebcdic-cp-us
EBCDIC with Euro symbol	ibm1140
Chinese, PRC	gb2312
Chinese, Big5	Big5
Cyrillic	koi8-r
Japanese, Shift JIS	Shift_JIS
Korean, Extended UNIX code	euc-kr

Some implementations or ports of Xerces-C provide support for additional encodings. The exact set will depend on the supplier of the parser and on the character set transcoding services in use.

What character encoding should I use when creating XML documents?

The best choice in most cases is either utf-8 or utf-16. Advantages of these encodings include:

- The best portability. These encodings are more widely supported by XML processors than any others, meaning that your documents will have the best possible chance of being read correctly, no matter where they end up.
- Full international character support. Both utf-8 and utf-16 cover the full Unicode character set, which includes all of the characters from all major national, international and industry character sets.
- Efficient. utf-8 has the smaller storage requirements for documents that are primarily composed of of characters from the Latin alphabet. utf-16 is more efficient for encoding Asian languages. But both encodings cover all languages without loss.

The only drawback of utf-8 or utf-16 is that they are not the native text file format for most systems, meaning that common text file editors and viewers can not be directly used.

A second choice of encoding would be any of the others listed in the table above. This works best when the xml encoding is the same as the default system encoding on the machine where the XML document is being prepared, because the document will then display correctly as a plain text file. For UNIX systems in countries speaking Western European languages, the encoding will usually be iso-8859-1.

The versions of Xerces distributed by IBM, both C and Java (known respectively as XML4C and XML4J), include all of the encodings listed in the above table, on all platforms.

A word of caution for Windows users: The default character set on Windows systems is windows-1252, not iso-8859-1. While Xerces C++ does recognize this Windows encoding, it is a poor choice for portable XML data because it is not widely recognized by other XML processing tools. If you are using a Windows-based editing tool to generate XML, check which character set it generates, and make sure that the resulting XML specifies the correct name in the `encoding="..."` declaration.

I find memory leaks in Xerces C++. How do I eliminate it?

The "leaks" that are reported through a leak-detector or heap-analysis tools aren't really leaks in most application, in that the memory usage does not grow over time as the XML parser is used and re-used.

What you are seeing as leaks are actually lazily evaluated data allocated into static variables. This data gets released when the application ends. You can make a call to `XMLPlatformUtil::terminate()` to release all the lazily allocated variables before you exit your program.

Is EBCDIC supported?

Yes, Xerces C++ supports EBCDIC. When creating EBCDIC encoded XML data, the preferred encoding is ibm1140. Also supported is ibm037 (and its alternate name, ebcdic-cp-us); this encoding is almost the

same as ibm1140, but it lacks the Euro symbol.

These two encodings, ibm1140 and ibm037, are available on both Xerces-C and IBM XML4C, on all platforms.

On IBM System 390, XML4C also supports two alternative forms, ibm037-s390 and ibm1140-s390. These are similar to the base ibm037 and ibm1140 encodings, but with alternate mappings of the EBCDIC new-line character, which allows them to appear as normal text files on System 390s. These encodings are not supported on other platforms, and should not be used for portable data.

XML4C on System 390 and AS/400 also provides additional EBCDIC encodings, including those for the character sets of different countries. The exact set supported will be platform dependent, and these encodings are not recommended for portable XML data.

How to write out a DOM tree into an XML file?

This feature is not yet availabe in the parser. Take a look at the DOMPrint sample for an example on parsing XML file, then writing it out back to the screen. You can use that code.

Is it OK to call the XMLPlatformUtils::Initialize/Terminate pair of routines multiple times in one program?

No. XMLPlatformUtils::Initialize() can only be called once per process. Call Initialize() when you start and Terminate() when you end.

Why does deleting a transcoded string result in assertion on windows?

Both your application program and the Xerces DLL must use the same *DLL* version of the runtime library. If either statically links to the runtime library, the problem will still occur. For example, for a Win32/VC6 build, the runtime library build setting MUST be "Multithreaded DLL" for release builds and "Debug Multithreaded DLL" for debug builds.

How do I transcode to/from something besides the local code page?

XMLString::transcode() will transcode from XMLCh to the local code page, and other APIs which take a char* assume that the source text is in the local code page. If this is not true, you must transcode the text yourself. You can do this using local transcoding support on your OS, such as Iconv on Unix or or IBM's ICU package. However, if your transcoding needs are simple, you can achieve some better portability by using the Xerces parser's transcoder wrappers. You get a transcoder like this:

- 1. Call XMLPlatformUtils::fgTransServer- >MakeNewTranscoderFor() and provide the name of the encoding you wish to create a transcoder for. This will return a transcoder to you, which you own and must delete when you are through with it. NOTE: You must provide a maximum block size that you will pass to the transcoder at one time, and you must blocks of characters of this count or smaller when you do your transcoding. The reason for this is that this is really an internal API and is used by the parser itself to do transcoding. The parser always does transcoding in known block sizes, and this allows transcoders to be much more efficient for internal use since it knows the max size it will ever have to deal with and can set itself up for that internally. In general, you should stick to block sizes in the 4 to 64K range.
- 2. The returned transcoder is something derived from XMLTranscoder, so they are all returned to you via that interface.
- 3. This object is really just a wrapper around the underlying transcoding system actually in use by your version of Xerces, and does whatever is necessary to handle differences between the XMLCh representation and the representation uesd by that underying transocding system.
- 4. The transcoder object has two primary APIs, transcodeFrom() and transcodeTo(). These transcode between the XMLCh format and the encoding you indicated.
- 5. These APIs will transcode as much of the source data as will fit into the outgoing buffer you

provide. They will tell you how much of the source they ate and how much of the target they filled. You can use this information to continue the process until all source is consumed.

- 6. char* data is always dealt with in terms of bytes, and XMLCh data is always dealt with in terms of characters. Don't mix up which you are dealing with or you will not get the correct results, since many encodings don't have a one to one relationship of characters to bytes.
- 7. When transcoding from XMLCh to the target encoding, the transcodeTo() method provides an 'unrepresentable flag' parameter, which tells the transcoder how to deal with an XMLCh code point that cannot be converted legally to the target encoding, which can easily happen since XMLCh is Unicode and can represent thousands of code points. The options are to use a default replacement character (which the underlying transcoding service will choose, and which is guaranteed to be legal for the target encoding), or to throw an exception.

Why DOM_Node::cloneNode() does not clone the pointer assigned to a DOM_Node via DOM_Node::setUserData()?

There are several possible options for how cloneNode should handle userData:

- 1) Copy the pointer. May be a Very Bad Idea if you really wanted the data associated with a particular node object.
- 2) Clone the object being pointed at. Maybe a Very Bad Idea if that object, in turn, wasn't designed to be cloned at this time.
- 3) A complex call-back API has been proposed which would allow the userData object to tell the DOM which of these three options should be taken, but that would require that only objects implementing that API be registered as userData. That doesn't seem to be a good option.
- 4) Do nothing. This is by far the lowest-overhead and safest choice. And since cloneNode is a DOM operation, and userData is _not_ defined by the standard DOM API, one can make a very strong case for this being the "most correct" option.

We chose (4), very deliberately. If you want one of the others, you can implement it by creating your own wrapper operation for cloneNode() and calling that.

NOTE that userData should be considered a nonportable, experimental feature of the Xerces DOM. It may evaporate entirely in favor of a scheme based on the DOM Level 3 "node key" mechanism, when that becomes officially available.

Other Xerces C++ Questions

How do I determine the version of Xerces C++ I am using?

The version string for Xerces C++ is in one of the header files. Look inside the file `src/util/XercesDefs.hpp` or, in the binary distribution, look in `include/utils/XercesDefs.hpp`. Search for the static variable `gXercesFullVersionStr` and look at its definition. (It is usually a string like "1_4_0" or something similar). This is the version of Xerces C++ you are using.

If you don't have the header files, you have to find the version information from the shared library name. On Windows NT/95/98 right click on the DLL name xerces-c_1_5_1.dll in the bin directory and look up properties. The version information may be found on the Version tab.

On AIX, just look for the library name libxerces-c1_5_1.a (or libxerces-c1_5_1.so on Solaris/Linux and libxerces-c1_5_1.sl on HP-UX). The version number is coded in the name of the library.

I can't use C++. Do you have a Java version?

Yes. The Xerces family of products also has a Java version. More information is available at: http://xml.apache.org/xerces-j/index.html [36]

Where can I find additional information on XML?

The Web. http://www.oasis-open.org/cover/xml.html [37] is an excellent place to start, with links to overviews, FAQs, specifications, industry news, applications and other software, related standards, etc.

Is there any kind of support available for Xerces C++?

Xerces C++ comes with **no** formal support.

Every volunteer project obtains its strength from the people involved in it. Mailing lists provide a simple and effective communication mechanism. You are welcome to join any of these mailing lists (or all of them if you wish). You can choose to lurk, or to actively participate. It's up to you. Before you join these lists, you should look over the resources in the Reference Library section

Instructions for subscribing are at http://xml.apache.org/mail.html. Archives of the lists are available from http://archive.covalent.net

I found a defect -- how do I report it?

See Bug Reporting.

I have a patch to the Xerces C++ source code. How do I submit it?

Mail it to the Xerces C++ mailing list [15] at Apache. (You must be a subscriber to post to this list. But if you're considering changing the code you really want to be a subscriber, in any case.) There are no set rules about how or what must be included -- if you've fixed a problem or enhanced the code in some way, we really would like to get your changes, and will take them in any reasonable form.

Generally a diff of the changed files against the current sources from CVS is good, along with some kind of description of what the change is. (Working with the current sources is important!)

Where can I get predefined character entity definitions??

Download http://www.w3.org/TR/xhtml1/xhtml1.zip. [38]

8
Programming Guide

This page has sections on the following topics:

- SAX Programming Guide
  - Constructing a parser
  - Using the SAX API
- SAX2 Programming Guide
  - Constructing an XML Reader
  - Using the SAX2 API
  - Supported Features
- DOM Programming Guide
  - Comparision of Java and C++ DOM's
    - Accessing the API from application code
    - Class Names
    - Objects and Memory Management
  - DOMString
    - Equality Testing
  - Downcasting
  - Subclassing
- Experimental IDOM Programming Guide
  - Constructing a parser
  - Comparision of C++ DOM and IDOM
    - Motivation behind new design
    - Class Names
    - Objects and Memory Management
    - DOMString vs. XMLCh

SAX1 Programming Guide

Constructing a parser

In order to use Xerces C++ to parse XML files, you will need to create an instance of the SAXParser class. The example below shows the code you need in order to create an instance of SAXParser. The DocumentHandler and ErrorHandler instances required by the SAX API are provided using the HandlerBase class supplied with Xerces C++.

```
    int main (int argc, char* args[]) {

        try {
            XMLPlatformUtils::Initialize();
```

```
        }
        catch (const XMLException& toCatch) {
            cout << "Error during initialization! :\n"
                 << toCatch.getMessage() << "\n";
            return 1;
        }

        char* xmlFile = "x1.xml";
        SAXParser* parser = new SAXParser();
        parser->setDoValidation(true);    // optional.
            parser->setDoNamespaces(true);    // optional

        DocumentHandler* docHandler = new HandlerBase();
        ErrorHandler* errHandler = (ErrorHandler*) docHandler;
        parser->setDocumentHandler(docHandler);
        parser->setErrorHandler(errHandler);

        try {
            parser->parse(xmlFile);
        }
        catch (const XMLException& toCatch) {
            cout << "\nFile not found: '" << xmlFile << "'\n"
                 << "Exception message is: \n"
                 << toCatch.getMessage() << "\n" ;
            return -1;
        }
    }
```

Using the SAX API

The SAX API for XML parsers was originally developed for Java. Please be aware that there is no standard SAX API for C++, and that use of the Xerces C++ SAX API does not guarantee client code compatibility with other C++ XML parsers.

The SAX API presents a callback based API to the parser. An application that uses SAX provides an instance of a handler class to the parser. When the parser detects XML constructs, it calls the methods of the handler class, passing them information about the construct that was detected. The most commonly used handler classes are DocumentHandler which is called when XML constructs are recognized, and ErrorHandler which is called when an error occurs. The header files for the various SAX handler classes are in '<xerces-c1_5_1 >/include/sax'

As a convenience, Xerces C++ provides the class HandlerBase, which is a single class which is publicly derived from all the Handler classes. HandlerBase's default implementation of the handler callback methods is to do nothing. A convenient way to get started with Xerces C++ is to derive your own handler class from HandlerBase and override just those methods in HandlerBase which you are interested in customizing. This simple example shows how to create a handler which will print element names, and print fatal error messages. The source code for the sample applications show additional examples of how to write handler classes.

This is the header file MySAXHandler.hpp:

```
    #include <sax/HandlerBase.hpp>

    class MySAXHandler : public HandlerBase {
```

```
public:
    void startElement(const XMLCh* const, AttributeList&);
    void fatalError(const SAXParseException&);
};
```

This is the implementation file MySAXHandler.cpp:

```
#include "MySAXHandler.hpp"
#include <iostream.h>

MySAXHandler::MySAXHandler()
{
}

MySAXHandler::startElement(const XMLCh* const name,
                           AttributeList& attributes)
{
    // transcode() is an user application defined function which
    // converts unicode strings to usual 'char *'. Look at
    // the sample program SAXCount for an example implementation.
    cout << "I saw element: " << transcode(name) << endl;
}

MySAXHandler::fatalError(const SAXParseException& exception)
{
    cout << "Fatal Error: " << transcode(exception.getMessage())
         << " at line: " << exception.getLineNumber()
         << endl;
}
```

The XMLCh and AttributeList types are supplied by Xerces C++ and are documented in the include files. Examples of their usage appear in the source code to the sample applications.

SAX2 Programming Guide

Constructing an XML Reader

In order to use Xerces C++ to parse XML files, you will need to create an instance of the SAX2XMLReader class. The example below shows the code you need in order to create an instance of SAX2XMLReader. The ContentHandler and ErrorHandler instances required by the SAX API are provided using the DefaultHandler class supplied with Xerces C++.

```
int main (int argc, char* args[]) {

    try {
        XMLPlatformUtils::Initialize();
    }
    catch (const XMLException& toCatch) {
        cout << "Error during initialization! :\n"
             << toCatch.getMessage() << "\n";
        return 1;
    }

    char* xmlFile = "x1.xml";
```

```
        SAX2XMLReader* parser = XMLReaderFactory::createXMLReader();
        parser->setFeature(XMLString::transcode("http://xml.org/sax/features/validation",
    true)   // optional
        parser->setFeature(XMLString::transcode("http://xml.org/sax/features/namespaces",
    true)   // optional

        ContentHandler* contentHandler = new DefaultHandler();
        ErrorHandler* errHandler = (ErrorHandler*) contentHandler;
        parser->setContentHandler(contentHandler);
        parser->setErrorHandler(errHandler);

        try {
            parser->parse(xmlFile);
        }
        catch (const XMLException& toCatch) {
            cout << "\nFile not found: '" << xmlFile << "'\n"
                 << "Exception message is: \n"
                 << toCatch.getMessage() << "\n" ;
            return -1;
        }
    }
```

Using the SAX2 API

The SAX2 API for XML parsers was originally developed for Java. Please be aware that there is no standard SAX2 API for C++, and that use of the Xerces C++ SAX2 API does not guarantee client code compatibility with other C++ XML parsers.

The SAX2 API presents a callback based API to the parser. An application that uses SAX2 provides an instance of a handler class to the parser. When the parser detects XML constructs, it calls the methods of the handler class, passing them information about the construct that was detected. The most commonly used handler classes are ContentHandler which is called when XML constructs are recognized, and ErrorHandler which is called when an error occurs. The header files for the various SAX2 handler classes are in '<xerces-c1_5_1 >/include/sax2'

As a convenience, Xerces C++ provides the class DefaultHandler, which is a single class which is publicly derived from all the Handler classes. DefaultHandler's default implementation of the handler callback methods is to do nothing. A convenient way to get started with Xerces C++ is to derive your own handler class from DefaultHandler and override just those methods in HandlerBase which you are interested in customizing. This simple example shows how to create a handler which will print element names, and print fatal error messages. The source code for the sample applications show additional examples of how to write handler classes.

This is the header file MySAX2Handler.hpp:

```
    #include <sax2/DefaultHandler.hpp>

    class MySAX2Handler : public DefaultHandler {
    public:
        void startElement(
            const   XMLCh* const    uri,
            const   XMLCh* const    localname,
            const   XMLCh* const    qname,
            const   Attributes&     attrs
```

```
        );
        void fatalError(const SAXParseException&);
    };
```

This is the implementation file MySAX2Handler.cpp:

```
    #include "MySAX2Handler.hpp"
    #include <iostream.h>

    MySAX2Handler::MySAX2Handler()
    {
    }

    MySAX2Handler::startElement(const   XMLCh* const    uri,
                                const   XMLCh* const    localname,
                                const   XMLCh* const    qname,
                                const   Attributes&     attrs)
    {
        // transcode() is an user application defined function which
        // converts unicode strings to usual 'char *'. Look at
        // the sample program SAX2Count for an example implementation.
        cout << "I saw element: " << transcode(qname) << endl;
    }

    MySAX2Handler::fatalError(const SAXParseException& exception)
    {
        cout << "Fatal Error: " << transcode(exception.getMessage())
             << " at line: " << exception.getLineNumber()
             << endl;
    }
```

The XMLCh and Attributes types are supplied by Xerces C++ and are documented in the include files. Examples of their usage appear in the source code to the sample applications.

Xerces SAX2 Supported Features

The behavior of the SAX2XMLReader is dependant on the values of the following features. All of the features below can be set using the `SAX2XMLReader::setFeature(XMLCh*,bool)` function. None of these features can be modified in the middle of a parse, or an exception will be thrown.

http://xml.org/sax/features/namespaces	
true:	Perform Namespace processing (default)
false:	Optionally do not perform Namespace processing

http://xml.org/sax/features/namespace-prefixes	
true:	Report the orignal prefixed names and attributes used for Namespace declarations (default)
false:	Do not report attributes used for Namespace declarations, and optionally do not report original prefixed names.

http://xml.org/sax/features/validation	
true:	Report all validation errors. (default)
false:	Do not report validation errors.

http://apache.org/xml/features/validation/dynamic	
true:	The parser will validate the document only if a grammar is specified. (http://xml.org/sax/features/validation must be true)
false:	Validation is determined by the state of the http://xml.org/sax/features/validation feature (default)

http://apache.org/xml/features/validation/schema	
true:	Enable the parser's schema support. (default)
false:	Disable the parser's schema support.

http://apache.org/xml/features/validation/reuse-grammar	
true:	The parser will reuse grammar information from previous parses in subsequent parses.
false:	The parser will not reuse any grammar information. (default)

http://apache.org/xml/features/validation/reuse-validator (deprecated)	
true:	The parser will reuse grammar information from previous parses in subsequent parses.
false:	The parser will not reuse any grammar information. (default)

DOM Programming Guide

Java and C++ DOM comparisons

The C++ DOM API is very similar in design and use, to the Java DOM API bindings. As a consequence, conversion of existing Java code that makes use of the DOM to C++ is a straight forward process.

This section outlines the differences between Java and C++ bindings.

Accessing the API from application code

```
// C++
#include <dom/DOM.hpp>
```

```
// Java
import org.w3c.dom.*
```

The header file <dom/DOM.hpp> includes all the individual headers for the DOM API classes.

Class Names

The C++ class names are prefixed with "DOM_". The intent is to prevent conflicts between DOM class names and other names that may already be in use by an application or other libraries that a DOM based application must link with.

The use of C++ namespaces would also have solved this conflict problem, but for the fact that many compilers do not yet support them.

```
DOM_Document   myDocument;   // C++
DOM_Node       aNode;
DOM_Text       someText;
```

```
Document       myDocument;   // Java
Node           aNode;
Text           someText;
```

If you wish to use the Java class names in C++, then you need to typedef them in C++. This is not advisable for the general case - conflicts really do occur - but can be very useful when converting a body of existing Java code to C++.

```
typedef DOM_Document  Document;
typedef DOM_Node      Node;

Document   myDocument;        // Now C++ usage is
                              // indistinguishable from Java
Node       aNode;
```

Objects and Memory Management

The C++ DOM implementation uses automatic memory management, implemented using reference counting. As a result, the C++ code for most DOM operations is very similar to the equivalent Java code, right down to the use of factory methods in the DOM document class for nearly all object creation, and the lack of any explicit object deletion.

Consider the following code snippets

```
// This is C++
DOM_Node       aNode;
aNode = someDocument.createElement("ElementName");
DOM_Node docRootNode = someDoc.getDocumentElement();
docRootNode.AppendChild(aNode);
```

```
// This is Java
Node       aNode;
aNode = someDocument.createElement("ElementName");
Node docRootNode = someDoc.getDocumentElement();
docRootNode.AppendChild(aNode);
```

The Java and the C++ are identical on the surface, except for the class names, and this similarity remains true for most DOM code.

However, Java and C++ handle objects in somewhat different ways, making it important to understand a little bit of what is going on beneath the surface.

In Java, the variable `aNode` is an object reference , essentially a pointer. It is initially == null, and references an object only after the assignment statement in the second line of the code.

In C++ the variable `aNode` is, from the C++ language's perspective, an actual live object. It is

constructed when the first line of the code executes, and DOM_Node::operator = () executes at the second line. The C++ class DOM_Node essentially a form of a smart-pointer; it implements much of the behavior of a Java Object Reference variable, and delegates the DOM behaviors to an implementation class that lives behind the scenes.

Key points to remember when using the C++ DOM classes:

- Create them as local variables, or as member variables of some other class. Never "new" a DOM object into the heap or make an ordinary C pointer variable to one, as this will greatly confuse the automatic memory management.
- The "real" DOM objects - nodes, attributes, CData sections, whatever, do live on the heap, are created with the create... methods on class DOM_Document. DOM_Node and the other DOM classes serve as reference variables to the underlying heap objects.
- The visible DOM classes may be freely copied (assigned), passed as parameters to functions, or returned by value from functions.
- Memory management of the underlying DOM heap objects is automatic, implemented by means of reference counting. So long as some part of a document can be reached, directly or indirectly, via reference variables that are still alive in the application program, the corresponding document data will stay alive in the heap. When all possible paths of access have been closed off (all of the application's DOM objects have gone out of scope) the heap data itself will be automatically deleted.
- There are restrictions on the ability to subclass the DOM classes.

DOMString

Class DOMString provides the mechanism for passing string data to and from the DOM API. DOMString is not intended to be a completely general string class, but rather to meet the specific needs of the DOM API.

The design derives from two primary sources: from the DOM's CharacterData interface and from class `java.lang.string`.

Main features are:

- It stores Unicode text.
- Automatic memory management, using reference counting.
- DOMStrings are mutable - characters can be inserted, deleted or appended.

When a string is passed into a method of the DOM, when setting the value of a Node, for example, the string is cloned so that any subsequent alteration or reuse of the string by the application will not alter the document contents. Similarly, when strings from the document are returned to an application via the DOM API, the string is cloned so that the document can not be inadvertently altered by subsequent edits to the string.

> ***Note:*** *The ICU classes are a more general solution to UNICODE character handling for C++ applications. ICU is an Open Source Unicode library, available at the IBM DeveloperWorks website [11] .*

Equality Testing

The DOMString equality operators (and all of the rest of the DOM class conventions) are modeled after the Java equivalents. The equals() method compares the content of the string, while the == operator checks whether the string reference variables (the application program variables) refer to the same underlying string in memory. This is also true of DOM_Node, DOM_Element, etc., in that operator == tells whether the variables in the application are referring to the same actual node or not. It's all very Java-like

- bool operator == () is true if the DOMString variables refer to the same underlying storage.

· bool equals() is true if the strings contain the same characters.

Here is an example of how the equality operators work:

```
DOMString a = "Hello";
DOMString b = a;
DOMString c = a.clone();
if (b == a)           //  This is true
if (a == c)           //  This is false
if (a.equals(c))       //  This is true
b = b + " World";
if (b == a)           // Still true, and the string's
                      //    value is "Hello World"
if (a.equals(c))      // false.  a is "Hello World";
                      //    c is still "Hello".
```

Downcasting

Application code sometimes must cast an object reference from DOM_Node to one of the classes deriving from DOM_Node, DOM_Element, for example. The syntax for doing this in C++ is different from that in Java.

```
// This is C++
DOM_Node       aNode = someFunctionReturningNode();
DOM_Element    el = (Element &) aNode;
```

```
// This is Java
Node       aNode = someFunctionReturningNode();
Element    el = (Element) aNode;
```

The C++ cast is not type-safe; the Java cast is checked for compatible types at runtime. If necessary, a type-check can be made in C++ using the node type information:

```
// This is C++

DOM_Node       aNode = someFunctionReturningNode();
DOM_Element    el;    // by default, el will == null.

if (anode.getNodeType() == DOM_Node::ELEMENT_NODE)
   el = (Element &) aNode;
else
   // aNode does not refer to an element.
   // Do something to recover here.
```

Subclassing

The C++ DOM classes, DOM_Node, DOM_Attr, DOM_Document, etc., are not designed to be subclassed by an application program.

As an alternative, the DOM_Node class provides a User Data field for use by applications as a hook for extending nodes by referencing additional data or objects. See the API description for DOM_Node for details.

Experimental IDOM Programming Guide

The experimental IDOM API is a new design of the C++ DOM API. Please note that this experimental IDOM API is only a prototype and is subject to change.

Constructing a parser

In order to use Xerces C++ to parse XML files using IDOM, you will need to create an instance of the IDOMParser class. The example below shows the code you need in order to create an instance of the IDOMParser.

```
int main (int argc, char* args[]) {

    try {
        XMLPlatformUtils::Initialize();
    }
    catch (const XMLException& toCatch) {
        cout << "Error during initialization! :\n"
             << toCatch.getMessage() << "\n";
        return 1;
    }

    char* xmlFile = "x1.xml";
    IDOMParser* parser = new IDOMParser();
    parser->setValidationScheme(IDOMParser::Val_Always);    // optional.
    parser->setDoNamespaces(true);    // optional

    ErrorHandler* errHandler = (ErrorHandler*) new HandlerBase();
    parser->setErrorHandler(errHandler);

    try {
        parser->parse(xmlFile);
    }
    catch (const XMLException& toCatch) {
        cout << "\nFile not found: '" << xmlFile << "'\n"
             << "Exception message is: \n"
             << toCatch.getMessage() << "\n" ;
       return -1;
    }

    return 0;
}
```

Comparision of C++ DOM and IDOM

This section outlines the differences between the C++ DOM and IDOM APIs.

Motivation behind new design

The performance of the C++ DOM has not been as good as it might be, especially for use in server style applications. The DOM's reference counted automatic memory management has been the biggest time consumer. The situation becomes worse when running multi-threaded applications.

The experimental C++ IDOM is a new alternative to the C++ DOM, and aims at meeting the following requirements:

- Reduced memory footprint.
- Fast.

- Good scalability on multiprocessor systems.
- More C++ like and less Java like.

Class Names

The IDOM class names are prefixed with "IDOM_". The intent is to prevent conflicts between IDOM class names and DOM class names that may already be in use by an application or other libraries that a DOM based application must link with.

```
IDOM_Document*   myDocument;   // IDOM
IDOM_Node*       aNode;
IDOM_Text*       someText;
```

```
DOM_Document     myDocument;   // DOM
DOM_Node         aNode;
DOM_Text         someText;
```

Objects and Memory Management

The C++ IDOM implementation no longer uses reference counting for automatic memory management. The storage for a DOM document is associated with the document node object. Applications would use normal C++ pointers to directly access the implementation objects for Nodes in IDOM C++, while they would use object references in DOM C++.

Consider the following code snippets

```
// IDOM C++
IDOM_Node*       aNode;
IDOM_Node* docRootNode;
aNode = someDocument->createElement("ElementName");
docRootNode = someDocument->getDocumentElement();
docRootNode->appendChild(aNode);
```

```
// DOM C++
DOM_Node       aNode;
DOM_Node docRootNode;
aNode = someDocument.createElement("ElementName");
docRootNode = someDocument.getDocumentElement();
docRootNode.appendChild(aNode);
```

The IDOM C++ uses an independent storage allocator per document. The advantage here is that allocation would require no synchronization in most cases (based on the the same threading model that we have now - one thread active per document, but any number of documents running in parallel with separate threads).

The allocator does not support a delete operation at all - all allocated memory would persist for the life of the document, and then the larger blocks would be returned to the system without separately deleting all of the individual nodes and strings within the document.

The C++ DOM and IDOM are similar in the use of factory methods in the document class for all object creation. They differ in the object deletion mechanism.

In C++ DOM, there is no explicit object deletion. The deallocation of memory is automatically taken care of by the reference counting.

In C++ IDOM, there is an implict and explict object deletion. When parsing a document using an IDOMParser, the storage allocated will be automatically deleted when the parser instance is deleted (implicit). If a user is manually building a DOM tree in memory using the document factory methods, then the user needs to explicilty delete the document object to free all allocated memory.

Consider the following code snippets:

```
// C++ IDOM - explicit deletion
IDOM_Document*   myDocument;
IDOM_Node*       aNode;
myDocument = IDOM_DOMImplementation::getImplementation()->createDocument();
aNode = myDocument->createElement("ElementName");
myDocument->appendChild(aNode);
delete myDocument;
```

```
// C++ DOM - implicit deletion
IDOM_Document   myDocument;
DOM_Node        aNode;
myDocument = DOM_DOMImplementation::getImplementation().createDocument();
aNode = myDocument.createElement("ElementName");
myDocument.appendChild(aNode);
```

Key points to remember when using the C++ IDOM classes:

- The DOM objects are accessed via C++ pointers.
- The DOM objects - nodes, attributes, CData sections, etc., are created with the factory methods (create...) in the document class.
- If you are manually building a DOM tree in memory, you need to explicitly delete the document object. Memory management will be automatically taken care of by the IDOM parser when parsing an instance document.

DOMString vs. XMLCh

The IDOM C++ no longer uses DOMString to pass string data to and from the DOM API. Instead, the IDOM C++ uses plain, null-terminated (XMLCh *) utf-16 strings. The (XMLCh*) utf-16 type string is much simpler with lower overhead. All the string data would remain in memory until the document object is deleted.

```
//C++ IDOM
const XMLCh* nodeValue = aNode->getNodeValue();
```

```
//C++ DOM
DOMString    nodeValue = aNode.getNodeValue();
```

9
Migration

Migrating from Xerces C++ 1.4.0 to Xerces C++ 1.5.1

This document is a discussion of the technical differences between Xerces C++ 1.4.0 code base and the new Xerces C++ 1.5.1 code base.

Topics discussed are:

- General Improvements
  - Compliance
  - Bug Fixes
  - Speed
- Changes required to migrate to Xerces C++ 1.5.1
  - Validator directory Reorganization
  - DTDValidator
- New features in Xerces C++ 1.5.1
  - Schema Subset Support
  - Experiemental IDOM

General Improvements

The new version is improved in many ways. Some general improvements are: significantly better conformance to the XML spec, cleaner internal architecture, many bug fixes, and faster speed.

Compliance

Except for a couple of the very obscure (mostly related to the 'standalone' mode), this version should be quite compliant to XML 1.0 [2] . It also tracks the latest changes to DOM, SAX and Namespace Specification. We have more than a thousand tests, some collected from various public sources and some IBM generated, which are used to do regression testing. The C++ parser is now passing all but a handful of them.

Bug Fixes

This version has many bug fixes since last release. Some of these were reported by users and some were brought up by way of the conformance testing.

Speed

Much work was done to speed up this version. Some of the new features, such as experiemental IDOM ended up eating up some of these gains, but overall the new version is significantly faster than previous versions, even while doing more.

Changes required to migrate to Xerces C++ 1.5.1

There are some architectural changes between the Xerces C++ 1.4.0 and the Xerces C++ 1.5.1 releases of the parser, and as a result, some code has undergone restructuring as shown below.

Validator directory Reorganization

- common content model files such as DFAContentModel ... are moved to a new directory called src/validators/common
- DTD related files are moved to a new directory called src/validators/DTD
- new directory src/validators/Datatype is created to store all datatype validators
- new directory src/validators/schema is created to store Schema related files

DTDValidator

DTDValidator was design to scan, validate and store the DTD in Xerces C++ 1.4.0 or earlier. In Xerces C++ 1.5.1, this process is broken down into three components:

- new class DTDScanner - to scan the DTD
- new class DTDGrammar - to store the DTD Grammar
- DTDValidator - to validate the DTD only

New features in Xerces C++ 1.5.1

Schema subset support is provided in this release. See supported schema features in Xerces C++ 1.5.1.. An experiemental IDOM is also available as well.

Schema Subset Support

- Schema Subset support is added
  - New function "setDoSchema" is added to DOM/SAX parser.
  - New feature "http://apache.org/xml/features/validation/schema" is recognized by SAX2XMLReader.
  - New classes such as SchemaValidator, TraverseSchema ... are added.
  - The Scanner is enhanced to process schema.
- New sample data files personal-schema.xml and personal.xsd.
- New command line option "-s" for samples.

See Schema Usage

Experiemental IDOM

The experimental IDOM API is a new design of the C++ DOM API. If you would like to migrate from DOM to the experimental IDOM, please refer to IDOM programming guide. Please note that this experimental IDOM API is only a prototype and is subject to change.

Migration Archive

For migration information from XML4C 2.x to Xerces C++ 1.4.0, please refer to Migration Archive.

10
Migration Archive

Migrating from XML4C 2.x to Xerces C++ 1.4.0

This document is a discussion of the technical differences between XML4C 2.x code base and the new Xerces C++ 1.4.0 code base.

Topics discussed are:

- General Improvements
  - Compliance
  - Bug Fixes
  - Speed
- Summary of changes required to migrate from XML4C 2.x to Xerces C++ 1.4.0
- The Samples
- Parser Classes
- DOM Level 2 support
- Progressive Parsing
- Namespace support
- Moved Classes to src/framework
- Loadable Message Text
- Pluggable Validators
- Pluggable Transcoders
- Util directory Reorganization
  - util - The platform independent utility stuff

General Improvements

The new version is improved in many ways. Some general improvements are: significantly better conformance to the XML spec, cleaner internal architecture, many bug fixes, and faster speed.

Compliance

Except for a couple of the very obscure (mostly related to the 'standalone' mode), this version should be quite compliant. We have more than a thousand tests, some collected from various public sources and some IBM generated, which are used to do regression testing. The C++ parser is now passing all but a handful of them.

Bug Fixes

This version has many bug fixes with regard to XML4C version 2.x. Some of these were reported by users and some were brought up by way of the conformance testing.

Speed

Much work was done to speed up this version. Some of the new features, such as namespaces, and conformance checks ended up eating up some of these gains, but overall the new version is significantly faster than previous versions, even while doing more.

Summary of changes required to migrate from XML4C 2.x to Xerces C++ 1.4.0

As mentioned, there are some major architectural changes between the 2.3.x and Xerces C++ 1.4.0 releases of the parser, and as a result the code has undergone significant restructuring. The list below mentions the public api's which existed in 2.3.x and no longer exist in Xerces C++ 1.4.0. It also mentions the Xerces C++ 1.4.0 api which will give you the same functionality. Note: This list is not exhaustive. The API docs (and ultimately the header files) supplement this information.

- `parsers/[Non]Validating[DOM/SAX]parser.hpp`
  These files/classes have all been consolidated in the new version to just two files/classes: `[DOM/SAX]Parser.hpp`. Validation is now a property which may be set before invoking the `parse`. Now, the `setDoValidation()` method controls the validation processing.
- The `framework/XMLDocumentTypeHandler.hpp` been replaced with `validators/DTD/DocTypeHandler.hpp`.
- The following methods now have different set of parameters because the underlying base class methods have changed in the 3.x release. These methods belong to one of `XMLDocumentHandler`, `XMLErrorReporter` or `DocTypeHandler` interfaces.
  - `[Non]Validating[DOM/SAX]Parser::docComment`
  - `[Non]Validating[DOM/SAX]Parser::doctypePI`
  - `[Non]ValidatingSAXParser::elementDecl`
  - `[Non]ValidatingSAXParser::endAttList`
  - `[Non]ValidatingSAXParser::entityDecl`
  - `[Non]ValidatingSAXParser::notationDecl`
  - `[Non]ValidatingSAXParser::startAttList`
  - `[Non]ValidatingSAXParser::TextDecl`
  - `[Non]ValidatingSAXParser::docComment`
  - `[Non]ValidatingSAXParser::docPI`
  - `[Non]Validating[DOM/SAX]Parser::endElement`
  - `[Non]Validating[DOM/SAX]Parser::startElement`
  - `[Non]Validating[DOM/SAX]Parser::XMLDecl`
  - `[Non]Validating[DOM/SAX]Parser::error`
- The following methods/data members changed visibility from `protected` in 2.3.x to `private` (with public setters and getters, as appropriate).
  - `[Non]ValidatingDOMParser::fDocument`
  - `[Non]ValidatingDOMParser::fCurrentParent`
  - `[Non]ValidatingDOMParser::fCurrentNode`
  - `[Non]ValidatingDOMParser::fNodeStack`
- The following files have moved, possibly requiring changes in the `#include` statements.
  - `MemBufInputSource.hpp`
  - `StdInInputSource.hpp`
  - `URLInputSource.hpp`
- All the DTD validator code was moved from `internal` to separate `validators/DTD` directory.
- The error code definitions which were earlier in `internal/ErrorCodes.hpp` are now splitup

into the following files:

- `framework/XMLErrorCodes.hpp` - Core XML errors
- `framework/XMLValidityCodes.hpp` - DTD validity errors
- `util/XMLExceptMsgs.hpp` - C++ specific exception codes.

The Samples

The sample programs no longer use any of the unsupported util/xxx classes. They only existed to allow us to write portable samples. But, since we feel that the wide character APIs are supported on a lot of platforms these days, it was decided to go ahead and just write the samples in terms of these. If your system does not support these APIs, you will not be able to build and run the samples. On some platforms, these APIs might perhaps be optional packages or require runtime updates or some such action.

More samples have been added as well. These highlight some of the new functionality introduced in the new code base. And the existing ones have been cleaned up as well.

The new samples are:

1. PParse - Demonstrates 'progressive parse' (see below)
2. StdInParse - Demonstrates use of the standard in input source
3. EnumVal - Shows how to enumerate the markup decls in a DTD Validator

Parser Classes

In the XML4C 2.x code base, there were the following parser classes (in the src/parsers/ source directory): NonValidatingSAXParser, ValidatingSAXParser, NonValidatingDOMParser, ValidatingDOMParser. The non-validating ones were the base classes and the validating ones just derived from them and turned on the validation. This was deemed a little bit overblown, considering the tiny amount of code required to turn on validation and the fact that it makes people use a pointer to the parser in most cases (if they needed to support either validating or non-validating versions.)

The new code base just has SAXParer and DOMParser classes. These are capable of handling both validating and non-validating modes, according to the state of a flag that you can set on them. For instance, here is a code snippet that shows this in action.

```
void ParseThis(const  XMLCh* const fileToParse,
               const bool validate)
{
  //
  // Create a SAXParser. It can now just be
  // created by value on the stack if we want
  // to parse something within this scope.
  //
  SAXParser myParser;

  // Tell it whether to validate or not
  myParser.setDoValidation(validate);

  // Parse and catch exceptions...
  try
  {
    myParser.parse(fileToParse);
  }
    ...
};
```

We feel that this is a simpler architecture, and that it makes things easier for you. In the above example, for instance, the parser will be cleaned up for you automatically upon exit since you don't have to allocate it anymore.

DOM Level 2 support

Experimental early support for some parts of the DOM level 2 specification have been added. These address some of the shortcomings in our DOM implementation, such as a simple, standard mechanism for tree traversal.

Progressive Parsing

The new parser classes support, in addition to the *parse()* method, two new parsing methods, *parseFirst()* and *parseNext()*. These are designed to support 'progressive parsing', so that you don't have to depend upon throwing an exception to terminate the parsing operation. Calling parseFirst() will cause the DTD (or in the future, Schema) to be parsed (both internal and external subsets) and any pre-content, i.e. everything up to but not including the root element. Subsequent calls to parseNext() will cause one more pieces of markup to be parsed, and spit out from the core scanning code to the parser (and hence either on to you if using SAX or into the DOM tree if using DOM.) You can quit the parse any time by just not calling parseNext() anymore and breaking out of the loop. When you call parseNext() and the end of the root element is the next piece of markup, the parser will continue on to the end of the file and return false, to let you know that the parse is done. So a typical progressive parse loop will look like this:

```
    // Create a progressive scan token
    XMLPScanToken token;

    if (!parser.parseFirst(xmlFile, token))
    {
      cerr << "scanFirst() failed\n" << endl;
      return 1;
    }

    //
    // We started ok, so lets call scanNext()
    // until we find what we want or hit the end.
    //
    bool gotMore = true;
    while (gotMore && !handler.getDone())
      gotMore = parser.parseNext(token);
```

In this case, our event handler object (named 'handler' surprisingly enough) is watching form some criteria and will return a status from its getDone() method. Since the handler sees the SAX events coming out of the SAXParser, it can tell when it finds what it wants. So we loop until we get no more data or our handler indicates that it saw what it wanted to see.

When doing non-progressive parses, the parser can easily know when the parse is complete and insure that any used resources are cleaned up. Even in the case of a fatal parsing error, it can clean up all per-parse resources. However, when progressive parsing is done, the client code doing the parse loop might choose to stop the parse before the end of the primary file is reached. In such cases, the parser will not know that the parse has ended, so any resources will not be reclaimed until the parser is destroyed or another parse is started.

This might not seem like such a bad thing; however, in this case, the files and sockets which were opened

in order to parse the referenced XML entities will remain open. This could cause serious problems. Therefore, you should destroy the parser instance in such cases, or restart another parse immediately. In a future release, a reset method will be provided to do this more cleanly.

Also note that you must create a scan token and pass it back in on each call. This insures that things don't get done out of sequence. When you call parseFirst() or parse(), any previous scan tokens are invalidated and will cause an error if used again. This prevents incorrect mixed use of the two different parsing schemes or incorrect calls to parseNext().

Namespace support

The C++ parser now supports namespaces. With current XML interfaces (SAX/DOM) this doesn't mean very much because these APIs are incapable of passing on the namespace information. However, if you are using our internal APIs to write your own parsers, you can make use of this new information. Since the internal event APIs must be able to now support both namespace and non-namespace information, they have more parameters. These allow namespace information to be passed along.

Most of the samples now have a new command line parameter to turn on namespace support. You turn on namespaces like this:

```
    SAXParser myParser;
    // Tell it whether to do namespace
    myParser.setDoNamespaces(true);
```

Moved Classes to src/framework

Some of the classes previously in the src/internal/ directory have been moved to their more correct location in the src/framework/ directory. These are classes used by the outside world and should have been framework classes to begin with. Also, to avoid name classes in the absense of C++ namespace support, some of these clashes have been renamed to make them more XML specific and less likely to clash. More classes might end up being moved to framework as well.

So you might have to change a few include statements to find these classes in their new locations. And you might have to rename some of the names of the classes, if you used any of the ones whose names were changed.

Loadable Message Text

The system now supoprts loadable message text, instead of having it hard coded into the program. The current drop still just supports English, but it can now support other languages. Anyone interested in contributing any translations should contact us. This would be an extremely useful service.

In order to support the local message loading services, we have created a pretty flexible framework for supporting loadable text. Firstly, there is now an XML file, in the src/NLS/ directory, which contains all of the error messages. There is a simple program, in the Tools/NLSXlat/ directory, which can spit out that text in various formats. It currently supports a simple 'in memory' format (i.e. an array of strings), the Win32 resource format, and the message catalog format. The 'in memory' format is intended for very simple installations or for use when porting to a new platform (since you can use it until you can get your own local message loading support done.)

In the src/util/ directory, there is now an XMLMsgLoader class. This is an abstraction from which any number of message loading services can be derived. Your platform driver file can create whichever type of message loader it wants to use on that platform. We currently have versions for the in memory format, the Win32 resource format, and the message catalog format. An ICU one is present but not implemented yet. Some of the platforms can support multiple message loaders, in which case a #define token is used to

control which one is used. You can set this in your build projects to control the message loader type used.

Both the Java and C++ parsers emit the same messages for an XML error since they are being taken from the same message file.

Pluggable Validators

In a preliminary move to support Schemas, and to make them first class citizens just like DTDs, the system has been reworked internally to make validators completely pluggable. So now the DTD validator code is under the src/validators/DTD/ directory, with a future Schema validator probably going into the src/validators. The core scanner architecture now works completely in terms of the framework/XMLValidator abstract interface and knows almost nothing about DTDs or Schemas. For now, if you don't pass in a validator to the parsers, they will just create a DTDValidator. This means that, theoretically, you could write your own validator. But we would not encourage this for a while, until the semantics of the XMLValidator interface are completely worked out and proven to handle DTD and Schema cleanly.

Pluggable Transcoders

Another abstract framework added in the src/util/ directory is to support pluggable transcoding services. The XMLTransService class is an abtract API that can be derived from, to support any desired transcoding service. XMLTranscoder is the abstract API for a particular instance of a transcoder for a particular encoding. The platform driver file decides what specific type of transcoder to use, which allows each platform to use its native transcoding services, or the ICU service if desired.

Implementations are provided for Win32 native services, ICU services, and the *iconv* services available on many Unix platforms. The Win32 version only provides native code page services, so it can only handle XML code in the intrinsic encodings ASCII, UTF-8, UTF-16 (Big/Small Endian), UCS4 (Big/Small Endian), EBCDIC code pages IBM037 and IBM1140 encodings, ISO-8859-1 (aka Latin1) and Windows-1252. The ICU version provides all of the encodings that ICU supports. The *iconv* version will support the encodings supported by the local system. You can use transcoders we provide or create your own if you feel ours are insufficient in some way, or if your platform requires an implementation that we do not provide.

Util directory Reorganization

The src/util directory was becoming somewhat of a dumping ground of platform and compiler stuff. So we reworked that directory to better spread things out. The new scheme is:

util - The platform independent utility stuff

- MsgLoaders - Holds the msg loader implementations
  1. ICU
  2. InMemory
  3. MsgCatalog
  4. Win32
- Compilers - All the compiler specific files
- Transcoders - Holds the transcoder implementations
  1. Iconv
  2. ICU
  3. Win32
- Platforms
  1. AIX

2. HP-UX
3. Linux
4. Solaris
5. ....
6. Win32

This organization makes things much easier to understand. And it makes it easier to find which files you need and which are optional. Note that only per-platform files have any hard coded references to specific message loaders or transcoders. So if you don't include the ICU implementations of these services, you don't need to link in ICU or use any ICU headers. The rest of the system works only in terms of the abstraction APIs.

11 Releases

Xerces C++ Version 1.5.1: July 18, 2001

Date	Contributor	Description
2001-07-17	Khaled Noaman	[Bug 2643] - derivation by extension of complex types does not permit addition of ONLY element content.
2001-07-16	Tinny Ng	[Bug 2410] DOMParser::parse() throws undocumented exceptions.
2001-07-16	Tinny Ng	[Bug 2512] typing mistake in code example of chapter "Constructing an XML Reader".
2001-07-16	Tinny Ng	APIDocs fix: default for schema processing in DOMParser, IDOMParser, and SAXParser should be false.
2001-07-15	James Berry	Add new files to UnionTypeValidator and ListDataTypeValidator to MacOS Project files.
2001-07-09	Khaled Noaman	Add constraint checking for simple types.
2001-07-11	Pei Yong Zhang	Fix to normalizeWhiteSpace: synchronize fDatatypeBuffer with toFill.
2001-07-05	Pei Yong Zhang	Add ListDatatypeValidator and UnionDatatypeValidator.
2001-07-10	Tinny Ng	Give proper error messsage when scanning external id.
2001-07-10	Tinny Ng	The first char of PI Target Name should be checked.
2001-07-09	Khaled Noaman	Add <any> declaration.
2001-07-09	Khaled Noaman	Fixes for import/include declarations.
2001-07-09	Tinny Ng	Partial Markup in Parameter Entity is validity constraint and thus should be just error, not fatal error.
2001-07-08	James Berry	Add new samples projects: IDOMPPrint and SAX2Print for ProjectBuilder

2001-07-08	James Berry	Update ProjectBuilder Xerces project for latest file additions.
2001-07-08	James Berry	[Bug 2486] Files missing from XercesLib.mcp.
2001-07-08	James Berry	Add new samples for CodeWarrior build: IDOMPrint and SAX2Print.
2001-07-08	James Berry	New file for use in building Carbon samples.
2001-07-08	James Berry	Simplify file existance checks.
2001-07-08	James Berry	[Bug 2495] Missing (in xerces-c-src1_5_0/obj/Makefile.in.
2001-07-08	James Berry	Fix clean and distclean targets; broken because rm fails if passed no files.
2001-07-06	Tinny Ng	[Bug 2472] Linker options ignored on IRIX.
2001-07-06	Martin Kalen	Automatic build of single-threaded library.
2001-07-05	Tinny Ng	Encoding String must present for external entity text decl.
2001-07-05	Tinny Ng	Standalone checking is validity constraint and thus should be just error, not fatal error.
2001-07-05	Pei Yong Zhang	Add NotationDatatypeValidator, QNameDatatypeValidator and ENTITYDatatypeValidator.
2001-07-04	Pei Yong Zhang	Add IDREFDatatypeValidator and IDDatatypeValidator.
2001-07-04	Pei Yong Zhang	XMLString:isValidName(): to validate Name (XML [4][5]).
2001-07-03	Tinny Ng	Some compilers (e.g. the HP compiler) has mistaken the parameter 'std', which is short for standalone as the special prefix used by the standard libraries.
2001-07-03	Miroslaw Dobrzanski-Neumann	Supporting dce threading on AIX and Solaris.
2001-06-27	David Bertoni	[Bug 2365] Huge performance problem with the parser in XMLScanner::sendCharData().
2001-06-27	David Bertoni	[Bug 2363] XMLScanner::sendCharData() can send the wrong length to the handler.
2001-06-27	Khaled Noaman	[Bug 2353] Validating Parser parses after validation failed.
2001-06-27	Murray Cumming	[Bug 1147] Headers install in wrong directory.
2001-06-26	Tinny Ng	[Bug 2119] DOMString::print() should use DOMString::transcode() for transcoding.
2001-06-25	Stephen Dulin	OS390 updates.

2001-06-25	Linda Swan	AS400 updates.
2001-06-25	Pei Yong Zhang	[Bug 1393] Converting from Unicode to iso8859.
2001-06-25	Matt Lovett	[Bug 965] scanDocTypeDecl messes up the source offsets.
2001-06-25	Khaled Noaman	Add constraint checking on elements in complex types.
2001-06-22	James Berry	[Bug 2277] Bad argument to ConvertFromUnicodeToText.
2001-06-22	Pei Yong Zhang	[Bug 2263] 'SIZE' : redefinition (BooleanDatatypeValidator.cpp).
2001-06-22	Khaled Noaman	[Bug 2258] Bug in Iconv and Iconv390.
2001-06-22	Tinny Ng	[Bug 2225] assignment vs. comparison in if clause.
2001-06-22	Tinny Ng	[Bug 2257] 1.5 thinks a ?xml-stylesheet ...> tag is a <?xml ...> tag.
2001-06-21	Khaled Noaman	[Bug 1946] Standalone validity check only for external decl.
2001-06-21	Tinny Ng	[Bug 2262] Duplicated header guard.
2001-06-20	Pei Yong Zhang	Proper Debug Guard: Reported by Dean.
2001-06-19	Tinny Ng	Namespace should be off by default in XMLScanner.
2001/06/19	Tinny Ng	Add installAdvDocHandler to SAX2XMLReader as the code is there already.
2001-06-19	Khaled Noaman	Handle maxChars > length(toTranscode).
2001-06-18	Erik Rydgren	Memory leak fix: to addlevel().
2001-06-18	Khaled Noaman and Pei Yong Zhang	Add support for 'fixed' facet.
2001-06-15	Khaled Noaman	Added constraint checking for ref on elements.
2001-06-15	Tinny Ng	ICU 1.8.1 update.

Xerces C++ Version 1.5.0: June 15, 2001

Date	Contributor	Description
2001-06-15	Tinny Ng	Schema: Add Schema support in XMLParsers (DOM/SAX/SAX2), XMLScanner. Create SchemaValidator. Add Grammar Model. Support xsi:nil. Support xsi:schemaLocation and xsi:noNamespaceSchemaLocation. Update samples to enable schema.

2001-06-15	Tinny Ng	Break DTDValidator into DTDGrammar, DTDScanner, and DTDValidator.
2001-06-15	Tinny Ng	IDOM: Complete the Range, TreeWalker, NodeIterator, and other memory fixes. Support IDOM on UNIX platform. Add samples IDOMPrint, and IDOMCount. Add test cases IRangeTest and ITraversal.
2001-06-15	Khaled Noaman	Schema: Add Regular Expression. Add Schema Messages. Add Schema Simple Type Support. Add Schema Complex Type Support (Except Group). Add Schema Attribute Declarations support. Add Schema Element Declarations support. Support Simple Content and Complex Content. Support Element and attribute reuse using "ref". Support Schema Choice and Sequence. Support Schema Import and Include.
2001-06-15	Khaled Noaman	DatatypeValidator: Add DatatypeValidator and DatatypeValidatorFactory.
2001-06-15	PeiYong Zhang	Schema: Add Schema support in Content Model. Add Schema Exception Handling. Add Schema XUtil. Add QName Support. Support SubstitutionGroup.
2001-06-15	PeiYong Zhang	DatatypeValidator: Support Base64DatatypeValidator, BooleanDatatypeValidator, DecimalDatatypeValidator, HexBinDatatypeValidator, StringDatatypeValidator, InvalidDatatypeFacetException, InvalidDatatypeValueException.
2001-06-13	Erik Rydgren	[Bug 812] Memory leak with multiple !ATTLIST on single !ELEMENT.
2001-06-08	Tinny Ng	[Bug 2043] XMLFormatter unallocates arrays incorrectly.
2001-06-08	PeiYong Zhang	Documentation and project files update for Xerces 1.5.
2001-06-08	Khaled Noaman	IDOM Documentation.
2001-06-07	Khaled Noaman	Fix no error message for faulted-in attributes if reuse grammar for 3+ times.
2001-06-06	Peter A. Volchek	/Platforms/Win32/Win32PlatformUtils.cpp Include stdlib.h.
2001-06-06	James Berry	Update Mac OS ProjectBuilder projects.
2001-06-06	James Berry	Fix invalid file references in project.

2001-06-06	James Berry	/src/util XMLString.cpp Clean up compiler warning.
2001-06-06	James Berry	/src/util/regx RegxParser.cpp Fix two improper NULL tests.
2001-06-05	James Berry	Add support for Mac OS X command line configuration and build.
2001-06-5	Peter A. Volchek	Add 'const' to getGrammar.
2001-06-04	PeiYong Zhang	The start tag "<?xml" could be followed by (#x20 | #x9 | #xD | #xA)+.
2001-06-04	James Berry	Add support for tracking error count during parse; enables simple parse without requiring error handler.
2001-06-01	Tinny Ng	/scripts/packageSources.pl Keep the BCB4 project files in the source package.
2001-05-22	James Berry	Check for existance of MacOS Unicode Converter routines prior to instanciating our transcoder object; Xerces will thus panic, rather than crash, if they don't exist. Add support to check for existance of MacOS Unicode Converter to avoid calling through NULL pointer.
2001-05-16	Henry Zongaro	IDOM: Add DeepNodeList support.
2001-05-16	Henry Zongaro	IDOM: Add namespace support.
2001-05-10	Christian Schuhegger	[Bug 1158] built-in buffer limit could be smaller than system limit, use PATH_MAX instead.
2001-05-10	Arnaud LeHors	[Bug 1605] AttrNSImpl.cpp: fixed typo in constructor.
2001-05-09	Curt Arnold	[Bug 1500] The public id was set twice and the system id was not set on Notations.
2001-05-04	Tinny Ng	DOMPrint: Check error before continuing.
2001-05-03	Tinny Ng	ICU 1.8 update.
2001-05-03	Khaled Noaman	Added new option to the parsers so that the NEL (0x85) char can be treated as a newline character.
2001-04-23	Erik Rydgren	DTDScanner: Reuse grammar should allow users to use any stored element decl as root.
2001-04-19	William L Hopper	Win32PlatformUtils: InterlockedCompareExchange on different Windows.
2001-04-19	William L Hopper	BCB project changes.
2001-04-16	James Berry	MacOSUnicodeConverter: Fix include path, Updates to reflect changes for Mac OS X final and Update MacOS projects for Mac OS X final ProjectBuilder.
2001-04-11	Arnaud LeHors	[Bug 1303] AttrImpl: allow value to be set to null.
2001-04-11	Tinny Ng	DOMParser: Attribute default values not printed in document type internal subset interface.
2001-04-10	Tinny Ng	createdocs.bat: fix PDF generation.
2001-04-04	Alberto Massari	DTDElementDecl: Error checking for null content spec.
2001-04-02	Andy Heninger	IDOM: imported.
2001-04-02	Andy Heninger	IThreadTest: imported.
2001-03-30	Tinny Ng	[Bug 1150] Problems with Namespaces and validating parsing.

2001-03-27	Roman Sulzhyk	[Bug 1069] Explicit Makefile dependency for 'lib' build.
2001-03-26	PeiYong Zhang	When Standalone="yes", it is NOT supposed to accept element which is defined in external DTD with #FIXED attribute.
2001-03-26	Andy Heninger	Update packageBinaries.pl for ICU 1.8. ICU debug .lib file names and locations changed.
2001-03-23	Jeff Harrell	[Bug 1018] AutoSense looks for "IRIX" when it should look for "sgi" or "__sgi".
2001-03-22	Roman Sulzhyk	[Bug 1069] The Makefiles fail to locate .cpp - > .o dependency and rebuild .o all the time.
2001-03-22	John Rope	[Bug 1021] Accessing an XML file using the file "protocol" and a UNC path fails to open the file.
2001-03-09	Tinny Ng	[Bug 733] Seg fault when trying to parse empty filename.
2001-03-06	Tinny Ng	[Bug 677] Infinite loop caused by malformed XML. Happen when namespace is on.
2001-03-02	Martin Kalen	Enabling libWWW NetAccessor support under UNIX. Tested with latest tarball of libWWW (w3c-libwww-5.3.2) under RedHat Linux 6.1.
2001-02-27	Tinny Ng	[Bug 676] Linux for S/390 build requires -fPIC.
2001-02-22	Tinny Ng	[Bug 678] StdInParse doesn't output filename or duration.
2001-02-21	Matt Lovett	ICUTranscoder::transcodeFrom() expects ICU function ucnv_toUnicode to return an extra element in fSrcOffsets to allow us to figure out the last char size, which in fact it is not. The fix is to compute the last char size ourselves using the total bytes used.
2001/02/16	Andy Heninger	Change limit test to reduce spurious pointer assignment warnings from BoundsChecker.
2001-02-14	Bob Kline	Better FAQ for the checksum error.
2001-02-14	Mark Everline	Core dump when UTF-16 encoding contradicts actual encoding.
2001-02-13	Hiram Clawson	Update samples/tests files for on UnixWare 7.1.1 with gcc 2.95. Add UNIXWARE platform defines to Makefile.incl, add recognition of sysv5uw7 to configure.in, and add unixware as recognized platform to runConfigure.
2001-02-09	Martin Kalen	Update support for SCO UnixWare 7 (gcc). Tested under UnixWare 7.1.1 with gcc version 2.95.2 19991024 (release) with gmake 3.79.1.
2001-02-08	Martin Kalen	Enable COMPAQ Tru64 UNIX machines to build xerces-c with gcc (tested using COMPAQ gcc version2.95.2 19991024 (release) and Tru64 V5.0 1094).
2001-02-07	Bill Schindler	Rearranged statements in Initialize() so that platformInit() is called before an XMLMutex is created.
2001-02-07	Richard Ko	Storage overlay in ucnv_setFromUCallBack.
2001-02-05	Tinny Ng	[Bug 766] /src/util/Compilers/CSetDefs.hpp: define NO_NATIVE_BOOL macro only if not pre-defined/reserved.

2001-02-05	Jordan Naftolin	Add createPDF.jar and apachPDFStyle.xsl to convert documentation xml files to pdf format.

Release Archive

For release information about Xerces C++ 1.4.0 or earlier, please refer to Release Archive.

12
Releases Archive

Xerces C++ Version 1.4.0: January 31, 2001

Date	Contributor	Description
2001-01-26	Walker Curtis	Undefined symbol error when building a single threaded version of the xerces lib on irix.
2001-01-25	Arnaud LeHors	Added a flag to turn off error checking in the DOM, this is primarily used while building the DOM from the parser to get better performance.
2001-01-25	Khaled Noaman	Let users add their encoding to the intrinsic mapping table.
2001-01-25	Khaled Noaman	const should be used instead of static const. And other clean up bug fixes.
2001-01-24	Arnaud LeHors	Fixed replaceChild to handle the case where a node is replaced by itself. Cleaned up insertBefore.
2001-01-24	Tinny Ng	Guard the use of '-ptr${OUTDIR}' in EnumVal/Makefile.in
2001-01-22	Curt Arnold.	Loads winsock dynamically.
2001-01-19	Curt Arnold.	COM various updates: updated the GUID's so both can coexist, better error reporting and fixed a new minor bugs.
2001-01-18	Bill Schindler	FAQ spell check, fix typos, fix grammar, readability editing, clean up formatting, re-organize so related topics appear together.
2001-01-18	Bill Schindler	Project file updated due to removal of ChildAndParentNode.cpp.
2001-01-17	Arnaud LeHors	DOM Implementation Optimization.
2001-01-17	Volker Krause	ElementImpl::getAttributeNS should check null pointer.
2001-01-17	Arnaud LeHors	Have a single counter global to the document. Removed node basis change counter.
2001-01-17	Arnaud LeHors	Removed unused field in NodeImpl that was left over.
2001-01-17	Tinny Ng	Access violations and stack overflows in insertBefore.
2001-01-15	David Bertoni	Performance Patches.
2001-01-12	Tinny Ng	Fix style-ibm.zip for documentation generation.
2001-01-12	Tinny Ng	Remove the two obsolete file: stylesheets\Copy of book2project.xsl and stylesheets\Copy of document2html.xsl in style-apachexml.jar

2001-01-12	Tinny Ng	Documentation Enhancement: explain values of Val_Scheme.
2001-01-12	Tinny Ng	Documentation Enhancement: Add list of SAX2 feature strings that are supported.
2001-01-04	Khaled Noaman	Assertion `size > 0' failure when cloning a node if the last attributes has been removed.
2000-12-28	James Berry	Omit include carbon.h in favor of specific include files.
2000-12-28	James Berry	Add or modify cvs header in various files.
2000-12-28	James Berry	Eliminate compiler warning in RangeImpl.cpp.
2000-12-28	James Berry	Replace include of Carbon.h with specific include files.
2000-12-28	James Berry	Move away from include of Carbon.h; include only needed files instead. Fix bug in parsing of upwardly relative paths under classic (thanks to Lawrence You).
2000-12-22	Tinny Ng	XMLUni::fgEmptyString which is defined as "EMPTY" is incorrectly used as an empty string; in fact XMLUni::fgZeroLenString should be used instead.
2000-12-22	Tinny Ng	Add the new header LexicalHandler.hpp to Makefile.in.
2000-12-22	Murray Cumming	removes '-instances=static' from the Linux link sections.
2000-12-22	David Bertoni	SAX2-ext's LexicalHandler support.
2000-12-14	Tinny Ng	Better instruction for using packageBinaries.pl. Use symbol XercesCInstallDir and XercesCSrcInstallDir instead of hardcoding the Xerces version number in the file.
2000-12-14	Tinny Ng	Fix API document generation warning: "Warning: end of member group without matching begin".
2000-12-14	Tinny Ng	Add RangeTest as part of the xerces-all MSVC++ workspace.
2000-12-12	Gareth Reakes	null pointer bug.
2000-12-08	Tinny Ng	Entity Reference cleanup dumping core if the last entity reference is deleted.
2000-12-06	Tinny Ng	fix the link to FAQ.
2000-12-06	Tinny Ng	further fixes to Range, and update RangeTest.cpp with more test coverage.
2000-11-30	Bill Schindler	Spell check, fix typos, fix grammar, readability editing, clean up formatting.
2000-11-30	Bill Schindler	Remove dead code (old StdOut and StdErr functions); minor clean-up.
2000-11-30	Tinny Ng	patch to fix a number of Range problems. See mail of 11/21/2000.
2000-11-30	Tinny Ng	DOM_Text::splitText(), fix off by one error in the test for index too big error.
2000-11-30	Tinny Ng	reuseValidator - fix bugs (spurious errors) that occured on reuse due to pools already containing some items.
2000-11-08	Andrei Smirnov	Build updates for Solaris 2.8 64 bit.
2000/11/07	Tinny Ng	Bug fix for DTD entity reference problem reported by Tony Wuebben on 10/25.
2000-11-07	Tinny Ng	config.guess and config.sub updated to newer versions.
2000-11-07	Pieter Van-Dyck	Change InterlockedCompareExchange for compatibility with Borland BCB5

2000-11-07	Pieter Van-Dyck	Fix incorrect version number in gXercesMinVersion.
2000-11-01	Tinny Ng	SAX bug fix: Attribute lists were throwing exceptions rather than returning null when an attribute could not be found by name.
2000-11-01	Tinny Ng	Scanner bug fix: with progessive parsing, namesapce and validation options were not being set correctly. Symptoms included failure to detect ignorable white space.
2000-10-31	Tinny Ng	DOM NodeIterator bug fix: iterators would sometimes continue beyond their starting (root) node.
2000-10-20	Andy Heninger	DOMParser bug fix - erroneous attempt to look up name space URIs while scanning default attribute values in DTD removed. Was a crashing bug when namespaces were enabled.
2000-10-20	Andy Heninger	DOM NodeFilter - define values for FilterAction enum to match those in the DOM spec.
2000-10-19	Andy Heninger	SAXCount sample, allow multiple files on command line. DOMCount sample, rename error handler class to say that it is an error handler.
2000-10-18	James Berry	MacOS project file updates. Small code optimization. Add comments to clarify and to reflect new fixed XMLCh size.
2000-10-17	Andy Heninger	Bug Fix - problems with multi-byte characters on input buffer boundaries.
2000-10-17	Andy Heninger	DOMPRintFormatTarget, bad override of writeChars fixed (missing const). XMLFormatTarget, removed version of writeChars with no length. Can not be safely used, and obscured other errors.
2000-10-16	Andy Heninger	Change XMLCh back to unsigned short on all platforms
2000-10-13	Devin Barnhart	COM: interpret BSTR as UTF-16 in documents
2000-10-13	Edward Bortner	Solaris: change detection for native support for type bool to defined(_BOOL).
2000-10-13	Nadav Aharoni	MXLString::trim() bug fix: failure to null terminate result.
2000-10-10	Bill Schindler	XMLFormatter: Fix problems with output to multi-byte encodings.
2000-10-10	Andy Heninger	From Janitor, remove the addition that is having compile problems in MSVC.
2000-10-10	James Berry	Fix a bug in returned length of transcoded string. Add a few comments.
2000-10-09	James Berry	ProjectBuilder project to build Xerces.

2000-10-09	James Berry	Numerous Changes: - Increase environmental sensitivity with hope of supporting pre OS 9 OS versions. - Enhanced path creation/interpretation to support proper unix style paths under Mac OS X instead of the volume rooted paths we previously used. Paths under Classic remain the same. - Better timer resolution. - Detect functionality via unresolved symbols rather than Gestalt where possible. - Softly back away from URLAccess...if it's not installed, we just don't support a net accessor. - Additional support for XMLCh/UniChar size differences under GCC on Mac OS X. - Fix Mac OS X support. GCC in this environment sets wchar_t to a 32 bit value which requires an additional transcoding stage (bleh...) - Improve sensitivity to environment in order to support a broader range of system versions. - Fix a few compiler sensitivities. - Carbon.h header support
2000-10-09	James Berry	Add some auto_ptr functionality to allow modification of monitored pointer value. This eases use of Janitor in some situations.
2000-10-09	James Berry	Autosense.hpp: modify sensing of Mac OS X.
2000-09-28	Andy Heninger	DOM_Document::putIdentifier() removed. There never was an implementation for this function.
2000-09-28	Curt Arnold	COM wrappers updated.
2000-09-28	Linda Swan	AS400 related changes.
2000-09-28	Andy Heninger	DOM_Document - remove the un-implemented function putIdentifier() from the header.
2000-09-28	Andy Heninger	DOMParser MemoryLeak fixed. Occured when a document redefined the a builtin entity, e.g. <.
2000-09-28	Andy Heninger	DOMPrint sample: add deletes before exit so boundschecker runs cleanly.
2000-09-22	James Berry	Change file access permissions to fsRdPerm. Since we never write, there's no reason to request write access. Thanks to John Mostrom @ Adobe. Also nuke a few spaces and the entire defunct support for reading directly from MacOS resources.
2000-09-22	Arundhari Bhowmick	DOM Parser: internal subset entity printing update.

Xerces C++ Version 1.3.0: Sept 21, 2000

Date	Contributor	Description
2000-09-21	Torbjörn Bäckström	HPUX - Incorrect use of Array Janitor in Platform Utils removed.
2000-09-21	Arundhati Bhowmick	DOMPrint - DTD internal subset, printing of attribute value enumerations was broken.
2000/09/19	Arundhati Bhowmick	DOMPrint - output entity reference nodes as XML entity references, instead of just printing their children.
2000-09-19	Bill Schindler	OS/2 - port update
2000-09-18	Arundhati Bhowmick	DOM EntityReferences, fixed bugs with length() and hasChildNodes() methods.

2000-09-12	Arundhati Bhowmick	DOM: changed name of expandEntityReferences option to createEntityReferenceNodes. More accurately describes what it does. Fixed bugs that caused creation of Entity Reference nodes to fail.
2000-09-12	IBM	AS400 - transcoder updates.
2000-09-11	Shengkai Qu	OS390 - makefile updates
2000-09-11	Kirk Wylie	Alpha processor support update in config.sub.
2000-09-08	Kirk Wylie	Reordered member variables in ThrowEOEJanitor.
2000-09-08	Arnaud LeHors	DOM NamedNodeMap - because in many cases we may have to deal with both nodes with a namespace and nodes without any, NS methods through findNamePoint must handle both types of nodes.
2000-09-08	Kirk Wylie	Some destructors not virtual that should have been; some members of DOM_Entity virtual that should not have been.
2000-09-08	Andy Heninger	Removed incorrect detection of nested CDATA sections. Problem reported by Johannes Lipp.
2000-09-08	Andy Heninger	DOMPrint incorrectly handled DOCTYPE declarations containing both a public and system id. Problem reported by Jesse Pelton.
2000-09-08	Radovan Chytracek	MSVC: RangeTest project settings incorrect, build failed.
2000-09-07	Bob Kline	XMLReader::skippedString(), failed under certain rare circumstances.
2000-09-07	Andy Heninger	Fix SAXException assignment operator. Now non-virtual, and SAXParseException subclass invokes base class operator.
2000-09-06	William L. Hopper	Borland updates. It had fallen way behind.
2000-09-06	Andy Heninger	HPUX 11, packageBinaries build script, DCEThreads no longer default
2000-09-06	James Berry	Macintosh: Add support for new compile time options defined in prefix file. These control the selection of the msgloader, transcoder, and netaccessor. Add a tiny bit of robustness to the nasty panic method..
2000-09-06	Shengkai Qu	S390: socket related changes
2000-09-06	James Berry	Macintosh: Allow ShortenFiles to work even when destination directory already exists.
2000-09-06	Arundhati Bhowmick	HP compile options modified for ICU compatibility
2000-09-05	Michael Crawford	Macintosh: Fix atomic increment & decrement to return value after operation rather than before.
2000-09-05	Andy Heninger	Cleaned up various compiler warnings.
2000-09-05	Andy Heninger	SAX parser: added advanced callback support for XMLDecl
2000-09-01	Andy Heninger	Fix ICU transcoding service, crashing bug on Linux, Solaris
2000-08-30	Andy Heninger	Builds - clean up a number of compiler warnings.

2000-08-24	Andy Heninger	DOMPrint - fixed crash when input xml file was not found.
2000-08-23	Andy Heninger	Build Script updates and cleanups
2000-08-18	Andy Heninger	Version number bumped to 1.3 in preparation for the upcoming xerces 1.3 / xml4c 3.3 release
2000-08-17	Arnaud Lehors	DOM: Rewrote code updating the linked list on node addition and removal. I believe it is now easier to read and it uses fewer tests so it is also a little faster.
2000-08-17	Arnaud Lehors	DOM: small cleanup: renamed a set of [] boolean flag methods. yes, I know, I also wish I got them right in the first place...
2000-08-17	Sumit Chawla	PTX port updates
2000-08-16	Andy Heninger	Fixed crash when XML text content has very long lines. Bug pointed out by Simon Fell.
2000-08-14	Joe Polastre	SAX2 DefaultHandler, inconsistency in const parameters fixed.
2000-08-11	Arundhati Bhowmick	ICU Transcoding - updates to support ICU 1.6
2000-08-09	Arundhati Bhowmick	DOM Range: Add const to API where appropriate.
2000-08-09	Joe Polastre	Many conformance and stability changes: - ContentHandler::resetDocument() removed - attrs param of ContentHandler::startDocument() made const - SAXExceptions thrown now have msgs - removed duplicate function signatures that had 'const' [eg: getContentHander()] - changed getFeature and getProperty to apply to const objs - setProperty now takes a void* instead of const void* - SAX2XMLReaderImpl does not inherit from SAXParser anymore - Reuse Validator (http://apache.org/xml/features/reuse-validator) implemented - Features & Properties now read-only during parse
2000-08-09	Joe Polastre	Namespaces bug - bogus default namespace removed.
2000-08-09	Joe Polastre	SAXException enhanced, messages added.
2000-08-08	Joe Polastre	SAX2Count - new sample program for SAX2.
2000-08-07	Arundhati Bhowmick	Remove detach() method from TreeWalker.
2000-08-03	James Berry	Add Mac Codewarrior projects.
2000-08-01	Joe Polastre	SAX2 support added
2000-08-01	Gary Gale	Compaq Tru64 port added.
2000-07-31	Joe Polastre	bug fix in removeAll() to zero out all the pointers.
2000-07-31	Andy Heninger	utf-8 byte order mark recognition

2000-07-29	James Berry	Mac OS Port, general cleanups.
2000-07-28	James Berry	Addition of NetAccessor functionality for MacOS, built on URLAccess library.
2000-07-28	Arundhati Bhowmick	ICU Transcoding service: changes for move to ICU 1.6
2000-07-27	Arundhati Bhowmick	DOM Range added. (Major new feature)
2000-07-27	Murray Cumming	makefile fixes for SUNW_0.7
2000-07-25	Arundhati Bhowmick	XMLCh character constants definitions moved to XMLUniDefs.h. Removes name clashes with application defined symbols.
2000-07-25	Joe Polastre	allow nesting of PlatformUtils::Init() and Terminate()
2000-07-25	Gary Gale	ICU transcoding: fix off by one error.
2000-07-21	<check>	Change wcsupr to _wcsupr
2000-07-21	Eric Schroeder	Win32TransService - fix error in use of hashtables
2000-07-21	Joe Polastre	DOMPrint: fixed error in handling of null CDATA sections.
2000-07-20	Andy Heninger	Improved net access (parse of URLs). Still weak, though.
2000-07-20	Erik Schroeder	XMLScaner.cpp bugfix: call startDocument() at beginning of scan.
2000-07-20	Arundhati Bhowmick	DOMCount exception handling cleaned up.
2000-07-19	Todd Collins	runConfigure: modified to take "configureoptions"
2000-07-19	<check>	Add 'make install' target to src/util/Platforms/Makefile.in
2000-07-19	<check>	DOM: BugFix: DocumentType nodes can not have children.
2000-07-19	<check>	DOM: Bug in NodeIDMap constructor.
2000-07-18	Anupam Bagchi	Documentation generation tools updated.
2000-07-17	James Berry	Mac OS port brought up to date (was very old)
2000-07-17	Andy Heninger	Change windows project to link with ws2_32.lib instead of winsock32.lib
2000-07-17	Grace Yan, Joe Kesselman	DOM NodeIterator: bug fix for SHOW_ELEMENT flag incorrectly being retrieved.
2000-07-17	Joe Polastre	switched scanMisc() with endDoc() in scanNext. Pointed out by Dean Roddey.
2000-07-17	Jim Reitz	fix for uninitialized variable gotData bug in XMLScanner.cpp.
2000-07-12	Arundhati Bhowmick	DOM: fix bug in setting previous sibling pointer during insertNode
2000-07-07	Joe Polastre	Update to use of hashtables.
2000-07-07	Joe Polastre	DOM userdata: several bug fixes.
2000-07-06	Andy Heninger	Speedups in XMLScanner, XMLReader
2000-07-07	<check>	bug fixes in IXMLDOM*
2000-07-06	Joe Polastre	Performance tweaks, added more inlines.
2000-07-05	Anupam Bagchi	Documentation updates.

2000-07-05	Joe Polastre	DOM: Attribute node default value handling implemented.
2000-07-05	Joe Polastre	DOM Attr nodes - fixed setting of specified when cloning. (change may be in error)
2000-07-04	Dean Roddey	Fixed a memory leak when namespaces are enabled.
2000-06-28	Curt Arnold	COM object usage documentation update.
2000-06-28	Joe Polastre	DOM Userdata - put pointers in a hash table rather than having one pre-allocated per node. Memory footprint reduction.
2000-06-27	Joe Polastre	extended the (implementation) hash table classes.
2000-06-26	John Roper@iOra.com	Bug fix: check if initialized in Terminate() to stop access violations.
2000-06-26	<check>	Solaris build - template directory related changes.

Xerces C++ Version 1.2.0: June 22, 2000

2000/06/22	<check>	OS/2 Port updated.
2000-06-22	Joe Polastre	DOM Attr nodes, specified flag not set correctly by parser. Fixed.
2000-06-20	Rahul, Joe, Arundhati	Many doc updates in preparation for release of version 1.2
2000-06-19	Rahul Jain	Update Package Binaries script to build Xerces with ICU.
2000-06-19	Joe Polastre	Added help messages to PParse and StdInParse samples.
2000-06-19	Joe Polastre	Changed "XML4C" to "Xerces-C" in DOMPrint. (Missed in earlier mass name change.)
2000-06-19	Arundhati Bhowmick	Moved version.incl up one directory level.
2000-06-19	Curt Arnold	Improved Windows project file.
2000-06-16	John Smirl	Bug Fix: Document Handler was not called for PIs occurring before the document element. Bug identified by John Smirl and Rich Taylor
2000-06-16	Rahul Jain	DOMPrint, SAXPrint: remove extra space in printing PIs.
2000-06-16	Rahul Jain	Windows Debug Build: add 'D' suffix to DLL name in VCPPDefs.hpp
2000-06-16	Rahul Jain	Samples: added -v option (validate always). Needed for testing scripts.
2000-06-14	Joe Polastre	Fixed null ptr failures in DOM NamedNodeMap
2000-06-12	Andy Heninger	Fixed bug in XMLString::trim(), reported by Michele Laghi
2000-06-07	Joe Polastre	DOM: reduced memory usage for elements with no attributes.
2000-06-01	Andy Heninger	DOMString - add const to return type of const XMLCh *DOMString::rawBuffer()

2000-06-01	Arundhati Bhowmick	Fix crash with Solaris optimized build. Modified XMLURL.cpp to dodge compiler code generation error.
2000-06-01	Joe Polastre	Bug fix: DOM Attr Specified flag was incorrectly set when cloning or importing attributes.
2000-05-31	Andy Heninger	MSVC projects modified to produce separate debug and release versions of Xerces lib and dll.
2000-05-31	Rahul Jain	Bug fix: DOMPrint, SAXPrint produced garbage output on Solaris. Solaris library problem.
2000-05-31	Joe Polastre	Fixed incorrect error check for end of file in Win32 platform utils.
2000-05-31	Rahul Jain	DOMPrint enhancements. Add options for specifying character encoding of the output, better control over escaping of characters, better handling of CDATA sections. Default validation is now "auto"
2000-05-22	Dean Roddey	XMLFormatter now escapes characters, as reqd., occurring midway in strings. Reported by Hugo Duncan.
2000-05-22	Andy Heninger	Bug fix in implementation of DOM_Document::GetElementById()
2000-05-18	Anupam Bagchi	Documentation, DTD for source xml files moved into xerces-c project, sbk: prefixes removed, xml can now be validated locally.
2000-05-15	Dean	Fixed 'fatal error' when 'reusing the validator' problem reported by Rocky Raccoon (rrockey@bigfoot.com). Fix submitted by Dean Roddey (droddey@charmedquark.com).
2000-05-15	James Berry	Changed #include <memory.h> to <string.h> everywhere. <jberry@criticalpath.com>
2000-05-15	Andy H.	DOMTest: removed incorrectly failing entity tests
2000-05-12	Andy H.	Revised implementation of DOMDocument::getElementsById(), removed memory leaks, new test program for it.
2000-05-12	Dean	Bug fix - A PE ref appearing at the start of a skipped conditional section was incorrectly being processed rather than ignored. Fix from Dean Roddey.
2000-05-11	Rahul Jain	Start using the socket based netaccessor by default on most Unix platforms.
2000-05-11	Rahul Jain	Update ICUTransService to work with latest revision of ICU which provides a hard linked data DLL. i.e. icudata.dll will be loaded when xerces-c is loaded.
2000-05-05	Dean	Problem with progressive parsing. parseNext() would through an exception when the document contains entities, either or external.
2000-05-11	Sean MacRoibeaird	Add missing validity checks for stand-alone documents, character range and Well-formed parsed entities.

2000-05-10	Radovan Chytracek	Fix compilation problems on MSVC 5. Radovan.Chytracek@cern.ch>
2000-05-10	Dean	Fix XMLReader defect reported by SHOGO SAWAKI
2000-05-09	Andy H	Fix problem with Windows filenames containing '\' in Japanese and Korean encodings.
2000-05-08	Andy H	Memory Cleanup. XMLPlatformUtils::Terminate() deletes all lazily allocated memory
2000-05-05	Dean	Fixed defect in progressive parsing 'parseNext()' reported by Tim Johnston
2000-05-03	Tom Jordahl	Fixed Solaris build problems with static character constants. Tom Jordahl <tomj@allaire.com>
2000-04-28	Arnaud LeHors	Reduced memory usage for DOM Attributes.
2000-04-28	boercher@kidata.de	New runConfigure options -P and -C
2000-04-27	Andy H	Memory leaks in TransService. Joseph Chen JosephC@plumtree.com>
2000-04-27	Arnaud LeHors	DOM - storage requirements for nodes substantially reduced.
2000-04-27	Arundhati	Added DOM XMLDecl node type; provides access to XML declaration.
2000-04-20	Arundhati	Added DOM access to DTD subset (DOM Level 2 feature)
2000-04-19	Anupam Bagchi	API document generation changed to DOxygen from Doc++
2000-04-18	Arundhati	Full support for DOM_EntityReference, DOM_Entity and DOM_DocumentType introduced
2000-04-18	Dean Roddey	Don't allow spaces before PI target. Bug #42
2000-04-17	Anupam Bagchi	Follow the SMP/E procedures for the OS/390 BATCH install
2000-04-12	Dean Roddey	Auto-validate mode. Validate only when a DTD is present.
2000-04-11	Dean Roddey	If a SAX error handler is installed, then the resetErrors() event handler should call the one on the installed SAX error handler.
2000-04-10	Dean Roddey	Allow an empty DOCTYPE declaration, with just the root name.
2000-04-06	Dean Roddey	Add low level support for transcoding XML output to different character encodings.
2000-04-06	Arnaud Lehors	DOM node memory footprint reduction.
2000-04-06	Dean Roddey	Fixed hanging bug in character transcoding.
2000-04-05	Dean Roddey	Enable installation of DTDHandler on SAX parser.
2000-04-04	Anupam Bagchi	Support for PTX platform
2000-04-03		IRIX 6.5 port
2000-03-30		COM wrappers
2000-03-24	Jeff Lewis	DOM_Document::GetElementsByTagId() added.
2000-03-23	Chih Hsiang Chou	DOM: support for identifying "ignorable white space" text nodes.
2000-03-23	Rahul Jain	URL Net Accessor added.

2000-03-20	Dean Roddey	Fix null pointer exception with some bad documents.
2000-03-17	Dean Roddey	Initial support for two-way transcoding.
2000-03-17	Dean Roddey	Intrinsic transcoding table generation utility added.
2000-03-17	Anupam Bagchi	UNIX build: Now generates object files in platform-specific directories
2000-03-13	Anupam Bagchi	Fix GCC build problem: Changed XML_GNUG to XML_GCC
2000-03-13	Helmut Eiken	Fixed #54. Changed self-assignment to now use the parameter value. Reported by Helmut Eiken <H.Eiken@cli.de>
2000-03-10	Chih Hsiang Chou	Fix bug # 19, add const keyword to API. As a result, update test case.
2000-03-10	Chih Hsiang Chou	DOM: "specified" flag of attributes now set correctly.
2000-03-08	Dean Roddey	Some fixes for content models that have multiple, trailing, empty PE refs (for content model extension.)
2000-03-07	Dean Roddey	First cut for additions to Win32 xcode. Based very loosely on a prototype from Eric Ulevik.
2000-03-03	Dean Roddey	Fixed a bug in SimpleContentModel that allowed an <a/> to be taken as valid for a content model of (a,b).
2000-03-02	Dean Roddey	Added a scanReset()/parseReset() method to the scanner and parsers, to allow for reset after early exit from a progressive parse. Added calls to new Terminate() call to all of the samples. Improved documentation in SAX and DOM parsers.
2000-03-02	Dean Roddey	Change "XML4C" to "Xerces" in many places Add a cleanup method to XMLPlatformUtils. Implement the Locator scheme for SAX. Add a -n option to most of the samples, to enable namespaces Fix an error where XMLScanner::parseNext() was falling through on an exception instead of return a failure. Implement the specialized string loading for Win98, since LoadStringW() doesn't work on 98 and makes the loaded error text from the Win32 message loader come out junk fix error when two trailing entity references in a content model, like so: <!ELEMENT foo (a|b|c|d|e %one;%two;)*>

Xerces C++ Version 1.1.0: Feb 28, 2000

2000/02/18	Dean Roddey	XMLCh defaults to wchar_t on platforms where wchar_t uses Unicode.
2000-02-18	Dean Roddey	Add Windows-1252 as a built in encoding
2000-02-17	Dean Roddey	Fixed an infinite loop caused while trying to trim leading white space from the raw URL during parsing.
2000-02-17	Rahul Jain	Add LibWWW based net accessor
2000-02-17	Chih Hsiang Chou	DOM: NodeIterator, TreeWalker added.
2000-02-16	Dean Roddey	Updates for EBCDIC code page issues.
2000-02-15	Chih Hsiang Chou	DOM: several namespace bugfixes
2000-02-14	Dean Roddey	Disallow EBCDIC documents without an encoding declaration
2000-02-10	Bill Schindler	Fixed defect in compare[N]IString function. Defect and fix reported by Bill Schindler from developer@bitranch.com
2000-02-10	Anupam Bagchi	Sample source code cleaned up.
2000-02-08	Dean Roddey	Fixed bug: xmlns:xxx="" should affect the mapping of the prefixes of sibling attributes
2000-02-07	Dean Roddey	Don't weave base and relative paths unless relative part is really relative.
2000-02-03	Dietrich Wolf	C++-Builder 4 support
2000-02-03	Robert Weir	DOMString enhancements
2000-01-31	Dean Roddey	Win32 mutex implementation was changed to use critical sections for speed.
2000-01-28	Dean Roddey	The API is not in place to allow client code to make sense of start/end entity ref calls from attribute values. So suppress them for now.
2000-01-28	Andy Heninger	Fix multi-threading problem in DOM.
2000-01-27	Dean Roddey	Fixed bug: If an entity ends on the last > of some markup, then the end of entity won't be sent because the end of entity is not sensed.
2000-01-24	Dean Roddey	Fixes a bogus error about]]> in char data.
2000-01-24	Dean Roddey	Exposed the APIs to get to the byte offset in the source XML buffer.
2000-01-21	Dean Roddey	Added a check for a broken pipe error on file read.
2000-01-18	Dean Roddey	Update to support new ICU 1.4 release
2000-01-18	Dean Roddey	Remove dependence on old utils standard streams
2000-01-18	Rahul Jain	Added CreateDOMDocument sample.
2000-01-13	Dean Roddey	Added a NetAccessorException for use by implementations of the NetAccessor abstraction, if they need to report errors during processing
2000-01-12	Dean Roddey	get the C++ and Java versions of error messages more into sync.
2000-01-11	Dean Roddey	Moved the input source classes from / to framework/.
2000-01-11	Dean Roddey	Changes to deal with multiply nested, relative paths, entities

Xerces C++ Version 1.0.1: December 15, 1999

- Port to Solaris.

- Improved error recovery and clarified error messages.
- Added DOMTest program.

Xerces C++ Parser Version 1.0.0: December 7, 1999

- Released Xerces C++ after incorporating ICU as a value-added plug-in.
- Has bug fixes, better conformance, better speed and cleaner internal architecture
- Three additional samples added: PParse, StdInParse and EnumVal
- Experimental DOM Level 2 support
- Support for namespaces
- Loadable message text enabling future translations to be easily plugged-in
- Pluggable validators
- Pluggable transcoders
- Reorganized the util directory to better manage different platforms and compilers

Xerces C++ BETA November 5, 1999

- Created initial code base derived from IBM's XML4C Version 2.0
- Modified documentation to reflect new name (Xerces-C)

13
Bug Reporting

How to report bugs

Please report bugs to Bugzilla [21] , the Apache bug database. Pick the product "Xerces-C++: Apache XML parsers in C++" using the following components:

Component	Description
DOM	Items specific to DOM
SAX/SAX2	Items specific to SAX or SAX2
Non-Validating Parser	General Parsing Problem
Validating Parser (DTD)	DTD related parser issue
Validating Parser (Schema)	Schema related parser issue
Utilities	Items related to utilities like MessageLoader, Transcoder, NetAccessors, Platform specific utilities
Build	Problem with build, makefile, project files
Documentation	Documentation bugs such as FAQ, Programming Guide
Samples/Tests	Samples or test cases related issues
Miscellaneous	Items not covered in other categories

A copy of your bug report is sent automatically to the discussion list Xerces-C mailing list [15] .

Search frist

Check the Bugzilla [21] database before submitting your bug report to avoid creating a duplicate report. Even the bug has been reported already, you may add a comment to the existing report since your contribution may lead to a quicker identification/resolution to the bug reported.

Write good bug report

Writing a useful bug report, which makes the bug reproducible, is the first step towards the resolution of the bug. Specifics about the bug, like

- Xerces C++ version number
- Platform
- Operating system and version number
- Compiler and version number
- The XML document (or excerpt) that failed
- The C++ application code that failed
- Whether you built the Xerces C++ library yourself or used the binary distribution
- What happened

are all necessary information to allow developer to reproduce, identify, evaluate and eventually, fix the bug, which is the very purpose of your reporting of the bug.

14
Feedback Procedures

Questions or Comments

Please browse through this bundled documentation completely. Most of the common questions have been answered in the FAQ's. Specifically, do read the answer to " Is there any kind of support available for Xerces C++?". Browsing this documentation, may be the quickest way to get an answer. Of course, if all else fails, as mentioned in the link above, you can post a question to the Xerces-C mailing list [15] .

See Bug Reporting if you would like to report a defect (greatly appreciated!).

Acknowledgements

Ever since this source code base was initially created, many people have helped to port the code to different platforms, and provided patches for both new features and bug fixes.

Listed below are some names (in alphabetical order) of people to whom we would like to give special thanks.

- Nadav Aharoni
- Curt Arnold
- Anupam Bagchi
- Torbjörn Bäckström
- Matthew Baker
- Devin Barnhart
- James Berry
- David Bertoni
- John Bellardo
- Arundhati Bhowmick
- Edward Bortner
- Sumit Chawla
- Chih Hsiang Chou
- Radovan Chytracek
- Hiram Clawson
- Todd Collins
- Michael Crawford
- Murray Cumming
- Helmut Eiken
- Mark Everline
- Simon Fell
- Paul Ferguson

- Pierpaolo Fumagalli
- Gary Gale
- Herny Gongaro
- Susan Hardenbrook
- Jeff Harrell
- Andy Heninger
- William L. Hopper
- Rahul Jain
- Tom Jordahl
- Martin Kalen
- Joe Kesselman
- Bob Kline
- Richard Ko
- Paul Kramer
- Volker Krause
- Arnaud LeHors
- Andy Levine
- Jeff Lewis
- Matt Lovett
- Sean MacRoibeaird
- Alberto Massari
- Jordan Naftolin
- Tinny Ng
- David Nickerson
- Khaled Noaman
- Michael Ottati
- Mike Pogue
- Joe Polastre
- John Ponzo
- Shengkai Qu
- Gareth Reakes
- Jim Reitz
- Dean Roddey
- John Roper
- Steven Rosenthal
- Erik Rydgren
- Bill Schindler
- Erik Schroeder
- Christian Schuhegger
- John Smirl
- Andrei Smirnov
- Gereon Steffens
- Rick J. Stevens
- Roman Sulzhyk
- Linda M. Swan
- Pieter Van-Dyck
- Curtis Walker
- Tom Watson

- Roger Webster
- Robert Weir
- Dietrich Wolf
- Kirk Wylie
- Peter A. Volchek
- Grace Yan
- PeiYong Zhang
- Henry Zongaro

15
Y2K Compliance

Apache Xerces Parser Year-2000 Readiness

Q: Are the Xerces parsers Year-2000-compliant?

Yes, Xerces-J and Xerces-C are Year 2000 compliant. They do not currently use any dates at all (at least until the XML Schema date datatypes are fully supported). However, you may still have Y2K problems if the underlying OS or Java implementation has problems with dates past year 2000 (e.g. OS calls which accept or return year numbers).

Most (UNIX) systems store dates internally as signed 32-bit integers which contain the number of seconds since 1st January 1970, so the magic boundary to worry about is the year 2038 and not 2000. But modern operating systems shouldn't cause any trouble at all.

The Apache Xerces project is an open-source software product of the Apache Software Foundation. The project and the Foundation cannot and does not offer legal assurances regarding any suitability of the software for your application. There are several commercial support organizations and derivative products available that may be able to certify the software and provide you with any assurances you may require (IBM's Websphere product is one of them).

The Apache HTTP server software is distributed with the following disclaimer, found in the software license:

```
    THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED
    WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
    OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED.  IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR
    ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
    SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
    LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
    USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
    ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
    OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
    OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
    SUCH DAMAGE.
```

16
PDF Documentation

PDF Documentation

You can get the entire Xerces C++ documentation in PDF format [39] for printing and offline reference.

> ***Note:*** *A word of caution! The tools to create the PDF documentation are still experimental. So the resulting PDF document is not perfect. We would be glad to receive your comments on the Xerces-C mailing list [15] .*

Appendix A
Links Reference

[1] http://www.w3.org/XML/

[2] http://www.w3.org/TR/REC-xml

[3] http://www.w3.org/TR/REC-DOM-Level-1/

[4] http://www.w3.org/TR/DOM-Level-2-Core/

[5] http://www.megginson.com/SAX/SAX1/index.html

[6] http://www.megginson.com/SAX/index.html

[7] http://www.w3.org/TR/REC-xml-names/

[8] http://www.w3.org/TR/xmlschema-0/

[9] http://www-4.ibm.com/software/ad/vacpp/

[10] http://www-4.ibm.com/software/ad/vacpp/service/csd.html

[11] http://oss.software.ibm.com/icu/

[12] http://www.gnu.org

[13] http://www.gnu.org/software/autoconf/autoconf.html

[14] http://www.gnu.org/software/make/make.html

[15] mailto:xerces-c-dev@xml.apache.org

[16] http://www.gnu.org/software/gcc/gcc.html

[17] mailto:rchgo400@us.ibm.com

[18] http://oss.software.ibm.com/developerworks/opensource/

[19] http://oss.software.ibm.com/icu/download/index.html

[20] http://archive.covalent.net

[21] http://nagoya.apache.org/bugzilla/

[22] http://www.stack.nl/~dimitri/doxygen/

[23] http://www.research.att.com/sw/tools/graphviz/

[24] http://www.w3.org/TR/xmlschema-1/

[25] http://www.w3.org/TR/xmlschema-2/

[26] http://www.x.org/terms.htm

[27] http://www.alphaworks.ibm.com/tech/xml4c

[28] http://xml.apache.org/xerces-c/index.html

[29] http://xml.apache.org/dist/xerces-c/

[30] http://xml.apache.org/

[31] http://xml.apache.org/cocoon/index.html

[32] http://www.gnu.org/software/tar/tar.html

[33] http://www.gnu.org/manual/tar/html_node/tar_117.html#SEC112

[34] http://sunsolve.sun.com

[35] http://www.w3.org/TR/REC-xml#charsets

[36] http://xml.apache.org/xerces-j/index.html

[37] http://www.oasis-open.org/cover/xml.html

[38] http://www.w3.org/TR/xhtml1/xhtml1.zip

[39] http://xml.apache.org/xerces-c/pdf/xerces-c.pdf