Error Parser Admin Manual

Daniel Prévost

Error Parser Admin Manual

by Daniel Prévost

Table of Contents

Chapter 1. Introduction

Error Parser is a small tool for application developers. It simplifies the management of error codes, error messages and documentation for both small and large projects.

This is accomplished by creating a master XML file containing all the necessary information on the errors (value, error message and documentation). Error Parser uses this file to generate the appropriate output files for different programming languages.

Control of the code generation is done by an option file written in XML. The format for both XML files is explained in a different document.

Chapter 2. New in this release

This release is a minor update; support for Java was added.

There are no known bugs at this time and no bugs were corrected for this release.

Chapter 3. Installation

The installation procedure of the program itself is relatively simple if you use the pre-packaged version of the software (RPM, Inno setup, etc.) or if you build the software from scratch and use **make install**.

This section will look into some potential installation issues in some details.

3.1. The xml catalog on Linux/Unix

The two DTD (Document Type Definition) for Error Parser can be installed in the global xml catalog (usually /etc/xml/catalog). Whether this is needed depends on your exact situation.

A DTD can be identified using a PUBLIC clause at the start of an the xml file). For example:

```
<!DOCTYPE errorlist PUBLIC
   "-//Error Parser project//DTD Error Parser XML V1.3//EN"
    "http://errorparser.sourceforge.net/xml/errorParser13.dtd">
```

In this case the DTD will be downloaded from the Error Parser web site.

Alternatively, the DTD can be identified with a SYSTEM clause:

```
<!DOCTYPE errorlist SYSTEM
   "/usr/local/share/xml/errorParser/errorParser13.dtd">
```

Using SYSTEM is perfectly correct in most cases. The main advantage of the PUBLIC clause is to make sure that multiple developers use the same DTD and not a locally modified version of it.

If you decide to use the PUBLIC clause and do not want to download the DTD on every invocation of Error Parser, adding the Error Parser catalog to your main catalog is the way to go. The program xmlcatalog, included in the libxml2 package, will simplify this task. Furthermore a bash script is provided by Error Parser. This script is installed in /usr/local/share/xml/errorParser/install_catalog.sh if Error Parser is installed in /usr/local. You'll need to edit this script to tailor it for your installation.

Recommendation: use the SYSTEM clause.

3.2. The xml catalog on Windows

The previous section already discussed the use of PUBLIC or SYSTEM clause for identifying a DTD. The biggest difference between Microsoft Windows and Linux, for example, is the absence of a centralized method to manage xml DTD and Schemas - each software manages its XML on its own. It makes even more sense to use SYSTEM here.

If you still want to install an xml catalog on Windows, the only thing you need to know is that libxml2 uses the environment variable XML_CATALOG_FILES to find the main xml catalog.

There is an example of a main catalog in the DTD directory (main_catalog_win32.xml). If the main catalog already exists you can use the script DTD\install_catalog.vbs. In both cases you will need to edit the files to tailor them to your installation.

Chapter 4. Building the Software

The only prerequisite software needed to build and run Error Parser is the library libxml2 (the Gnome project xml parser). If it is not installed on your system, you can download it at xmlsoft.org (http://xmlsoft.org/downloads.html).

You can find binaries for libxml2 (built for Microsoft Windows) at this location (http://www.zlatkovic.com/libxml.en.html).

To build the documentation, additional software is required as explained in the following sections.

4.1. Building on Linux/Unix from the repository

The latest version of the Error Parser software can be downloaded by using git. Note: git is a very fast distributed version control system created for the Linux kernel. Visit git home page (http://git.or.cz/index.html) for more information on this open-source software.

Error Parser uses three external git repositories. They are all kept in sync with each other. To retrieve the latest version from github.com, use the following command:

git-clone git@github.com:dprevost/errorparser.git my_repo

or use the following commands to get it from gitorious.org or sourceforge.net:

git-clone git@gitorious.org:errorparser/mainline.git my_repo

git-clone git://errorparser.git.sourceforge.net/gitroot/errorparser my_repo

Warning

Be careful as the code in the repository is not always stable (hint: unstable commits will usually have the string "work in progress" included in the commit message).

The next step is to run the shell script autogen.sh (in the trunk directory). This will generate everything you need (the makefiles, the configure script). The script will also run **configure** for you but you might want to run it again if you wish to change its default options.

The remaining instructions are identical to the instructions for building the software from a tar file.

4.2. Building on Linux/Unix from the tar file

Step by step instructions to build the package from a tar file:

- Run configure (./configure). Here is a list of options that you might want to use to tailor the package to your needs (you can use ./configure --help to see all the options):
  - --prefix=PREFIX

    To change the default installation directory (default is /usr/local). Example: --prefix=/usr.

  - --mandir=DIR

    To change the default installation for man documentation (default is [PREFIX/man]). Example --mandir=/usr/localshare/man.

- make
- make check (to run the test suite - optional)
- make install

4.3. Building the documentation on Linux/Unix

The documentation is provided as part of the source package. If you are only installing and using this software as is, without modifying it, you can skip this section.

The documentation is written in DocBook xml. The source for the current document is doc/manual.xml. The file parser/errorParser.xml is used to generate the man page.

To create or refresh the man page, you will need to install the DocBook DTD itself and the docbook2x package.

Warning

The program docbook2x-man is named db2x_docbook2man on some systems, for example on Fedora 9.

If docbook2x-man is present on the system, **autoconf** (./configure) will detect it and **make** will refresh the documentation if the source file was modified.

The html and pdf documentation are updated in similar ways. To update the pdf manual, **db2pdf** must be present on your system. For both html and pdf, you will need a program to transform the xml. The makefile is currently written to use **xsltproc**.

Note: the Microsoft specific chm file (used for Microsoft help files) can only be updated on Windows.

4.4. Building on Windows from the repository

If you want to retrieve the software directly from the repository, you will need a subversion client for Windows. Tortoise (http://tortoisesvn.tigris.org/) is one possible choice. The repository is:

```
https://errorparser.svn.sourceforge.net/svnroot/errorparser/trunk
```

Warning

Be careful as the code in the repository is not always stable (hint: unstable commits will usually have the string "work in progress" included in the commit message).

The remaining instructions are similar to the instructions for building the software from a zip file.

4.5. Building on Windows from a zip file

To build the software, simply use the solution file parser\parser.vs2005.snl for VC++ v8, known as VS 2005. If you use VC++ v9 (VS 2008) instead, use parser\parser.vs2008.snl.

In addition, you will need to tell Visual Studio the location of the header files and libraries of the libxml2 package. Select the Options window under the Tools menu to add these 2 locations to the default lists used by VS (exact location: Projects and Solutions\VC++ Directories).

To run the tests, from a command shell run **cscript RunTests.vbs** (in the tests directory). You will need the program Gnu diff (http://gnuwin32.sourceforge.net/packages/diffutils.htm) for Windows. A copy of it is in the tests directory.

You can also build the software and run the tests from the command line by using nmake, the Microsoft version of make:

```
nmake -f Makefile.vs2005 target
```

or

```
nmake -f Makefile.vs2008 target
```

The different targets available in Makefile.vs2005(8):

- all (the default target)

  Builds Error Parser.

- check

  Runs the test suite.

- checkdebug

  Runs the test suite using the debug version of Error Parser.

- docs

  Builds the documentation.

- clean

  Removes the executable, the object files and all intermediate files.

You might need to modify parser/Makefile.vs2005(8) if you install libxml2 in a non-standard directory; the hardcoded path for libxml2 is "C:\Program Files\libxml2".

To build the installation package itself, you will need the program Inno Setup (http://www.jrsoftware.org/isinfo.php). The two Inno Setup files are located in the installation folder under the names errorParser.vs2005.iss and errorParser.vs2008.iss.

4.6. Building the documentation on Windows

The documentation is provided as part of the source package. If you are only installing and using this software as is, without modifying it, you can skip this section.

The documentation is written in DocBook xml. The source for the current document is doc\admin.xml. The user's manual is in doc\users.xml. The file parser/errorParser.xml is used to generate the man page.

As mentioned earlier, the documentation can be updated using "nmake -f Makefile.vs2005(8) docs". This will update the two Microsoft help-file (chm) version of the manuals and the html version. The PDF version and the man page require a Unix/Linux environment.

To create the documentation, you will need the DocBook DTD itself, xsltproc and the Microsoft HTML Help Compiler (hhc.exe) which you can get from the HTML Help Workshop from Microsoft.