NAME

mono-shlib-cop − Shared Library Usage Checker

SYNOPSIS

mono-shlib-cop [OPTIONS]* [ASSEMBLY-FILE-NAME]*

OPTIONS

−p, --prefixes=PREFIX

Mono installation prefixes. This is to find $prefix/etc/mono/config. The default is based upon the location of mscorlib.dll, and is normally correct.

DESCRIPTION

mono-shlib-cop is a tool that inspects a managed assembly looking for erroneous or suspecious usage of shared libraries.

The tool takes one or more assembly filenames, and inspects each assembly specified.

The errors checked for include:

* Does the shared library exist?

* Does the requested symbol exist within the shared library?

The warnings checked for include:

* Is the target shared library a versioned library? (Relevant only on Unix systems, not Mac OS X or Windows.)

In general, only versioned libraries such as *libc.so.6* are present on the user's machine, and efforts to load *libc.so* will result in a **System.DllNotFoundException.** There are three solutions to this:

1. Require that the user install any *-devel* packages which provide the unversioned library. This usually requires that the user install a large number of additional packages, complicating the installation process.

2. Use a fully versioned name in your **DllImport** statements. This requires editing your source code and recompiling whenever you need to target a different version of the shared library.

3. Provide an *assembly.config* file which contains <dllmap/> elements to remap the shared library name used by your assembly to the actual versioned shared library present on the users system. Mono provides a number of pre-existing <dllmap/> entries, including ones for *libc.so* and *libX11.so.*

EXAMPLE

The following code contains examples of the above errors and warnings:

```
using System.Runtime.InteropServices; // for DllImport
class Demo {
        [DllImport ("bad-library-name")]
        private static extern void BadLibraryName ();

        [DllImport ("libc.so")]
        private static extern void BadSymbolName ();

        [DllImport ("libcap.so")]
        private static extern int cap_clear (IntPtr cap_p);
}
```

Bad library name

Assuming that the library *bad-library-name* doesn't exist on your machine, *Demo.BadLibrary-Name* will generate an error, as it requires a shared library which cannot be loaded. This may be ignorable; see **BUGS**

Bad symbol name

Demo.BadSymbolName will generate an error, as *libc.so* (remapped to *libc.so.6* by mono's *$prefix/etc/mono/config* file) doesn't contain the function *BadSymbolName*

Unversioned library dependency

Assuming you have the file *libcap.so* , *Demo.cap_clear* will generate a warning because, while *libcap.so* could be loaded, *libcap.so* might not exist on the users machine (on FC2, */lib/libcap.so* is provided by *libcap-devel* , and you can't assume that end users will have any *-devel* packages installed).

FIXING CODE

The fix depends on the warning or error:

Bad library names

Use a valid library name in the **DllImport** attribute, or provide a <dllmap/> entry to map your existing library name to a valid library name.

Bad symbol names

Reference a symbol that actually exists in the target library.

Unversioned library dependency

Provide a <dllmap/> entry to reference a properly versioned library, or ignore the warning (see **BUGS**).

DLLMAP ENTRIES

Mono looks for an *ASSEMBLY-NAME* mapping information. For example, with *mcs.exe* , Mono would read *mcs.exe.config* , and for *Mono.Posix.dll* , Mono would read *Mono.Posix.dll.config*

The *.config* file is an XML document containing a top-level <configuration/> section with nested <dllmap/> entries, which contains **dll** and **target** attributes. The dll attribute should contain the same string used in your **DllImport** attribute value, and the target attribute specifies which shared library mono should *actually* load at runtime.

A sample .config file is:

```
<configuration>
        <dllmap dll="gtkembedmoz" target="libgtkembedmoz.so" />
</configuration>
```

BUGS

* Only **DllImport** entries are checked; the surrounding IL is ignored. Consequently, if a runtime check is performed to choose which shared library to invoke, an error will be reported even though the specified library is never used. Consider this code:

```
using System.Runtime.InteropServices; // for DllImport
class Beep {
        [DllImport ("kernel32.dll")]
        private static extern int Beep (int dwFreq, int dwDuration);

        [DllImport ("libcurses.so")]
        private static extern int beep ();

        public static void Beep ()
        {
                if (System.IO.Path.DirectorySeparatorChar == '\\') {
                        Beep (750, 300);
                }
                else {
                        beep ();
                }
        }
}
```

If *mono-shlib-cop* is run on this assembly, an error will be reported for using *kernel32.dll* , even though *kernel32.dll* will never be used on Unix platforms.

* *mono-shlib-cop* currently only examines the shared library file extension to determine if a warning should be generated. A *.so* extension will always generate a warning, even if the *.so* is not a symlink, isn't provided in a *-devel* package, and there is no versioned shared library (possible examples including */usr/lib/libtcl8.4.so, /usr/lib/libubsec.so,* etc.).

  Consequently, warnings for any such libraries are useless, and incorrect.

  Windows and Mac OS X will never generate warnings, as these platforms use different shared library extensions.

MAILING LISTS

Visit http://lists.ximian.com/mailman/listinfo/mono-devel-list for details.

WEB SITE

Visit http://www.mono-project.com for details