**DFM: Build Problems FAQ**

**Prepared By : Sahu & Pradeep**

**===========================================**

**1. General Information**

**1.1 Where is the source code?**

**The main perforce depot is**

**//depot/prod/champagne/main/...**

**Near the end of each release, we create a branch for the release so that development can begin in "main" on the next release. The branches have names like**

**//depot/prod/champagne/Rchampagne2.1x/...**

**The most reliable way to create a client view is with a p4 command like**

**p4 -c your-name:dfm:main newclient -v champagne:main**

**DFM depends on many shared components, which you pull into your tree with this command from the top of the DFM source tree:**

**perl sync\_shared**

**1.2 What is sync\_shared?**

**The sync\_shared script uses a configuration file, SharedComponents.cfg, to define which portions of which other perforce depots should be copied into the DFM area for building.**

**DFM uses portions of the NetCache 5.3 tree, portions of zephyr, licensing components from netapplm, and so on.**

**If one of the shared components changes in a way that impacts DFM, or if DFM changes to use some newer interface in a shared component, you must run**

**perl sync\_shared**

**again to get the latest versions of the components.**

**1.3 sync\_shared didn't work!**

**If you've run sync\_shared, but some components are still out of date, you may need to forcefully synchronize the shared components. Use the -f option as below:**

**perl sync\_shared -f**

**In extreme cases, you may even need to completely remove the shared components first:**

**perl clean\_shared**

**perl sync\_shared**

**2. UNIX Problems**

**2.1 Preparing to build**

**On Solaris, the library building program ar is in /usr/ccs/bin, which may not be on your PATH. Add it with**

**$ PATH=:/usr/ccs/bin**

**$ export PATH**

**2.2 Build is looping generating .p files**

**Loops like this**

**Making all in config**

**make[2]: Entering directory `/x/yyy/src/plugins/config'**

**Generating obj/solaris/config.p ...**

**Generating obj/solaris/main.p ...**

**make[2]: Leaving directory `/x/yyy/src/plugins/config'**

**make[2]: Entering directory `/x/yyy/src/plugins/config'**

**Generating obj/solaris/config.p ...**

**Generating obj/solaris/main.p ...**

**make[2]: Leaving directory `/x/yyy/src/plugins/config'**

**make[2]: Entering directory `/x/yyy/src/plugins/config'**

**Generating obj/solaris/config.p ...**

**Generating obj/solaris/main.p ...**

**make[2]: Leaving directory `/x/yyy/src/plugins/config'**

**make[2]: Entering directory `/x/yyy/src/plugins/config'**

**Generating obj/solaris/config.p ...**

**Generating obj/solaris/main.p ...**

**are usually caused by .c files that reference a header file that isn't found in the list of include directories.**

**Sometimes this is a typo in the .c file; sometimes it is a missing header file.**

**Some hints:**

**a. Try to figure out which header file or files are having the problem. Do this by eye-balling the source, or running a tool like**

**b. ~hackwort/usr/bin/chkhdr \*.c**

**to try to identify which headers are missing.**

**c. Fix obvious typos if you notice any.**

**d. If the headers are supposed to be supplied via one of the shared components, try**

**e. perl sync\_shared**

**(possibly doing "clean\_shared" first or using -f).**

**2.3 How to enable Solaris/Linux nightly builds for different vendors?**

**The depot path "//depot/user/build/Nightly/..." contains default config file for each branch with a naming convention of Nightly\_<branch\_name>.config. This file is used by the nightly build daemon. It builds code for all vendors.**

**Default config files:**

**//depot/user/build/Nightly/Nightly\_Rchampagne.config**

**//depot/user/build/Nightly/Nightly\_champagne-freyareloaded.config**

**Nightly\_champagne-freyareloaded.config is used by the nightly for building champagne-freyareloaded branch and Nightly\_Rchampagne.config is used by nightly for building champagne/main branch.**

**If you want to skip building for all non-NetApp vendors, you can use another config file named Nightly\_<branch\_name>NetappOnly.config.**

**Config file to build Netapp version only:**

**//depot/user/build/Nightly/Nightly\_RchampagneNetappOnly.config**

**3. Windows Problems**

**3.1 Preparing to build**

**To build DFM for Windows, you'll need Microsoft's Visual C++ installed, plus some additional components:**

**• MS Windows Platform SDK, July 2002**

**• ActiveState Perl, 5.6.1 build 633**

**You can install the required Windows SDK by running \ayr\users\ctbuild\MS\_SDK\_July\_2002\Setup.exe from your DOS prompt. Install all the SDK components.**

**You'll need to customize VC++ to point it to the libraries and header files from the Platform SDK, Use Tools -> Options -> Directories, and specify**

**• Include Directories, in this order:**

**• drive:\Program Files\Microsoft SDK\include**

**• drive:\Program Files\Microsoft Visual Studio\VC98\include**

**• drive:\Program Files\Microsoft Visual Studio\VC98\MFC\include**

**• drive:\Program Files\Microsoft Visual Studio\VC98\ATL\include**

**• Library Directories, in this order:**

**• drive:\Program Files\Microsoft SDK\lib**

**• drive:\Program Files\Microsoft Visual Studio\VC98\lib**

**• drive:\Program Files\Microsoft Visual Studio\VC98\MFC\lib**

**• Executable Files. in this order:**

**• drive:\Program Files\Microsoft SDK\bin\winnt**

**• drive:\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin**

**• ...**

**3.2 Where can I find configuration used by build server for nightlies?**

**In the hitchhiker's guide, here. DFM uses Configuration D.**

**3.3 VC++ won't even load my project**

**The Windows build tools are very picky about the line terminators used on the the project (.dsw and .dsp) files.**

**To convert from the UNIX-style terminators -- the style we use for checkins to perforce -- to the Windows-style terminators, run**

**dsw2dos**

**from the "src" directory.**

**Remember to convert the files back to UNIX style terminators before checking in any changes:**

**dsw2unix**

**3.4 VC++ can't find .d files**

**DFM uses .d files for forward declarations of static functions within .c files.**

**The .d files are generated by the ctod script. This is done automatically on UNIX, but requires explicit rules in the .dsp files on Windows.**

**If you see an error like**

**h:\scratch\dfm-sv\src\cmd\report.c(155) : fatal error C1083:**

**Cannot open include file: 'report.d': No such file or directory**

**it probably means you need to add some custom build steps to the .dsp file for the project.**

**The rules to generate .d files from .c files look like**

**# Begin Source File**

**SOURCE=.\report.d**

**!IF "$(CFG)" == "cmd - Win32 Release"**

**# PROP Ignore\_Default\_Tool 1**

**USERDEP\_\_REPOR="$(InputName).c"**

**# Begin Custom Build - Running ctod on $(InputName).c**

**InputPath=.\report.d**

**InputName=report**

**"./$(InputName).d" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"**

**perl ../util/ctod $(InputName).c**

**# End Custom Build**

**!ELSEIF "$(CFG)" == "cmd - Win32 Debug"**

**# PROP Ignore\_Default\_Tool 1**

**USERDEP\_\_REPOR="$(InputName).c"**

**# Begin Custom Build - Running ctod on $(InputName).c**

**InputPath=.\report.d**

**InputName=report**

**"./$(InputName).d" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"**

**perl ../util/ctod $(InputName).c**

**# End Custom Build**

**!ENDIF**

**# End Source File**

**3.5 How to enable windows nightly builds for different vendors?**

**At the root of the perforce client for a champagne branch you can find Makeaw.pl file. This file is used by windows nightly build. It builds code for all vendors by default.**

**If you want to skip building code for all vendors other than NetApp, you can pass netapponly argument to Makeaw.pl.**

**# when invoking directly**

**C:\> perl Makeaw.pl netapponly**

**# when invoking via winbuild**

**% winbuild -run Makeaw.pl netapponly**

**4. Windows Build Server**

**4.1 I don't have a Windows box to build DFM. What options do I have?**

**You have two options:**

**a. Find a PC and build your own. See the answer on Preparing to build above for details. Depending on your familiarity with Windows, how long it takes to procure a machine, and how much you need to build from scratch, etc., it can take anywhere from a day to a week to build such a beast.**

**b. Obtain access to the Windows build server maintained joinly by the DFM dev team and Internal Test. It comes pre-installed with all the software required to build DFM. See the next section for more details.**

**4.2 How do I access the Windows build server?**

**The build server runs the Windows terminal service. This means you can log in remotely from your Windows, Linux, or Mac desktop machines using a terminal server client and use Visual C++ as if you're sitting in front of the build server.**

**Follow these steps to get access to the build server:**

**• Obtain your own licensed copy of Visual C++ 6.0. Although the compiler is already installed on the build server, you do need your own license in order to use the software legally. IT can help procure a copy.**

**Note: Microsoft has discontinued Visual C++ 6.0. However, you can order the latest Visual C++/.NET/whatever and get a license to use Visual C++ 6.0.**

**• Download the Terminal Server Client software for your system. The Windows version can be found here:**

**• /u/slam/software/win32/TerminalServiceClients/win32/**

**•**

**• or**

**•**

**• \\vfusers4\users\slam\software\win32\TerminalServiceClients\win32\setup.exe**

**The client for Mac OS X is called Remote Desktop Connection. The Linux client is called rdesktop.**

**• To obtain access to the server, send an email to slam@netapp.com stating that you have obtained a license for Visual C++. You'll be granted access within a few hours.**

**• The server is bldw1.eng.netapp.com.**

**4.3 Okay, I go access. How do I start building?**

**Read the section on Building for Windows From a UNIX Perforce Workspace in the Windows Development Guide.**

**4.4 What are the limitations?**

**• You must create and maintain your perforce tree from a UNIX compute server. p4win is not installed on the build server. To many people, this setup is a good thing. It allows you to build Windows, Solaris and Linux versions of DFM from the same perforce tree.**

**• This is a shared machine. You do not have administrator privilege to install any software globally.**

**5. Maintaining Zephyr/DFM Shared Components**

**5.1 How to pull zephyr changes into DFM**

**DFM pulls components from many other places using the sync\_shared script (which is driven from the SharedComponents.cfg file). Some of these components are source code that are copied into the DFM tree and compiled; other components are headers and libraries that are pulled into the DFM tree and used to compile and link against.**

**The zephyr components include some of each: some source code files (the libadt and libxml bits) and some libraries (everything else: libnetapp, libsnmp, pthread, ntapadmin, libndmp).**

**Code in zephyr is destined eventually for delivery to our partners in an SDK, so it should not have anything DFM-specific in it.**

**The procedure for getting zephyr changes incorporated into DFM is this:**

**a. Code, test, and checkin the changes to the zephyr tree; use COMPILE.PL on Windows (which copies headers into the include trees). This script requires perl5.6.1, which may be downloaded from http://www.activestate.com/. Invoke as:**

**b. C:\...> PERL5.6.1 COMPILE.PL**

**c. Update the adt, netapp, and xml entries in SharedComponents.cfg of the DFM tree to reflect your submission into the zephyr depot.**

**d. Make sure that the release/\*/include and release/\*/lib directories in the zephyr build tree accurately reflect what headers and libraries should be exported**

**e. After an official build of zephyr (either a nightly or a Special Build) completes, copy the "release" area of the zephyr build tree into the NetAppAdmin area of the depot.**

**I keep a //depot/prod/shared\_components/NetAppAdmin/... p4 client hanging around; when necessary, I go to that client and do**

**$ p4 sync**

**$ ./p4dircmp -n zephyrN\_021010\_1400 > out.txt**

**$ grep delete out.txt**

**(replacing the zephyr build string with the one I want to incorporate). The -n does a dry run to show what the script plans to do. Checking for unintended deletions ensures you're not attempting to submit changes before the build system completes (for example, if you're doing this immediately after submitting a special build).**

**When you are ready to submit your changes, do**

**$ p4 sync**

**$ ./p4dircmp -s zephyrN\_021010\_1400**

**The -s makes the script call "p4 submit" to publish the changes into p4.**

**Note that the command-line argument is "zephyrN\_021010\_1400" and not "/x/eng/rlse/zephyr/zephyrN\_021010\_1400" (it makes a difference, for other uses of p4dircmp).**

**f. Update the zephyr entry in SharedComponents.cfg of the DFM tree to reflect your submission into the NetAppAdmin depot.**

**g. Issuing "make" in a champagne/main tree will invoke "sync\_shared", which will now pick up your new zephyr code.**

**h. Submit your champagne/main changes.**

**5.2 How do I fix linux build error, undefined reference to `\_\_ctype\_b\_loc'?**

**Some of the linux hosts selected through althost mechanism have newer version of glibc. The files compiled on these hosts may not link on other linux hosts. You can fix the build by simply doing "make linux3-all" which selects only newer linux hosts.**

**See burt93069 for more information.**