Interoffice Memo

Date: 12/13/1999

To: NT Base Development Group

From: Mark Lucovsky

RE: Getting Setup on Source Depot

1 Introduction

The purpose of this document is to explain to all of you how you should go about setting up your development machines for post Windows 2000 work.

As you all know, we have migrated to a new source code control system based on Source Depot. Hopefully, you have all been to a talk on source depot, have played with the test depots, have your mouse pads, and know that http://sourcedepot is a good source of information. Now it is time for all of you to get started.

Our team's virtual build lab is operational and is able to release x86 free, x86 checked, ia64 free, and ia64 checked. We will begin doing axp64 checked shortly (need power which arrives on 12/18).

2 Build Servers and Releases

Our build servers are:

Machine Name	Architecture	Debug Type
Robsvbl1	X86	Free
Robsvbl2	X86	Checked
Robsvbl3	Ia64	Free
Robsvbl4	Ia64	Checked
Robsvbl5	Axp64	Checked

When we do builds on these servers, the resulting build is released to \\<machinename>\latest. Archived builds are located on \\<machinename>\release. In the next few days, we will have automatic mail sent whenever a build is "released". Builds are released in their untested, raw form, and the builds are released on their own schedule. For example, if we start all five machines building at the same time, the x86 machines will release in a little over 5 hours, the ia64 machines in about 8 hours, and the axp64 machine in about 10 hours. If some machines complete the build, and others fail, only builds from the successful machines will be released.

If errors occur during a build, the "NT Base Development Group" email alias receives build break email. You are all expected to respond to this mail and fix your build breaks in real time.

3 Configuring Your Dev Machines

If you plan on working on your source code from home over RAS, you will want to install Windows 2000 Server and configure in Terminal Server in Remote Administration Mode. There are many Source Depot operations that like to start your editor, so running source depot in a remote.exe cmd.exe window will not do. If you do not plan on working from home, Windows 2000 Professional will be fine.

If you build the Admin Depot, which contains the source code for setup, you will want to add your ntdev account to the local administrators group on your machine. This requirement comes from the way we build the hive files.

4 Enlisting

With your machine configured, you are ready to begin the enlist process. As you know from our presentations on Source Depot, you will be enlisting in a team level branch. This means that your changes are not visible to other teams until our team integrates our changes into the main branch. Other teams changes are similarly invisible to us. We have not decided how often we will integrate our changes into the main code line, or how often we will pick up changes from other teams. We will decide that in the near future.

We have simplified the amount of knowledge you need to have about the branch structure by having you enlist using enlistment profiles.

4.1 Enlist Profiles

The scripts used to enlist are located on \\glacier\sdx. If you browse this folder, you will find the \\glacier\sdx\profiles directory. This directory contains enlistment profile files that package of the enlistment options. The enlistment profiles that affect you are:

- RobS.N.All
- RobS.N.Base
- RobS.N.Drivers
- RobS.N.SetupTeam

We think these four profiles cover nearly everyone in our group.

4.1.1 The .All Profile

The .All profile is for those of you accustomed to building "the whole system". If you received a Dell 6300 from me this past year, this is you. This profile enlists you in every single depot. A clean build of this system produces an installable build in all flavors (dtc, srv, ent, wks).

4.1.2 The .Base Profile

The .Base profile enlists you into just the Base and Root depots. This will let you build the kernel, hals, kernel32, filesystems, etc. This profile easily enlists on a nice laptop and builds clean on a Toshiba 7020 in a little over 2 hours.

4.1.3 The .Drivers Profile

The .Drivers profile enlists you into just the Drivers and Root depots. If you work for Bobri, this is probably the default profile for you.

4.1.4 The .SetupTeam Profile

The .SetupTeam profile enlists you in the Admin, Base, and Root depots.

4.2 No Enlist Profile for Me

Enlist profiles are very simple files. We took a stab at creating enlistment profiles that would work well for large groups of people in a variety of teams. If there isn't a profile for you, don't worry, there are two options. A new profile can easily be created that enlists your group in the depots that you want to be enlisted in. If this is too much hassle, adding a depot to your enlistment is easily done after you have enlisted using one of the existing profiles.

4.3 Enlisting Your Machines

The new tree layout will allow you to name your root enlistment directory anything. Please be aware that virtually all of our development has been done using a single directory at the root of a drive (*e.g. z:\nt, z:\newnt, etc.*). We have not tested deeper directory roots so we advise against enlisting deeper than a single directory off of the root.

Assuming you are using the drive Z: convention, we suggest that your enlistment directory be z:\nt. To enlist in your branch in this location perform the following steps.

- Z:
- Mkdir \nt
- Set SDXROOT=z:\nt
- \\glacier\sdx\sdx enlist -p \\glacier\sdx\profiles\<yourprofile> or:
    - \\glacier\sdx\sdx enlist -p \\glacier\sdx\profiles\Robs.All
    - \\glacier\sdx\sdx enlist -p \\glacier\sdx\profiles\Robs.Base

- \\glacier\sdx\sdx enlist -p \\glacier\sdx\profiles\Robs.Drivers
- \\glacier\sdx\sdx enlist -p \\glacier\sdx\profiles\Robs.SetupTeam

This stage of the enlistment process takes a few minutes to complete. After starting this command, you will see a screen with the names of the depots in which you are enlisting. Take a look at this and make sure the enlist command is enlisting you in the depots you expect to enlist in. This process will prepare the z:\nt directory for syncing the source base, and will bootstrap your enlistment by syncing the tools directory of the root depot. This sync is needed so that you can create your first "Razzle" window and complete your enlistment with a full sync.

4.3.1 Creating Your Razzle Window

All of your command line operations done on your VBL machines are done in "Razzle" command windows. These windows are very simple to create and contain all of the environment variables needed by our build environment. Your "Razzle" windows are created by placing a shortcut on your desktop whose target command line is:

Cmd /k z:\nt\tools\razzle.cmd [arguments]

You should also set the "Start in" property to z:\nt.

For more information on the arguments, use razzle.cmd /?.

```
d:\nt>razzle /?

Usage: razzle <arguments>

where <arguments> can be one or more of:
        verbose - Enable verbose execution of this script
          Win64 - Build for Win64 target
                   (if on x86, build for ia64, if on alpha,
                    build for axp64)
           free - Build free bits (default is to build checked)
      ChkKernel - Checked kernel/hal/ntdll (use with 'free')
         no_opt - disable compiler optimizations
    no_binaries - disable creation of the
                  binaries.<arch><buildtype> dir.
   binaries_dir - must specify <basepath> immediately after this switch.
                  Use specified path as binplace dir instead of default.
                  Default value is "binaries".
       no_title - don't set the default window title bar.
       no_batch - don't use nmake batch feature
                  (compile one file at a time rather than all at once)
   no_certcheck - don't ensure the testroot certificate is installed.
restricted_path - set path to the same restricted path that build uses
                  (for debugging "why can't build find my
                  tool" problems)
  OfficialBuild - Build as one of the official build machines for
                  this branch.
                  The machine name must be identified in
                  d:\nt\tools\BuildMachines.txt.
   PrimaryBuild - Build as the primary build machine for this branch.
                  Similar to OfficialBuild except this machine will
                  be the one to actually checkin most of the published
                  headers and libraries.  Typically, this is a
                  x86fre build.
```

- Razzle windows default to producing checked builds. If you want to build free, specify free as the Razzle.cmd argument.
- Go ahead and create your Razzle window and then launch that window.

4.3.2 Completing Your Enlist

With your razzle windows configured, you are ready to complete your enlistment by performing a full sync to your branch.

Launch one of your razzle windows, and from z:\nt issue the following command:

sdx sync

This command will go to each depot you are enlisted in and sync all files in the depot. Since this is a full sync on an initial enlistment, you can expect this operation to take some amount of time. For a .All enlistment, the enlist can take from 2-4 hours. For a simple enlistment like Robs.Base, the enlist process should take about twenty minutes.

4.3.3 Adding a Depot

If one of the standard profiles didn't contain the exact depot set you were looking for, depots are easily added by doing:

- cd /d %sdxroot%
- sdx enlist <depotname>

Suppose your initial enlist was done using Robs.Base. To add an enlistment of the Drivers depot, and to fully sync all of your depots (including drivers) do:

- cd /d %sdxroot%
- sdx enlist Drivers
- sdx sync

The complete list of depots follows. Note that as of 12/15/1999, the depot name is compared case sensitive. Use the exact spelling and capitalization from the below table.

- Admin
- Base
- COM
- Drivers
- DS
- EndUser
- InetCore
- InetSrv
- MultiMedia
- Net
- PrintScan
- Root
- SdkTools
- Shell
- TermSrv
- Windows

SDX will always enlist you in the Root depot, so you won't need to enlist in that depot manually.

To defect from a depot, use sdx defect <depotname> -f.

Source depot does not support the notion of ghosting, so don't ask about this, and don't try to do this on your own. You are expected to be fully enlisted in your depots, and you are expected to fully sync and fully clean build your depots.

4.4 Enlist Summary

In summary, enlisting your machine involves the following:

- Determine which branch you are to enlist in and locate the appropriate enlist profile on \\glacier\sdx\profiles
- Z:
- Mkdir \nt
- Set SDXROOT=z:\nt
- \\glacier\sdx\sdx enlist -p \\glacier\sdx\profiles\<yourprofile>
- Create a desktop shortcut with cmd /k z:\nt\tools\razzle.cmd [arguments] and "start in" set to z:\nt
- Start a razzle window using the shortcut from the previous step
- Sdx sync

- Wait

5 Your First Build

Once you are enlisted, you are ready to begin doing your builds. One note of caution... Do not store ANYTHING that is not in source control under %sdxroot%. During the clean build process, all files not in source control are deleted from your source tree helping to ensure that your build is successful. If you forget to sd add a file and do a clean build, that file will be deleted. Hopefully no one will make this mistake, or if they do, it will only happen once.

5.1 Building Via Command Line

Building from the command line is highly automated. A single command script will:

- clean sync your machine
- clean build the system
- post process your build into an installable release tree

In order to perform these operations, execute the following command from within a razzle window, whose current directory is % sdxroot%.

- perl tools\timebuild.pl

After completion of your build, %sdxroot%\build.time will contain a running log of build times and status. If your build was successful, your build is automatically released. If your build failed, you will have to manually fix any build breaks, and resume the build process As of 12/14/1999, recovery after correcting your build errors is done by simply executing “postbuild.cmd”. In the very near future, recovery will be done by using the command “perl tools\timebuild.cmd –resume”.

5.2 Installing your Build

If your dev machine is a 4 way (or better) SMP system, your binary tree is an exact replica of the CD, with full compression and CD layout. To install this build use the following command:

\\<machine>\latest\{srv,wks,ent,dtc}\setup

If your dev machine is a not a 4 way SMP system, you do not have the horsepower to efficiently compress the binaries, so you will install using the following command:

\\<machine>\latest\{srv,wks,ent,dtc}\{i386,ia64,axp64}\winnt32\winnt32

You probably noticed that there is no /M switch to pull binaries that you didn’t build from “the build lab”. This action occurs, but is buried in the timebuild.pl script in “populatefromvbl.pl”.

6 Checkin Guidelines

This section really does not belong in this document, but it is included here since this is the first document you will see on enlisting, and you will likely want to begin work right after you enlist.

Future product plans are still being finalized. As a general guideline, please read and understand this mail snippet received from Brianv on the morning of 12/14/1999.

> *People should NOT NOT NOT be checking ANYTHING into a future release tree until we have detailed plans and schedules to support the feature, etc etc.*
>
> *People need to understand there is a very specific plan for the next release that needs to get developed and it will be met. There will be no random shit that is not approved allowed to be checked in. We are working these plans now - the next check point meeting is this Friday. Debbie is driving these meetings. We will focus on:*
>
> - *Ship no later than 4/01*
> - *NT everywhere release - get NT onto all computers. This is mainly a compat effort - but there are some logon changes that need to happen, etc. Some UI changes, etc... we know Win2000 is*

to hard for our moms to use and there is a few small things we can do to clean that up. Chris's group is working that plan and it's up to them to say what we can get done by 4/01 and then deliver it.

- *Engineering clean up - prefix everywhere, bugbug removal, etc etc etc - in fact this is what we should all do first and get it done - don't check new broken shit into the tree until we get control back of what's there already. This is what you and others are driving.*

- *Win64 port. This involves everyone.*

- *Making our existing services better internet aware services - like having AD be a good LDAP only directory for the internet, etc. Not big stuff - just make it so that ASP and ISPs can use our stuff. DaveTh owns this one.*

- *Deployment blockers. Whatever we need to do to keep Win2000 moving out the doors. Also gated by what can be done in 4/01. This is mainly a server thing and will come out of Dave's and Brian's planning efforts.*

There is nothing in a release like this that says it's drop as many features into as possible, etc etc etc. In fact, I'm going to be very hardcore on new stuff and make sure we don't get out of control. All of the teams are off developing plans and schedules now - but we should be hardcore on stuff coming in - there is no checkin window open for random stuff that doesn't have a full plan, spec, etc behind it and has been approved. And there never will be with this release.

6.1 Build Mail and Checkin Windows

Each team will determine their own check in windows and how check ins are communicated to the group. For Robs' team, we have not finalized these rules yet. Our goal is to allow a huge, frequent check in window with minimal communication. The following sections describe what we have discussed, but not 100% approved. You should follow these rules until you are told otherwise.

6.1.1 Build Mail

We will not use build mail following each and every check in. Our virtual build labs do not need to read build mail to track changes. Their mode of operation is full sync, followed by full build. You don't need build mail just to see what is going on. Use "sd changes –m 5" to see the last five changes in any given depot, or "sdx changes –m 5" to see the last five changes across all depots.

Mail should be sent to *loupdev* if:

- Your change affects in a material way, the way things build or run
- Your change fixes a setup break, boot break, or bugcheck that others are likely to hit
- Your change corrects a build break of any form (compile, link, postbuild)

Your mail should include information about what was fixed or changed. It should also include information about what depots were changed with the fix. If you can supply the sync and build instructions, that would be useful, but only do this if you really know what they are.

6.1.2 Checkin Windows

Eventually, we will automate our build process to perform clean syncs and builds at night. For the first month or two, we will be doing our clean syncs and builds during the daytime were we could easily react to build breaks and other problems that occur as we learn to use the new system.

Our initial checkin window will therefore be from 12:00 noon until 10:00 am the next morning. During this checkin window, we ask that you checkin code that compiles, links, and runs. Code that has been installed using setup as outlined in section 5.2, and finally, code that you believe belongs in the product you are working on based on Brianv's email excerpt found in section 6.

At times, the "it must compile and link and run" constraint might be a little too strong. For example, if you need to make a material change to something that is currently exported to other depots via the Public or Public\internal tree's, to ensure that your code compiles and links, and runs means that you need to be enlisted in all depots. This is reasonable if you have a development machine where this is possible (4 x XEON 450's or higher, 512Mb Ram, 72Gb Disk). If you don't have a machine like this, but find yourself

constantly making this magnitude of change, get a machine like this. If this class of machine isn't needed, then take advantage of "it doesn't have to compile Tuesday". Every Tuesday at 10am, we kick of a clean build just for fun. Check in your changes on Tuesday, but be on hand to fix build breaks as they occur during the build. If you have ensured that all of your depots build with your change, believe that your change is correct, have made a material change to files published to Public\ or Public\Internal, and have not physically compiled the other depots, you should check your code in any time from 8pm on Monday night until 10am Tuesday morning. During the build on Tuesday, we will monitor the build and fix our build breaks in realtime on the VBL build machines. If you are planning on making a checkin like this, you need to coordinate with robs and vinceo and will be on call during the entire build process to watch for errors and getting them fixed.

7 Private Branches

Many of you are anxious to begin working in source depot and immediately creating private branches. Pleas do not do this until you have been told how to create and manage and work in a private branch. We are not allowing branch creation or experimentation with branches until at least February 1, 2000. From now until then, don't even think about creating a private branch or asking anyone else to create one for you. You all need to become fluent in the use of source depot, the new tree layout, and the new build environment before you start experimenting with the complexities of branches.

8 Online Resources

A frequently asked questions (FAQ) document, and other online resources pertaining to the new build process and tools are available at http://ntbld/sdbuildprocess. Please make use of this document to see if your questions have already been answered.