To Build Windows Server 2003 RTM which was released on my birthday April 24th.

You need the source from the SSLIP Leak Account on the Internet archive from Microsoft's own Leak account. I'm not happy the way this is on the internet it used to be on Microsoft website here’s the press release. If you were able to sign up for the contract in 2002 like me there is a loophole of 700 resource hours. I do have a valid contract my System Integrator case is still open at Microsoft. Which is XP and 2003 SP1 Source. Only keep the 3.5, 4.0, XP/2003, Microsoft Patents and MS-DOS Source. Delete the rest. 2000 Source is only 33%.

For the Build Drive Install Windows Server 2003 R2 Enterprise 32-bit for SUA with 600GB 2 partition Oracle's Virtual Box Virtual Machine on Windows 10 2019 Enterprise LTSC. With 2048 MB of RAM and the I/0 AIPC setting checked. I haven't tested it on 2000 yet. In the D:\ Drive you need a Debug, Retail, VS, cpvsbuild, xpclient and depot shares for the Virtual Labs. The Microsoft Research Kernel You can get from my GitHub or from your Universities College of Mathematics and Computer Science System Admin. I added UK UNIX Bringingham Postscript Lectures.

You may want to try DR-DOS 7.03 from Caldera Licensing or Novell DR-DOS 7.0 from 1999 a updated from MS-DOS as the operating system stayed up to date until 2011 at Novell. and fetch the latest Perl distribution from the NT Sources Perl distribution and start DOS Globing. I'm working on a Alpha, VAX or MIPS server and I hope I can boot something from the DECUS archive. MIPS/NT allied at birth. Windows NT and VMS: The Rest of the Story. Visual Studio 2019 is in support until 2029 with Windows XP only for Visual Studio 2019 Compatibility.

The Official Build Number of the SSILP is 5.1.2600.6000

If you need to use 'expand /r' to X:\ENGLISH\WIN2003\ENT\I386* D:\binaries.x86fre from a retail DVD.

1. Build or find xcopy and findstr from the RTM DVD and place on path

2. Setting the Path:

```
path %path%;D:\NT\tools\perl\bin;D:\binaries.x86fre;D:\NT\tools\sp;D:\NT\tools\x86
```

```
set sdxroot=D:\NT
```

```
enlist projects ex: sdx enlist com -c
```

SET THE OFFICIAL BUILD MACHINE and BUILD NAME.

```
tools\ntnewver.cmd
```

3. Set the Signing Certificate

```
certmgr -add D:\NT\tools\testpca.cer -r localMachine -s ca
```

Run

```
tools\checktestroot.cmd and checktestca.cmd
```

4. Setting Razzle

```
tools\razzle free offline or tools\razzle win64 amd64 free offline
```

or for checked

```
tools\razzle offline or tools\razzle win64 and64 offline
```

```
perl xcopy2binplace. pl
```

build individual projects

in the directory of a makefile with

```
BUILD -cz
```

or

build the whole operating system

```
perl tools\timebuild. pl -NOCLEANBUILD -NOSYNC -NOSCORCH
```

```
BUILD /ZP
```

POSTBUILD:

.

```
tools\postbuildscripts\sanitycheckunicodefiles.cmd
```

We only create boot floppy images on fre compressed i386 builds

Generate the winnt32.msi for different SKUs

```
tools\postbuildscripts\winnt32msi.cmd
```

```
tools\postbuildscripts\makebuildname.cmd
```

```
tools\postbuildscripts\cdimage.cmd -d Release
```

move delobj.cmd the the tools directory to the NT Root and clean the source tree by deleting all the object files. after the build

Close Razzle Window run the VC7 vcvars32.bat from the VCBuilt share and set _NTTREE environment variable. Which is the binaries directory.

```
path %path%;D:\binaries.x86fre\bldtools
```

```
'copyddkfiles.cmd' the the NT\base\ddk
```

Copy the directory for the tool's directory run:

```
copyddkfiles.cmd ddk_base.ini ddk D:\NT\base
```

It should begin to copy the ddk files you will need to edit the ddk, hal, IFS INI files to fit your kit needs.

After the build run the post build batch file in the tools directory and read the error log you shouldn't get any errors in Windows XP/2003 Professional Edition.

5. certmgr.msc, go to Trusted Root Certification Authorities\Certificates and remove the Microsoft Test Root Authority certificate, Sign out and Sign in again.

It takes 6 hours to build a Free Build and 12 hrs a Checked.

After the build the Active Directory Migration Tools and Debugging Tools should be built. After the post build very edition and every supported language should be built. In the binaries directory the usa default build should be revision 6000 leading up to Windows Embedded 2009. After the Windows 6000 revision source build you might want to use the SLD files in the 'mantis' folder with XP embedded and make a Embedded Windows repository and engineer 2003 revision 6000 which was released in Embedded 2009.

A build lab (or simply a lab) commonly refers to a Microsoft Windows source code branch. By extension, it can also refer to the team that works on this branch.

Historically, the build lab was a physical room populated with machines that periodically produced mainline builds of Windows NT. At first, there was a single build lab for the entire project that all developers submitted their changes to. As the number of developers grew, the concept of virtual build labs was introduced, where each team developing a subset of the feature set works on its own separate branch and has one or more machines compiling periodic builds of that branch. Each developer submits their changes first to their team's branch, and they are merged to the mainline only after they meet its acceptance criteria. Other teams can then pull the changes from the mainline into their branches.

Builds that were not built by the build lab machines, but are rather compiled by individual Microsoft employees, are called private builds. They are identified by having the VS_FF_PRIVATEBUILD file flag set in the executable's version information and by including the account name of the individual or service that initiated the build in the branch part of the build tag.

Microsoft has used several branching systems since the start of Windows XP development, which differ in the hierarchy of labs as well as in the naming scheme. However, there are some specifics common to most schemes, such as the presence of a top-most main branch, or the use of special branches for important development milestones.

Whistler and pre-reset Longhorn

The top-most branch was called main, which integrated changes from all labs. Under it were several numbered labs, each of which was working on a separate part of Windows, such as:

Lab01: Kernel

Lab02: Networking

Lab03: Server[c]

Lab04: Terminal Services[1]

Lab06: User interface

Lab07: Internet Information Services/COM+

These labs also had a "_N" branch, which served as a buffer between main and the actual branch. For example, changes from the main branch would first get integrated into Lab06_N before later being integrated into Lab06 in a process called forward integration. The same applied for reverse integration, where the lab would first integrate changes into its N-branch before integrating them into the main branch.

There were also idx (internal development workstation/server) branches, builds from which are usually recompilations intended for TAP/OEM partners. However, they were also occasionally released for public testing, e.g. Windows Longhorn build 4074 or Windows XP build 2257.

Before the release of Windows XP, the main branch was forked into the xpclient branch while the main branch moved on to track Windows Server 2003 development. After the final version was shipped, new branches were created for updates, hotfixes and Service Pack development. Similarly, the dnsrv branch (short for Dot NET Server, i.e. Windows .NET Server) was forked from the main branch before the release of Windows Server 2003.

Since Longhorn reset

The lab hierarchy was overhauled after the development reset of Longhorn to address the flaws that plagued most of the pre-reset period. Instead of having a small amount of general virtual build labs each focusing on a different general scope of Windows functionality, a new hierarchical model with considerably more feature branches was introduced, which helped reduce the amount of code to reverse integrate for branch. Microsoft also set stricter criteria for reverse integrating changes from the labs into the main branch, which was now renamed to winmain.

The main branch was later renamed to rsmain at some point after the release of Windows 10, and then to rsmaster after the Windows source repository's conversion to Git, likely to comply with the Git convention of calling the top-most branch the master branch. At some point after February 2021,[2] the branch was renamed back to main. This was likely done in order to follow suit with other Microsoft projects after the master/slave terminology became a subject of controversy in 2020 due to slavery connotations.[3][4]

Feature branch prefixes

Microsoft has used multiple prefixes to refer to feature branches over time:

- vbl (virtual build lab) - Windows Vista
- fbl (feature branch level) - Windows 7 up to Windows 10 (original release)
- rs (Redstone) - Windows 10 Creators Update up to now
- fs (Firesteel) - Windows 11 (original release)[d]

Windows 10 November Update and Windows 10 Anniversary Update used their respective release branch prefix for feature branches - th2 and rs1, respectively.

Release branch prefixes

Similarly to the previous system, the main branch is forked off before release to contain update development. The following is the list of known prefixes:

vista, longhorn, lh - Windows Vista, Windows Server 2008

win7 - Windows 7, Windows Server 2008 R2

win8 - Windows 8, Windows Server 2012

winblue - Windows 8.1, Windows Server 2012 R2

th1 - Windows 10 (original release)

th2 - Windows 10 November Update

rs1 - Windows 10 Anniversary Update, Windows Server 2016

rs2 - Windows 10 Creators Update

rs3 - Windows 10 Fall Creators Update

rs4 - Windows 10 April 2018 Update

rs5 - Windows 10 October 2018 Update, Windows Server 2019

19h1 - Windows 10 May 2019 Update, Windows 10 November 2019 Update

vb (Vibranium) - Windows 10 May 2020 Update and later cumulative updates

mn (Manganese)

fe (Iron) - Windows Server 2022

co (Cobalt) - Windows 11 (original release)

ni (Nickel) - Windows 11 2022 Update, Windows 11 2023 Update

zn (Zinc) - Windows Server, version 23H2