The AutoInstaller Project

Microsoft's Windows RT's goal was to create a viable tablet OS that was affordable, easy, power thrifty and secure. With the success of the Windows 7 release, Microsoft quietly partnered with select IHVs with ARM and smartphone saavy. Texas Instruments (TI) was picked to provide an ARM tablet for Windows 8. The TI team's responsibility was to provide an ARM tablet product which met Microsoft's criteria – Bluetooth, USB, WiFI, hi-resolution Audio-Video, SD storage, next generation sensors and camera – all in a small, affordable package. With that hardware, we also needed to create ultra-reliable device drivers, which could pass the Windows Hardware Quality Assurance test suits (WHQL).

At the beginning of the MSFT-TI WOA collaboration, the ARM tablets were essentially beta-ware smartphones. TI had a solid history of producing world class smartphones and tablets for the Android and Windows Phone OEMs, but the specifications and feature sets for phones were far simpler than what Microsoft was requiring for WOA. Because Microsoft Windows (the NT family, including 7, 8 and 10) was designed to be processor portable, it didn't take long for Microsoft to come up with a generic ARM port of their WinPE (Windows Pre-Installation Environment) OS. The WinPE was used as an "Emergency" boot OS, but was not the full-fledged Windows.

The build cycle went like this. Weekly, Microsoft would give TI a pre-release Windows build with notes on its stability. TI would in turn give Microsoft the latest firmware and driver versions we had built and tested. The ARM tablet's hardware and peripherals had their own code, called the firmware. This was installed using the UEFI (a modern, unified BIOS – see More on UEFI), which runs at a level below the OS. The firmware controlled the devices, and the device drivers allowed the OS to engage with the devices. Both the firmware and device drivers were linked and needed to be in sync with each other.

The test team (me and my colleagues) would install this latest MS OS with our latest device drivers and do some basic sanity tests before deciding to release the OS to the team. Up the hill, our Microsoft friends would be trying our drivers/firmware with their latest OS. There were a lot of quality problems on both sides with the early OS and driver/firmware releases, but this was overcome.

The first few months, it was acceptable for developers to hand install the firmware, OS and drivers, but as development accelerated and the number of tablets available increased, it was clear that automated installation was needed – and that was one of my tasks.

The WOA OS was a clean installation only (no upgrades), and needed reimaging the device mass storage with fresh OS partitions prior to installation beginning. The tablets would boot off any bootable USB drive, so I created two of these for setup. The first would update the Firmware, and the second would launch the OS installation.

I include two sets of documents I wrote. "`Using the AutoInstaller`" was for end users. "`Installwoa Script`" describes the setup process.

The "`CallInstallWoA.cmd`" started the setup process once the ARM tablet booted from the Setup USB drive. This checks the environment, looks for any "persisted data" files, and then calls "`installwoa.cmd`", which does the actual installation, beginning with using the Windows DISKPART utility to reimage the mass storage device and set up the fresh OS partitions.

Although the Windows On ARM project is long over, these batch files do have some cunning tricks within. For example, I use the Windows PING command as a thirty second timer (some of the installation procedures needed a delay).

```
ping -n 30 127.0.0.1 > nul
```

I have also included the Image directory, which automates the reimaging of the mass storage device using the Windows DISKPART utility.

I have renamed the scripts with a .txt extension to prevent any accidental running (calling DISKPART on your system could be dangerous).

Hope you enjoy!

Percy Tierney