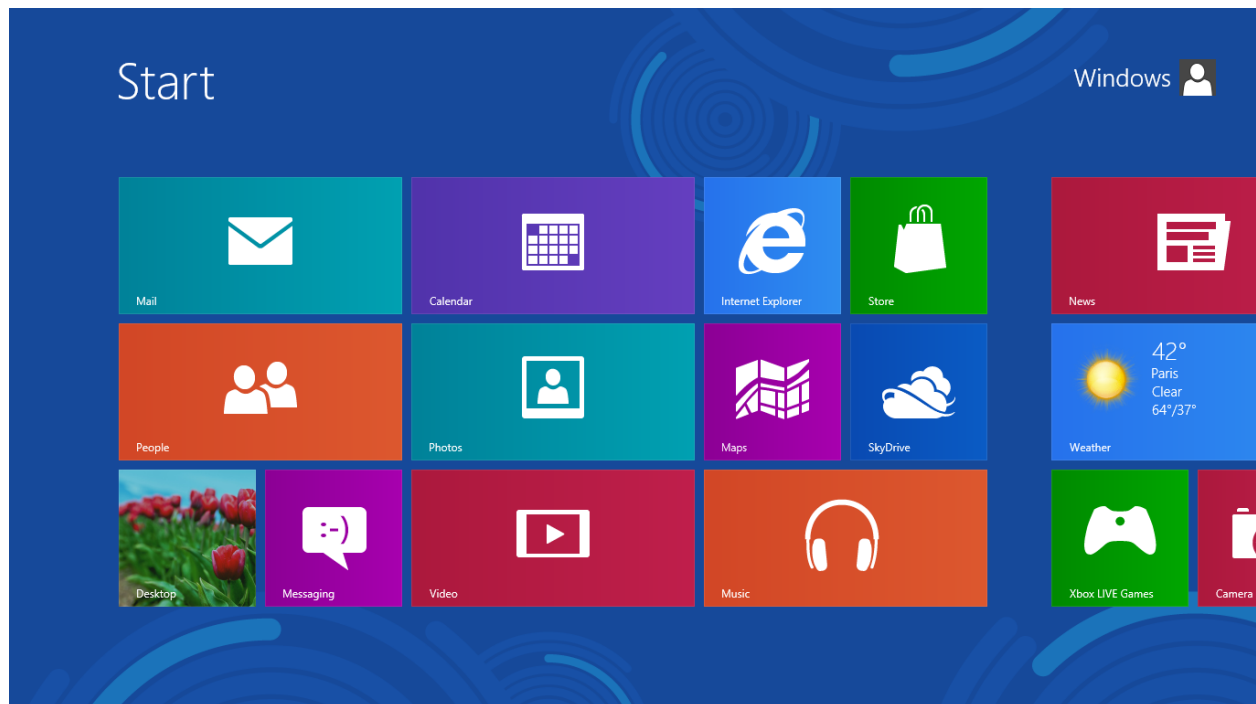


The Building Windows 8 Blog or “b8”

This blog was a product of the Windows 8 development team from August 2011 through October 2012. It was originally hosted on blogs.msdn.com which was relocated and in the process images and links were broken. I’ve produced a single PDF with all the posts and as many images as I could reconstruct for the benefit of *Hardcore Software* on hardcoresoftware.learningbyshipping.com and also contributed to the Internet Archive project.

Please enjoy, Steven Sinofsky August 2022



- 2011-08-15 Welcome to Building Windows 8
- 2011-08-17 About this blog and your comments
- 2011-08-17 Introducing the team
- 2011-08-22 Building robust USB 3.0 support
- 2011-08-23 Improving our file management basics copy, move, rename, and delete
- 2011-08-26 Designing the Windows 8 file name collision experience
- 2011-08-29 Improvements in Windows Explorer
- 2011-08-30 Accessing data in ISO and VHD files
- 2011-08-31 Designing for Metro style and the desktop
- 2011-09-01 Reflecting on our first conversations (part 1)
- 2011-09-02 Reflecting on our first conversations (part 2)
- 2011-09-07 Bringing Hyper-V to “Windows 8”
- 2011-09-08 Delivering fast boot times in Windows 8
- 2011-09-13 Experiencing Windows 8 touch on Windows 7 hardware

2011-09-13 Welcome to Windows 8 – The Developer Preview
2011-09-14 Best place to discuss Windows 8
2011-09-14 Metro style browsing and plug-in free HTML5
2011-09-14 Metro style browsing one engine, two experiences, no compromises
2011-09-15 Protecting you from malware
2011-09-16 Running Windows 8 Developer Preview in a virtual environment
2011-09-20 Reengineering the Windows boot experience
2011-09-22 Protecting the pre-OS environment with UEFI
2011-09-26 Signing in to Windows 8 with a Windows Live ID
2011-09-28 Extending Windows 8 apps to the cloud with SkyDrive
2011-10-03 Evolving the Start menu
2011-10-04 Designing the Start screen
2011-10-07 Reducing runtime memory in Windows 8
2011-10-11 Reflecting on your comments on the Start screen
2011-10-13 The Windows 8 Task Manager
2011-10-18 Designing search for the Start screen
2011-10-20 Optimizing for both landscape and portrait
2011-10-27 Using Task Manager with 64+ logical processors
2011-11-02 Updating live tiles without draining your battery
2011-11-08 Building a power-smart general-purpose Windows
2011-11-14 Minimizing restarts after automatic updating in Windows Update
2011-11-21 Improving the setup experience
2011-11-29 Enabling large disks and large sectors in Windows 8
2011-12-06 Windows Store event and blog
2011-12-14 Protecting your digital identity
2011-12-19 Optimizing picture password security
2012-01-04 Refresh and reset your PC
2012-01-05 Virtualizing storage for scale, resiliency, and efficiency
2012-01-16 Building the next generation file system for Windows ReFS
2012-01-20 Engineering Windows 8 for mobile networks
2012-01-24 Supporting sensors in Windows 8
2012-01-30 Acting on file management feedback
2012-02-07 Improving power efficiency for applications
2012-02-09 Building Windows for the ARM processor architecture
2012-02-14 Enabling accessibility
2012-02-16 Internet Explorer Performance Lab reliably measuring browser performance
2012-02-20 Connecting your apps, files, PCs and devices to the cloud with SkyDrive and Windows 8
2012-02-29 Running the Consumer Preview system recommendations
2012-02-29 Welcome to Windows 8 – The Consumer Preview
2012-03-06 Going behind the scenes building Windows 8
2012-03-13 Web browsing in Windows 8 Consumer Preview with IE10
2012-03-21 Scaling to different screens
2012-03-28 Touch hardware and Windows 8
2012-04-17 Reclaiming memory from Metro style apps
2012-04-19 Managing BYO PCs in the enterprise (including WOA)

2012-04-23 Making personal cloud storage for Windows available anywhere, with the new SkyDrive

2012-05-02 Cloud services for Windows 8 and Windows Phone Windows Live, reimagined

2012-05-03 Making Windows Media Center available in Windows 8

2012-05-04 FAQ – DVD playback and Windows Media Center in Windows 8

2012-05-09 Redesigning chkdsk and the new NTFS health model

2012-05-14 Keeping your family safer with Windows 8

2012-05-17 Delivering reliable and trustworthy Metro style apps

2012-05-18 Creating the Windows 8 user experience

2012-05-21 Enhancing Windows 8 for multiple monitors

2012-05-22 Designing for PCs that boot faster than ever before

2012-05-31 Delivering the Windows 8 Release Preview

2012-06-01 Web browsing in Windows 8 Release Preview with IE10

2012-06-05 Connecting with IPv6 in Windows 8

2012-06-08 Building a rich and extensible media platform

2012-06-11 Activating Windows 8 contracts in your app

2012-06-13 The People app the complete, cloud-powered address book for Windows 8

2012-06-14 Building the Mail app

2012-06-15 Designing the Windows 8 Calendar app

2012-06-26 Introducing the Photos app for Windows 8

2012-07-03 Ready Metro style apps for launch

2012-07-10 Protecting user files with File History

2012-07-17 Designing the Windows 8 touch keyboard

2012-07-18 Using the language you want

2012-07-18 Using your feedback to make Narrator work better with touch

2012-07-23 Hardware accelerating everything Windows 8 graphics

2012-07-25 Signing in with a picture password

2012-07-25 Simplifying printing in Windows 8

2012-08-01 Releasing Windows 8 – August 1, 2012

2012-08-13 Collaborating to deliver Windows RT PCs

2012-10-04 Updating our built-in apps for Windows 8

2012-10-09 Updating Windows 8 for General Availability

Welcome to Building Windows 8

Steven Sinofsky | [2011-08-15T12:02:00+00:00](#)

Building the next release of Microsoft Windows is an industry-wide effort that Microsoft approaches with a strong sense of responsibility and humility. Windows 8 *reimagines* Windows for a new generation of computing devices, and will be the very best operating system for hundreds of millions of PCs, new and old, used by well over a billion people globally.

We've been hard at work designing and building Windows 8, and today we want to begin an open dialog with those of you who will be trying out the pre-release version over the coming months. We intend to post regularly throughout the development of Windows 8, and to focus on the engineering of the product. Welcome to "Building Windows 8," or as we call it, "B8."

For the Windows team, this blog is an important part of developing Windows 8, as was our blog for Windows 7. Blogging allows us to have a two-way dialog with you about design choices, real-world data and usage, and new opportunities that are part of Windows 8. Together, we will start the unique adventure of bringing a major product to market. We're genuinely excited to talk about the development of Windows 8 and to engage thoughtfully with the community of passionate end-users, developers, and information professionals.

Reimagining Windows from chips to experience

Windows 8 reimagines Windows. That's a big statement and one that we will return to throughout this blog. It is also important to know that we're 100% committed to running the software and supporting the hardware that is compatible with over 400 million Windows 7 licenses already sold and all the Windows 7 yet to be sold.

But so much has changed since Windows 95—the last time Windows was significantly overhauled—when the "desktop" metaphor was established. Today more than two out of three PCs are mobile (laptops, netbooks, notebooks, tablets, slates, convertibles, etc.). Nearly every PC is capable of wireless connectivity. Screen sizes range from under 10" to wall-sized screens and multiple HD screens. Storage has jumped from megabytes to terabytes and has moved up to the cloud. The appearance of touch-screen mobile phones with the rich capabilities they bring, have together changed the way we all view computing. Most of all, computing is much more focused on applications and on people than on the operating system itself or the data. These changes in the landscape motivate the most significant changes to Windows, from the chips to the experience.

We showed you a [preview of Windows 8](#) in June, demonstrating the user experience and providing an [update on ARM SoC support](#). The next major event for Windows is our [BUILD conference](#) in September, where we will provide developers with more details about the full spectrum of tools and capabilities available to make the most of Windows 8. This blog is a chance for us to discuss the details and provide a behind the scenes look at the evolution of Windows 8.

With our preview in June, we started by showing you user experience, because it is the most visible change to Windows. Rest assured we've thoughtfully engineered changes across the full range of Windows capabilities. But this presents us a challenge in deciding where to start the dialog.

We know people who care a lot about networking want to know our plans there. We know people who are invested heavily in storage want to know what's new in that area. Many want to know about performance and fundamentals. We know developers, IT pros, and gamers all want to know what's new for them. There is so much packed into Windows 8 and there are so many unique and important lenses through which to view Windows 8, and so we want to be sure to take the time to cover as many of these topics as possible, to build up a shared understanding of why we've taken Windows where we have. So in the next weeks we will just start talking specifics of features, since there is no obvious place to start given the varying perspectives. From fundamentals, to user interface, to hardware support, and more, if something is important to you, we promise we'll get to it in some form or another.

We've heard people express frustration over how little we've communicated so far about Windows 8. We've certainly learned lessons over the years about the perils of talking about features before we have a solid understanding of our ability to execute.

Our intent with this pre-release blog is to make sure that we have a reasonable degree of confidence in what we talk about, before we talk about it. Our top priority is the responsibility we feel to our customers and partners, to make sure we're not stressing priorities, churning resource allocations, or causing strategic confusion among the tens of thousands of you who care deeply and have much invested in the evolution of Windows. Rather than generating traffic or building excitement, this blog is here to provide a two-way dialog about the complexities and tradeoffs of product development.

Focusing on engineering

We started the Engineering Windows 7 blog in 2008 in recognition of the need to re-engage the community and rebuild trust relative to the engineering and design of Windows. While engineering Windows 7, we learned some great lessons and renewed our sense of responsibility to the community.

As we moved on to building Windows 8, we took those values and have built on them. Our focus on performance, reliability, compatibility, security, and quality is now baked into our engineering process even as we change Windows for a new generation. With these changes come new ways of doing work on Windows PCs as well as continual investments in hardware, software, and peripherals.

We intend to continue our dialog around performance and fundamental engineering of Windows. The feedback on these topics and the desire to talk about them in depth was clear during the development of Windows 7.

Starting our dialog

We know that blogging about Windows 8 will bring out the passionate opinions of many people, including members of our team. As a team we're all going to participate—many of us will author posts, and all of us will read and take note of your comments on this blog. We'll participate in a constructive dialog with you. We'll also make mistakes and admit it when we do. It is almost certain that something will hit a nerve, with the team or with the community, or both, in the blog posts or in the product, or both. In any case, we'll work hard to have constructive conversations with you, share the data, and, when the situation calls for it, make thoughtful changes.

Feel free to send us your thoughts via comments or email—we can't respond to every question we receive, but your suggestions for blog topics are welcome. The email contact link in the right pane goes straight to my inbox without any filter (except spam filtering). Please note that we are also making this blog available in several other languages (acting on feedback from the Engineering Windows 7 blog) and you can expect to see those posts within 48 hours of the English language post.

If you're looking for notifications of posts, then be sure to follow us on Twitter **@BuildWindows8**. Look for shortened URLs at "**win8.ms**" with links to posts and videos.

With that, we'll just ask you to stay tuned and join us in this dialog about the engineering of Windows 8.

--Steven Sinofsky

About this blog and your comments

Steven Sinofsky | [2011-08-17T10:15:00+00:00](#)

Thank you very much for the warm reception. It is humbling to see the amount of interest and enthusiasm for the blog and for Windows. We've been digesting all the comments (lots of mail is going around internally about specifics) and I've been going through my overflowing inbox (I can't reply to every message, but I have been replying)! Thank you. The most popular question/comment is about "signing up for the beta". We will be up front and very visible with any pre-release software programs that you can opt into. Promise.

As we begin our discussion of building Windows 8 on this blog, two housekeeping topics are worth a post before we start talking about building Windows 8 and the product. We want to be up front about the writing on this blog and provide a view on comments.

This blog is 100% authentic "engineer written" and not a marketing or communications effort. We do not have ghost-writers, editors, or any process that attempts to sanitize the words of folks on the team other than some basic copy editing.

This has the benefit of giving you truly authentic posts that directly reflect the passion of those developing the product. It also means that this blog is not written by professional writers. Some posts will go into a lot of detail. Posts by different writers will all have different "voices." We ask that folks not be critical of the writing, keeping in mind the direct approach we are taking—I promise you that every writer will take personal comments, well, personally.

We had a long discussion about how to handle comments for Building Windows 8 ("B8"). Our experience with the Engineering Windows 7 blog ("E7") comments was mostly positive, but a non-zero number of folks abused their direct access to both email and comments. Let's keep the comments in the spirit of the community and avoid using comments for unrelated topics.

Of course, the primary goal of this blog is to have a two-way dialog, so comments are an important part of what we're looking for with our blogging efforts. So we opted for the commenting mechanism already used by thousands of MSDN blogs—anonymous comments are permitted, and they appear without moderation. This blog platform employs some minor security measures and a spam filter that we do not control.

That means we are seeking out comments. Everyone on the Windows team will be watching for comments and is looking forward to the dialog. When participating, we will work to make sure Microsoft employees represent themselves as such, especially indicating if they work on the area Windows being discussed. We ask that press (those that write, blog, tweet professionally) identify themselves accordingly as well.

Things we hope to see in comments:

- Lots of on-topic, good, interesting thoughts on Windows and the posts on B8
- Focus on the content of the post and not just the topic in general—seek out the details
- Dialog that is respectful and fun

We reserve the right to delete comments or otherwise edit what has been said. Things that will get comments edited or deleted:

- Offensive or abusive language or behavior as determined by a community standard
- Misrepresentation (i.e., claiming to be somebody you're not) — if you don't want to use your real name, that's fine, as long as your profile name isn't offensive, abusive, or misrepresentative
- Repeatedly posting the same comments or agenda, or attempting to fit a specific topic into every post, no matter what we blog about
- Blog-spam or link-abuse of any kind

We hope these rules will keep the discussion lively and on-topic.

--Steven

Introducing the team

Steven Sinofsky | [2011-08-17T13:00:00+00:00](#)

Thanks for the comments and the flood of email we received (and to the number of folks now following us on Twitter, too). It is definitely humbling to see all the enthusiasm and interest. There are clearly already few important threads in the initial comments, some of which are based on the previews of the Windows 8 user experience. We're definitely gearing up to discuss these issues, the design, and tradeoffs. Windows 8 has new features across the full breadth of the product. It takes quite a team to build Windows 8, and so I thought it would be a good idea to talk about the team structure—sometimes the “how” can help folks to understand the “what” and the “why.” This will give you an outline of the places we added features to Windows 8. It will also serve as a bit of a guide as we talk about the product.

It is tempting for some to think of Windows as one entity or group, or for some to think of Windows as just a set of specific people. Sometimes someone speaks at a conference or has a blog, and that comes to represent the product for you. In reality, Windows is always a product of the whole team and much of Microsoft. Nearly every development group contributes to building Windows 8 in some form or another. And Windows contributes efforts to most other groups as well.

Windows is a fairly broad project made up of a set of coordinated smaller projects. When we started building Windows 8 we had a clear sense of the direction we were heading and so we built a team structure to support that direction. Many of the teams work together while at the same time we try to break the work down into fairly independent groups—obviously as a customer you want things to work together, but as an engineer, you also want to be able to work independently. That's a fine balance we work to maintain.

A lot goes into building a team structure to get all the work of Windows done. The most important first step is deciding “what” we plan to get done, so that we can make sure we have the best teams in place and the best structure to do that work. At the same time we have to make sure all the engineering processes—like daily builds, integration, quality, security, and all the fundamentals—are integral from the start (lots to talk about on these topics!).

We have several engineering roles, or disciplines, that make up our team. The implementation work on Windows happens when [developers](#) write code. This code implements features that come from specifications written by [program management](#) along with interaction designs from our product designers. [Testers](#) are responsible for making sure the spec is complete and the code does what the spec says it should do. This is a simplified view of the relationship between roles, since we routinely walk a bit in each other's shoes. There are several other equally important roles on the team, but we tend to think of our engineering effort as development, test, and program management working together in lockstep throughout the entire release—each role has an equal voice in the outcome and choices we make.

We organize the work of Windows into “feature teams,” groups of developers who own a combination of architectural elements and scenarios across Windows. We have about 35 feature teams in the Windows 8 organization. Each feature team has anywhere from 25-40 developers, plus test and program management, all working together. Our teams are all focused on building a global product, and so some of our teams are located outside the US and are also delivering globally.

In general a feature team owns and builds what that most folks would identify as an area or component of Windows. “Feature” is always a tricky word—some folks think a feature is a broad architectural component like the “user interface” or networking, and other folks might think a feature is something more specific, like the “start menu” or IPv6. So the term is a bit overloaded. When we set up different feature teams, we pair the architecture (code, subsystems, components) with the scenarios (user experience) in which users will encounter it, while also working to make sure we keep teams small and manageable. We long ago stopped trying to count new features because of the difficulty in defining a feature. We do count work items, which do map to the work and specs that we build (but that is a pretty long list).

When folks do the math and come up with the number of developers on the team, we usually hear one of two reactions: “wow, that is a lot, and there is no way that can work,” or “wow, you build a product for a billion people with a pretty small number folks.” It is to our benefit to have the smallest number of people on the team possible, but it is to your benefit to have the largest number of people adding all the things that folks might want. So, we find a place in the middle. We want the team to be manageable and able to produce high quality, full-featured code.

I mentioned earlier that Windows contributes code to lots of other products and vice versa, so when you look at this list, keep in mind there are features from other groups (for example, our browser language runtime comes from the development tools group) and some of the work here goes into other products, too. For example, all of our kernel, networking, storage, virtualization, and other fundamental OS work is also part of Windows Server—that's right, one team delivers the full Windows Client OS and much of the foundation for the Windows Server OS. And some features are built in the core OS but are ultimately only part of the Server product.

Many of the teams listed below describe features or areas that you are familiar with or that you can probably figure out based on the name. As we post more, team members will identify themselves as part of these teams. We also have organized these teams in seven larger groups that pull related teams together—fundamentals, devices and networking, core OS, developer experience, user experience, web services, and our engineering system. The Windows Live group (Hotmail, Messenger, Skydrive, Photos, LiveID, and more) also has a similar structure. Internet Explorer group is also a couple of teams on its own, but obviously contributes across Windows 8.

- App Compatibility and Device Compatibility
- App Store
- Applications and Media Experience
- App Experience

- Core Experience Evolved
- Device Connectivity
- Devices & Networking Experience
- Ecosystem Fundamentals
- Engineer Desktop
- Engineering System
- Enterprise Networking
- Global Experience
- Graphics Platform
- Hardware Developer Experience
- Human Interaction Platform
- Hyper-V
- In Control of Your PC
- Kernel Platform
- Licensing and Deployment
- Media Platform
- Networking Core
- Performance
- Presentation and Composition
- Reliability, Security, and Privacy
- Runtime Experience
- Search, View, and Command
- Security & Identity
- Storage & Files Systems
- Sustained Engineering
- Telemetry
- User-Centered Experience
- Windows Online
- Windows Update
- Wireless and Networking services
- XAML

In addition to these teams made up of development, test, and program management, there are many others that are part of the product development team. Our content development team writes and edits our online assistance, website, deployment documents, and SDKs, to name a few things. Product planning leads customer and market research and also pays very close attention to feedback and telemetry around the pre-release software. Product design develops the overall interaction model, graphical language, and design language for Windows 8. Our research and usability team creates field and lab studies that show how existing products and proposed features perform with all types of customers. Localization brings Windows to over 100 languages (and localizes this blog). Our operations team runs services that are used by hundreds of millions of people and almost a billion PCs. Just to name a few..

When we started Windows 7 some people told us that the Windows team was too big and had reached a size that caused more engineering problems than it solved. At the same time, you can look at all the comments and see the incredible demand for new features across a very wide range of scenarios.

Folks want new things, and changes to existing things; they want features to be available globally, to be accessible, and to be super high quality; they want things to work on existing hardware, and to take advantage of the latest new hardware. Our job is to get as much done in as short a time as possible, at a very significant scale. That's all a pretty significant engineering effort.

For folks who are counting my words, I am still under 1,500 words, so I think I will call this an introduction to the team. Keep the comments coming, as they are helping us get ideas for posts and shape the dialog. I hope this post helps to develop some shared context in terms of talking about Windows 8.

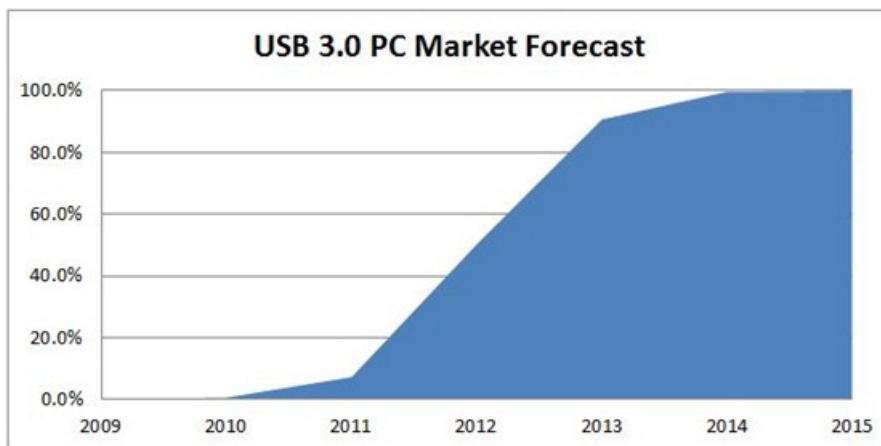
--Steven

Building robust USB 3.0 support

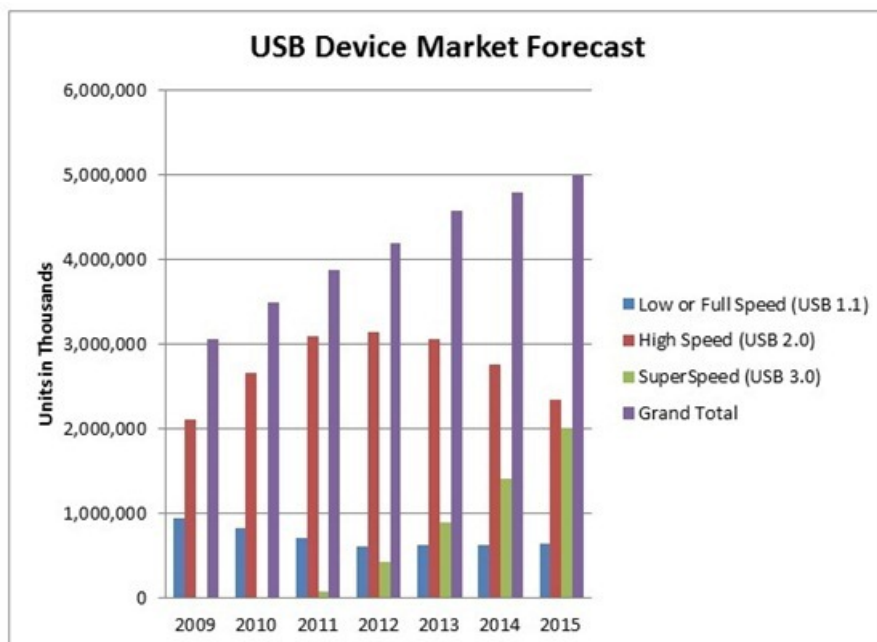
Steven Sinofsky | [2011-08-22T12:00:00+00:00](#)

One of the important roles Windows plays as part of a broad ecosystem is developing support for new hardware. This is a pretty involved process and so for this post we wanted to take a look at supporting USB 3.0, something we know everyone is anxious to be using because of the improvements it brings. This is also our first video post – we aimed for "engineering" videos and not high production values but I think we make our point (note videos are embedded in HTML5 and available for download). If you're like me when looking at the video, you might think that those file copy progress indicators are looking a bit dated...stay tuned. This post was authored by Dennis Flanagan, the Director of Program Management for the Devices and Networking group. –Steven

With throughput up to 10 times faster than USB 2.0 and improved power management that results in longer battery life, USB 3.0 introduces compelling reasons to improve the world's most popular PC interface. By 2015, all new PCs are expected to offer USB 3.0 ports, and over 2 billion new "SuperSpeed" USB devices will be sold in that year alone.



In-Stat, June 2011



In-Stat, June 2011

The decision to invest in USB 3.0 was an easy one to make, but doing so without compromising the existing USB ecosystem was a big challenge to overcome. Our design had to follow the revised 3.0 specification precisely in order to enable emerging USB 3.0 hardware. There are also billions of older USB devices that Windows must remain compatible with. How do you write a single piece of software to enable the latest technology on evolving hardware, while making sure it still works with 10 billion existing devices in homes and offices across the world?

First, a bit of history

In 1996, the [USB standards organization](#) released the first USB specification, which defined two speeds for USB devices: low speed (1.5 Mbps) and full speed (12 Mbps). At the time, the idea of "hot plugging" a device (plugging in and unplugging without needing to reboot), was revolutionary. USB also supported different ways to transfer data: *bulk*, for devices like printers that send a lot of data and forget about it;

isochronous, for devices like speakers that continuously receive data in a specific order; and *interrupt*, for devices like keyboards that only send data once in a while.

The 1996 specification also moved the complexity from the USB device into the PC, making devices cheaper and simpler to implement. These features made USB the most attractive external device connector. As a result, device makers adopted USB and joined the standards body to define common interfaces between software and hardware for different classes of devices. These common interfaces allow a single software driver, a class driver, to support an entire type of device. From the beginning, Microsoft embraced the USB technology and the standards organization, contributing to many specifications over the years. We introduced USB 1.1 support in Windows 95 OSR 2.

In 2000, the USB 2.0 specification came to light with a new, high speed (480 Mbps). Unfortunately, the host controller, the hardware used to connect a PC to devices, was not compatible with earlier versions. High speed devices worked with all controllers, but low and full speed devices could not work with USB 2.0 controllers. PCs needed to ship with two different controllers, or embed a USB 2.0 hub, in order to support all types of devices. In Windows XP SP1, we enhanced our existing software driver stack by adding USB 2.0 functionally.

The path to USB 3: start with a solid specification

By actively participating in the USB standards organization, we helped create a specification that was both compelling *and* interoperable. Like other members of the USB Implementer's Forum, we wanted to see a faster, more power-efficient version of USB, where, unlike USB 2.0, a single combination of hardware and software could work with all USB devices.

In 2008 the USB Standards organization released the new [USB 3.0 specification](#), which included a new host controller and defined the new "SuperSpeed" USB device (5 Gbps). Together, the controller and device could operate at theoretical speeds of up to 10 times faster than USB 2.0. With this new standard, you'd be able to copy a high definition movie from a USB storage drive in about 80 seconds instead of the 15 minutes it takes with USB 2.0. The specification also introduced a new transfer type —streams— which allows the storage drives to process reads and writes more efficiently.

The new specification provided 80% more power than USB 2.0. This meant faster charging and removed the need for odd "Y" cables used by external DVD drives and other high power devices. But charging isn't the only power consideration. With mobile computing, people want PCs that conserve battery life. By also introducing new low power states, finishing tasks more quickly, and powering down at every opportunity, USB 3.0 is more power efficient than its predecessors. This translates to longer battery life for notebooks and less power consumption for desktops.

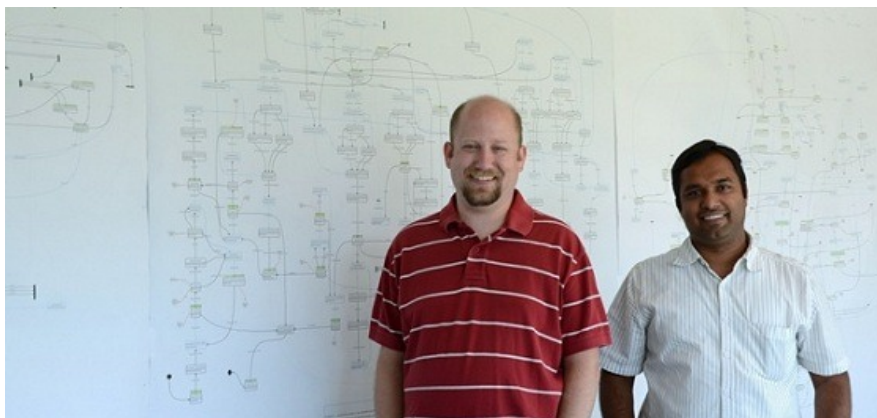
Most importantly, the specification promises to enable a new generation of USB while maintaining compatibility with full, low, and high speed devices. Even the plugs are backwards compatible.

Close partnership with the hardware industry

As the specification began to solidify, we started to design Windows and faced our first difficult decision. Do we update our existing USB software, which we've gradually modified since Windows 95 OSR2, or, do we write new software following modern design principles? Countless devices and their drivers rely on the behavior of our existing software, so we couldn't simply jump into a brand new design. The solution? Don't jump in. Instead, meticulously design a new USB software stack for the new controller while maintaining existing interfaces and behaviors, ensuring every device and driver will work. For older controllers, we retained our existing software stack.

To create a brand new USB software stack, we had to start work early. If we waited for hardware to be available we would be too late to support the budding USB 3.0 ecosystem. We decided to start before there were any USB 3.0 devices by building "virtual" devices. Virtual devices are software representations of real, physical USB hardware: the ports, the hubs, and other devices.

With virtual device development underway, we started designing and prototyping. USB software is complex because it has to manage hubs and devices while still dealing with any errors. To create something with longevity we needed to visualize and document the flow. We designed three massive flow charts and a code generator to automatically convert a Visio diagram into software. Together with [Microsoft Research](#), we refined a tool called [Zing](#), which could validate every aspect of this software model.



Flow chart with its architects, Randy Aull and Vivek Gupta

Once we finished some initial development, the first USB host controllers arrived. We recognized that simulation provided a great starting point, but it wasn't enough. With hardware, we identified incorrect assumptions, timing issues, and other problems unique to real-world scenarios. The path forward also required us to help foster a new ecosystem that would require a strong collaboration with our hardware partners. We needed to work together to prototype, exchange ideas, have deep technical discussions, and report bugs in both directions. We had to build a shared commitment to work together closely so we could identify issues before designs were final.

As USB 3.0 development progressed, so did a sense of community. Software is only as successful as the hardware it enables. Together with our partners in the hardware industry, we were finding issues, developing solutions, and creating the foundation for a new USB ecosystem.

I test, therefore I am

While we were focused on building support for USB 3.0 chips inside the PC, we couldn't ignore the world of devices. We had to think outside the box – literally. There are over 10 billion USB devices worldwide. Some are in use daily and some are tucked away collecting dust, but all were originally designed to work with Windows PCs. Compatibility is the Windows promise. Our customers have grown accustomed to expecting new versions of Windows to work with their existing devices and drivers. This commitment to compatibility remains a high priority for Windows 8 across the whole product.

For USB compatibility testing, the “brute force” approach does not work because there are so many devices, with new ones appearing every day, and many old devices that can no longer be purchased. We needed to develop a smart device test strategy. After analyzing the device statistics, we broke devices into three main categories:

1. Device popularity

When looking at the telemetry sources for the most popular devices, we noticed a pattern. Each device class (keyboard, webcam, printers, storage, etc.) had a handful of prevalent manufacturers with just a few main product lines. After projecting these findings, we could represent 70-80% of devices with a few hundred devices. Testers use the mathematic term "equivalence classing" to describe this work.

2. Chipset

We still had a significant percentage of devices that would go untested if we just settled for popularity data, so we looked deeper at the actual circuit design. Just like humans, devices that appear very different on the outside are pretty similar on the inside. If we could ensure all USB chipsets worked, then it's highly likely the devices that contained them would work as well. A relatively small number of chipset makers are embedded in devices, so we chose to represent their USB IP with development boards.

3. High-profile and challenging devices

It's not often a USB device tops the support call lines. When it does, we want to make sure Windows will work with it going forward.

After 10 years of USB experience, a dozen telemetry sources, and tons of research and brainstorming, we were able to reduce the USB compatibility effort to roughly 1000 unique devices that we regularly test in the Windows labs. We ensure the devices get recognized correctly when connected to PCs, that they sleep and resume appropriately to conserve power, and that they are able to withstand various stress conditions. Our telemetry data indicates that over 90% of devices rely on the 16 class drivers in Windows, but for the more customized devices, we verify that their drivers get seamlessly downloaded from Windows Update whenever possible (the device maker needs to cooperate to support this scenario). With USB 3.0 providing full backwards compatibility, older drivers will still work without any changes.

We also made a heavy investment in building a custom tool - the Microsoft USB Test Tool (MUTT) to simulate a full range of device behaviors that we'd observed over the years. We built the MUTTs from the ground up, in house. Our software test engineers laid out the circuit design with the help of fancy design tools (MS Paint ... seriously). They then developed the firmware and generated new test content to run internally. The MUTT was born – think of it as 1,000 devices on a USB thumb drive. Over time, we shared the MUTT with our hardware partners and they've used it to find and correct problems in their devices before releasing.



MUTT Designer, David Hargrove, with MUTT device

Demonstration video

Perhaps the most important aspect of USB 3.0 is the expectation that customers have of USB: it's just USB3 so it should just work, right? Each and every USB device, low, full, high, and SuperSpeed, has to work in Windows 8. That's our focus while also delivering the most robust and reliable USB stack.

Let's take a look at USB 3.0 in action as it takes on some pretty significant copy tasks and races against USB 2.0.

This HTML5 video isn't supported in your browser.

If you don't see a video here or can't play it, download it here: [High quality MP4](#) | [Low quality MP4](#)

--Dennis

Improving our file management basics: copy, move, rename, and delete

Steven Sinofsky | [2011-08-23T20:00:00+00:00](#)

We wanted to do an early Windows 8 post about one of the most used features, and one we have not improved substantially in a long time. With the increasing amount of local storage measured in terabytes, containing photos (in multiple formats and very large files), music, and video, these common operations are being taxed in new ways. These changes, along with consistent feedback about what we could improve, have inspired us to take a fresh look and redesign these operations. Of course this is just one feature among many, but we wanted to start with something we can all relate to. Alex Simons is a director of program management on our Windows engineering team and authored this post on the redesign of some Windows file management basics. (PS: A lot of folks asked about Building Windows 8 Video #1 -- this is the user experience demo, <http://win8.ms/uxpreview1>. The numbering seems to be confusing so this will be our last numbered video.)—Steven

Copying, moving, renaming, and deleting are far and away the most heavily used features within Windows Explorer, representing 50% of total command usage (based on Windows 7 telemetry data). For Windows 8, we want to make sure that using these core file management commands, which we collectively refer to as “copy jobs,” is a great experience.

We know from telemetry data (which is based on hundreds of millions of individuals opting in to provide anonymous data about product usage), that although 50% of these jobs take less than 10 seconds to complete, many people are also doing much larger jobs, 20% of which take more than 2 minutes to complete. Prior versions of Windows Explorer can handle these kinds of jobs, but Explorer isn’t optimized for high-volume jobs or for executing multiple copy jobs concurrently.

Usability studies confirm what most of us know—there are some pretty cluttered and confusing parts of the Windows 7 copy experience. This is particularly true when people need to deal with files and folders that have the same file names, in what we call file name collisions. Lastly, our telemetry shows that 5.61% of copy jobs fail to complete for a variety of different reasons ranging from network interruptions to people just canceling the operation.

We clearly have an opportunity to make some improvements in the experience of high-volume copying, in dealing with file name collisions, and in assuring the successful completion of copy jobs.

Many of you reading this blog post come at this from a slightly different perspective. Like me, you might already have a third-party copy management tool that already addresses these high-volume scenarios. Our telemetry data shows that the most popular of these add-ons (such as TeraCopy, FastCopy, and Copy Handler) are running on fewer than .45% of Windows 7 PCs. While that might be a large absolute number given the size of the Windows 7 customer base, it still tells us that most people do not have a great tool for high-volume copy jobs.

We aren’t aiming to match the feature sets of these add-ons. We expect that there will be a vibrant market for third-party add-ons for a long time. Our focus is on improving the experience of the person who is doing high-volume copying with Explorer today, who would like more control, more insight into what’s going on while copying, and a cleaner, more streamlined experience.

In Windows 8, we have three main goals for our improvements to the copy experience:

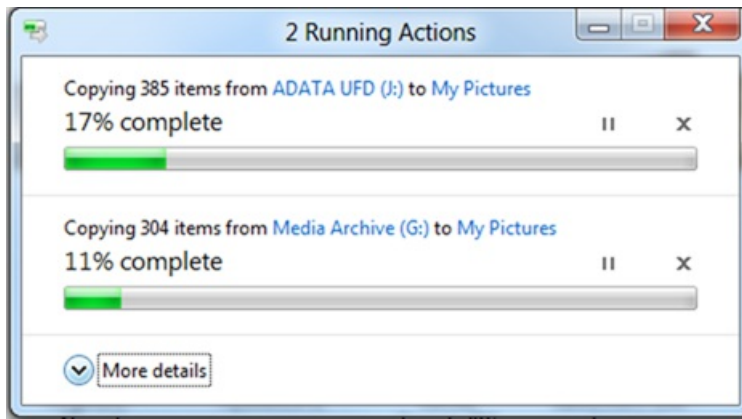
- **One place to manage all copy jobs:** Create one unified experience for managing and monitoring ongoing copy operations.
- **Clear and concise:** Remove distractions and give people the key information they need.
- **User in control:** Put people in control of their copy operations.

Based on these goals, we made four major improvements to the copy experience. Here is a short video demo of these improvements—but keep reading for a more detailed tour.

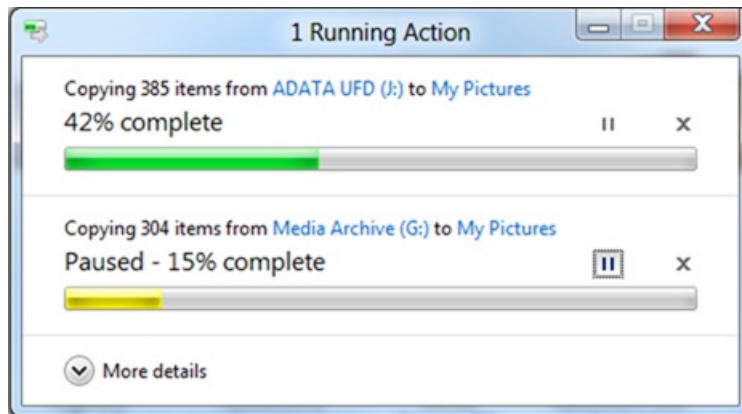
This HTML5 video isn't supported in your browser.

If you don't see a video here or can't play it, download it here: [High quality MP4](#) | [Low quality MP4](#)

First, we’ve consolidated the copy experience. You can now review and control all the Explorer copy jobs currently executing in one combined UI. Windows 8 presents all pending copy jobs in this single dialog, saving you from having to navigate through multiple floating dialogs looking for the one you need.

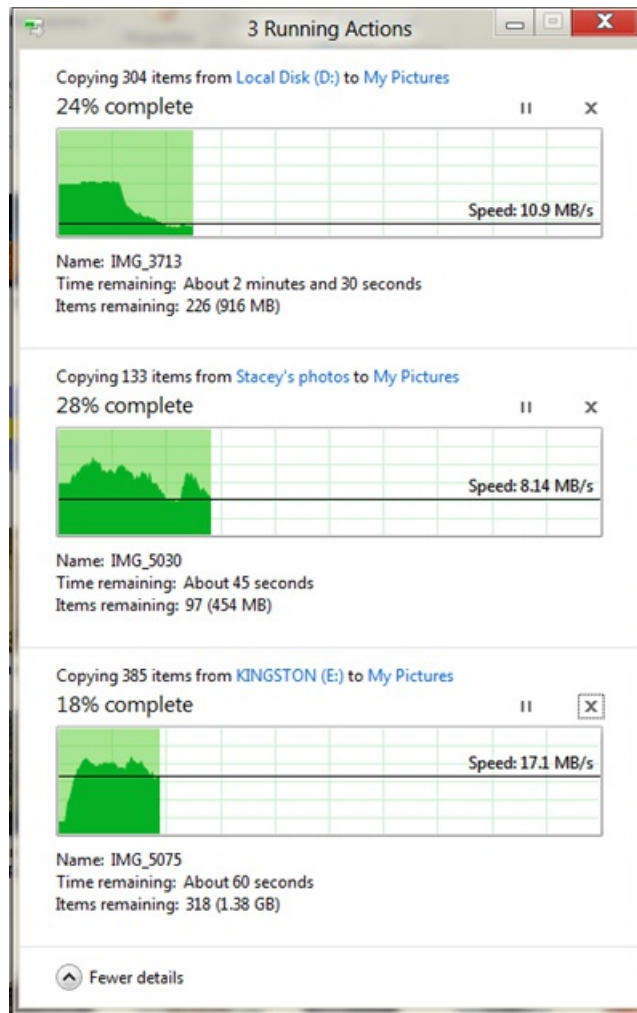


Next, we've added the ability to pause, resume, and stop each copy operation currently underway. This gives you control over which copy jobs will complete first. You can also click any of the source or destination folders while the copy operation is taking place and open up those folders.

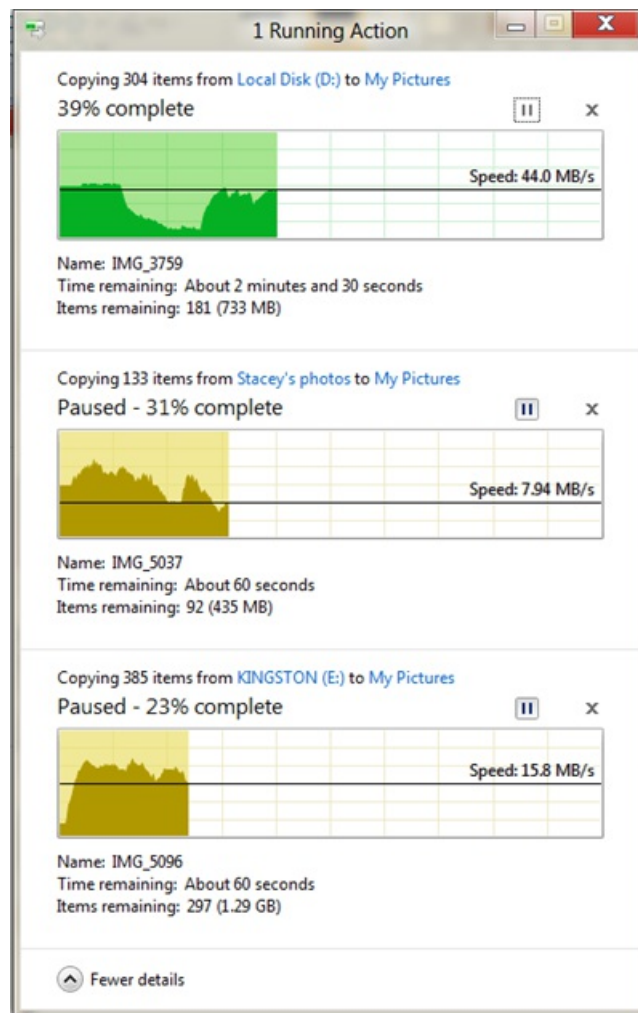


To support this new ability to prioritize and decide, we've added a detailed view with a real-time throughput graph. Now each copy job shows the speed of data transfer, the transfer rate trend, and how much data is left to transfer. While this is not designed for benchmarking, in many cases it can provide a quick and easy way to assess what is going on for a particular job.

Here you can see three copy jobs underway:



And here you can see how the speed of file transfer increases substantially when two of the copy jobs are paused:

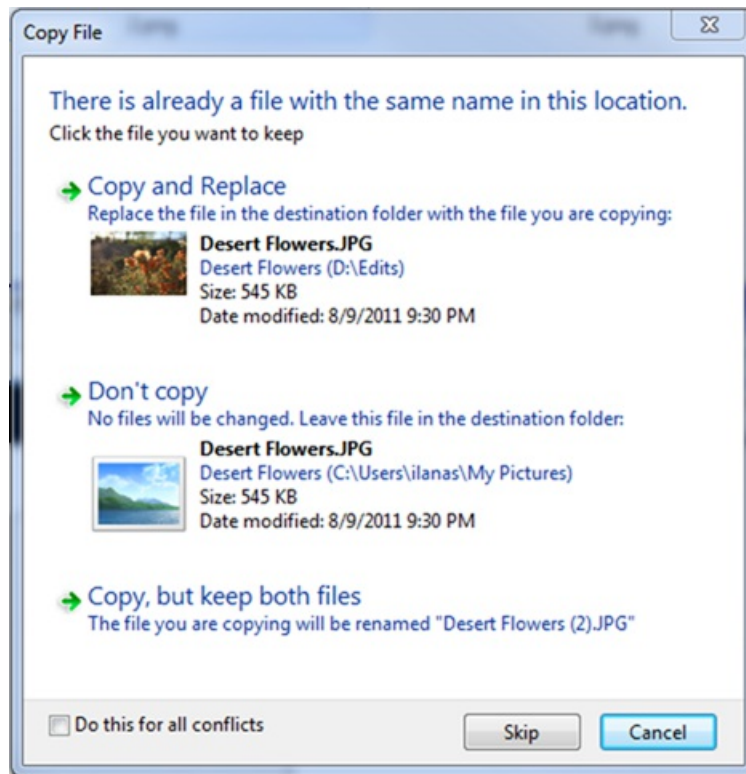


We're anticipating that many of you are going to want to know what we've done to improve the accuracy of the estimated time remaining for a copy to complete. (This has been the source of some pretty funny [jokes](#) over the years).

Estimating the time remaining to complete a copy is nearly impossible to do with any precision because there are many unpredictable and uncontrollable variables involved – for instance, how much network bandwidth will be available for the length of the copy job? Will your anti-virus software spin up and start scanning files? Will another application need to access the hard drive? Will the user start another copy job?

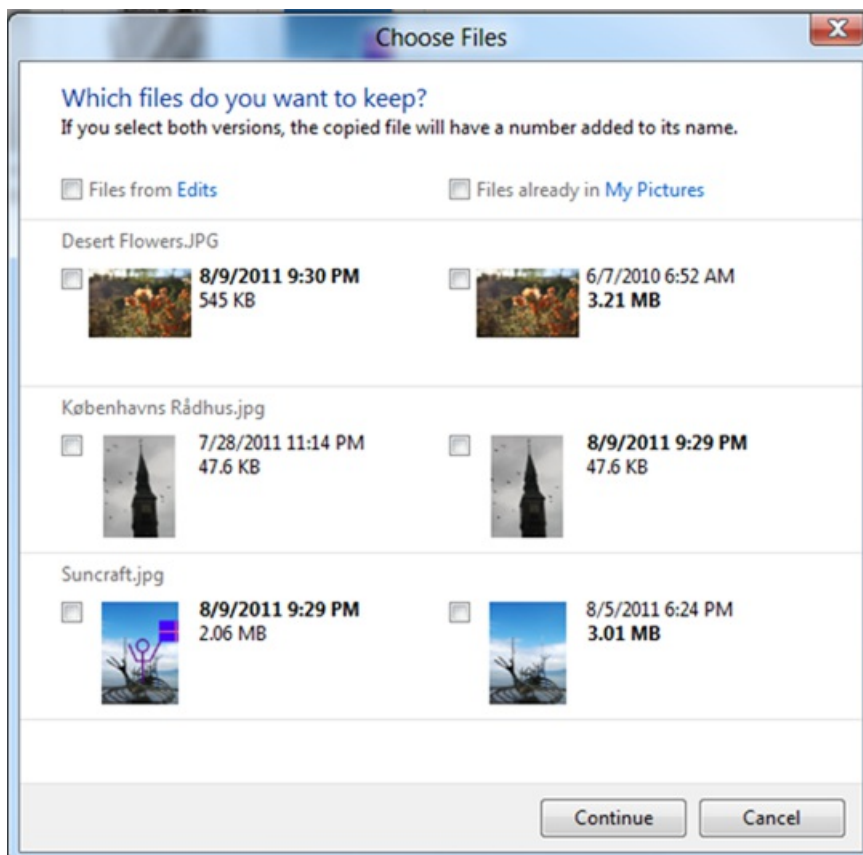
Rather than invest a lot of time coming up with a low confidence estimate that would be only slightly improved over the current one, we focused on presenting the information we were confident about in a useful and compelling way. This makes the most reliable information we have available to you so you can make more informed decisions.

Our last major set of improvements simplify and clean up the experience for resolving file name collisions, which we also refer to as “conflict resolution.” At this point we can admit that the current experience can be rather confusing. People don't know which files are which, and they find it challenging to find the information they need to make a decision.



Windows 7 Conflict Resolution dialog

Our new design is much more clear, concise, and efficient, providing a much more visible and actionable approach to conflict resolution. All the files from the source are on the left. All the files in the target location with file name collisions are on the right. The screen layout is easy to understand and shows you the critical information for all the collisions, front and center in one dialog.



The new Windows 8 Conflict Resolution dialog

If you need to know even more about the conflicting files, you can hover over the thumbnail image to see the file path or double-click it to open it from here.

Finally, in addition to these big improvements, we've also done a thorough scrub and removed many of the confirmation dialogs that you've told us

are annoying or feel redundant (i.e. “are you sure you want to move this file to the recycle bin?” or “are you sure you want to merge these folders?”) to create a quieter, less distracting experience.

All of this adds up to building a significantly improved copy experience, one that is unified, concise, and clear, and which puts you in control of your experience.

--Alex Simons

Designing the Windows 8 file name collision experience

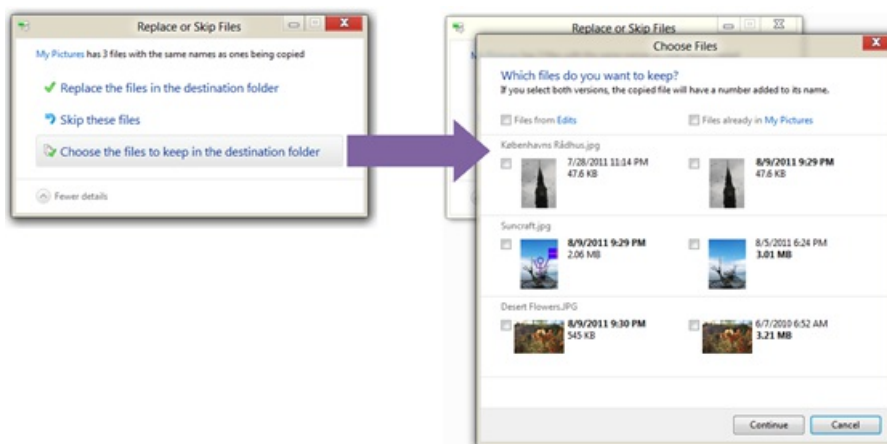
Steven Sinofsky | [2011-08-26T11:20:00+00:00](#)

Thank you so much for all the comments about our work to improve file management basics. We've been overwhelmed by the dialog—there's a huge amount of excitement for the changes we're making and a ton of energy around this topic. That's what makes working on Windows 8 so much fun. While there were comments and suggestions around many parts of what we talked about, by far the most back and forth (expressing all sides of the issue for sure) came from the discussion of the file name collision dialog (one dialog!). We thought it would be great to dig up the design archives from the development cycle and show you some of what we considered and how we got to where we are. Down the road we will of course circle back and talk about any changes we might make, but we thought spending some energy looking at our design path would be a useful effort. This post was authored by a set of folks who worked on the feature (they all worked on other parts of Windows 8 as well)-- Ben Truelove (designer), Matt Duignan (UX researcher), Jon Class and Ilana Smith (program managers) . --Steven

Our previous post about the [new copy experience](#) in Windows 8 generated a lot of questions and comments about the new “Choose Files” dialog for resolving file name collisions. Based on the level of interest, we thought it would be fun to share some of the design iterations and our usability testing that led us to this design.

In the implemented design, there are two levels of control when acting on file name collisions (or “conflicts”).

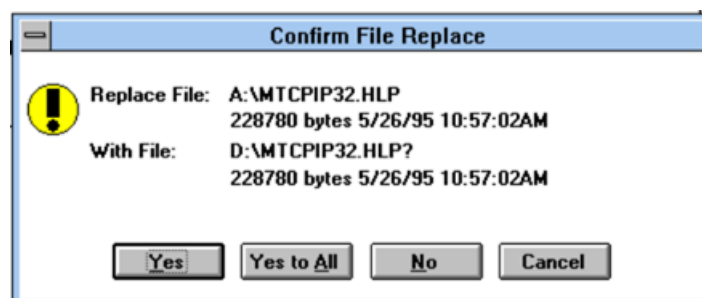
- The primary experience is a simplified, one-click, bulk management of all conflicts, offering “Replace all” or “Skip all.” We call this the “Simple Conflict Resolution dialog.”
- There is also an option to enter the secondary experience which offers more information and more fine-grained control. This is the “Detailed Conflict Resolution dialog.”



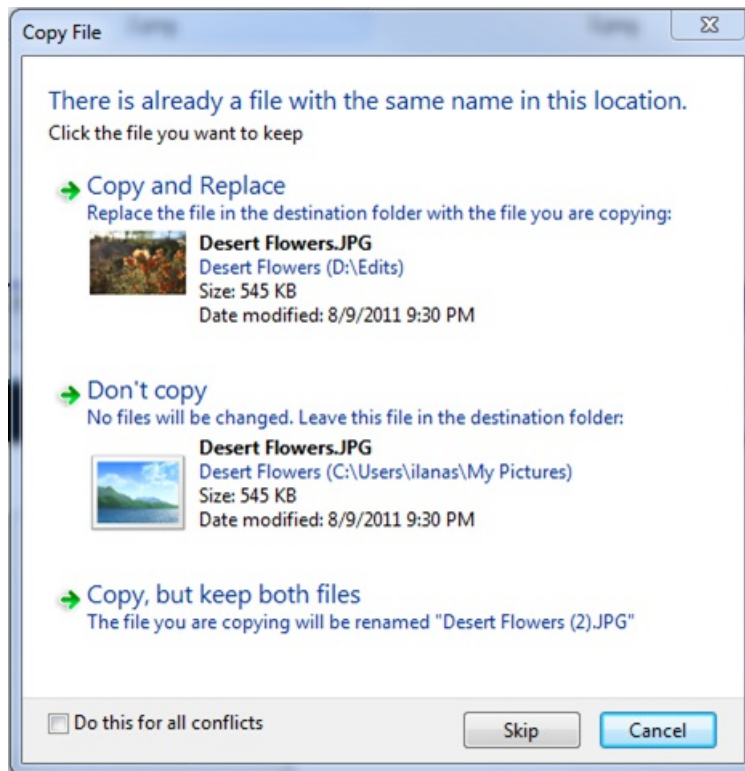
Windows 7 and before

Resolving file name collisions is an inherently tricky task, as it involves making a meaningful choice between two very similar things.

Here's how we did this back in Windows 3.1:



We'd certainly made some progress by the time we got to this in Windows 7:



In Windows 7, there's a lot of information to aid the choice, and more options about what action to take. For Windows 8, we thought we could improve this even further, so it's easier for you to make the right decisions more efficiently, and get your file transfer tasks completed faster. As we mentioned, the feedback and support calls on the existing dialog were clear—folks were having a hard time finding the information needed to make an informed choice in a fairly complex dialog. Even with the amount of work we do, sometimes it takes quite a while to surface something that isn't optimal. Keep in mind millions of people used pre-release Windows 7 and this was not a big topic of discussion on our forums (not to say that it didn't come up, but it was not a broadly raised topic).

Improving on the Windows 7 experience

First, we looked at ways to keep the experience basically the same, but to incrementally improve it by optimizing for the key information that is necessary for the decision.



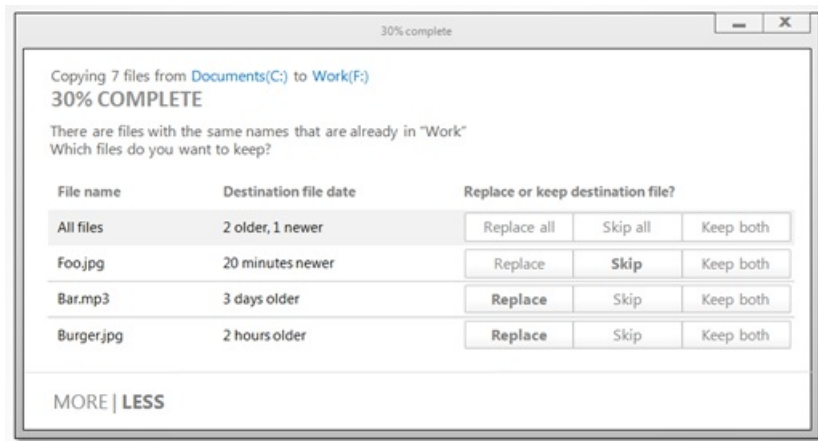
These designs introduced some concepts that really stuck around:

- Getting rid of unnecessary labels (like “Date modified:”) and obvious explanatory text enabled us to present the important details at a glance.
- Metadata adjectives were emphasized. Rather than requiring users to compare values like file size, using words like “Larger” gave users the right summary.
- Smart defaults were pre-selected, reducing the work for users.

Fast and fluid: better bulk management of conflicts

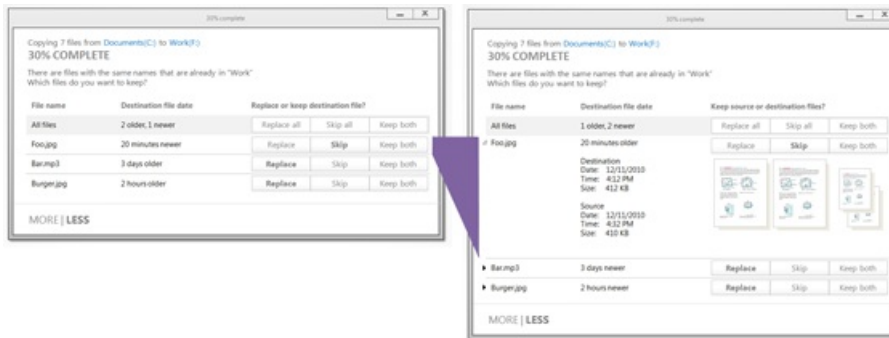
In Windows 8, we want you to be able to get stuff done more quickly and efficiently—“fast and fluid” are key design words around Windows 8 for all of our designs (for touch, mouse/keyboard or both together). The next major design iteration looked at ways that we could follow on from the cohesive copy progress experience, bring queued-up conflicts together into a single dialog, and provide you with the ability to manage them in a more streamlined way.

The idea of optimizing for the “Replace all” or “Skip all” choice was introduced. Most of the time, you know exactly what you're copying and why it is conflicting, and you can make a simple choice about what action to take.



For cases where you need more information or finer-grained control, we decided to disclose information in “tiers” of greater detail.

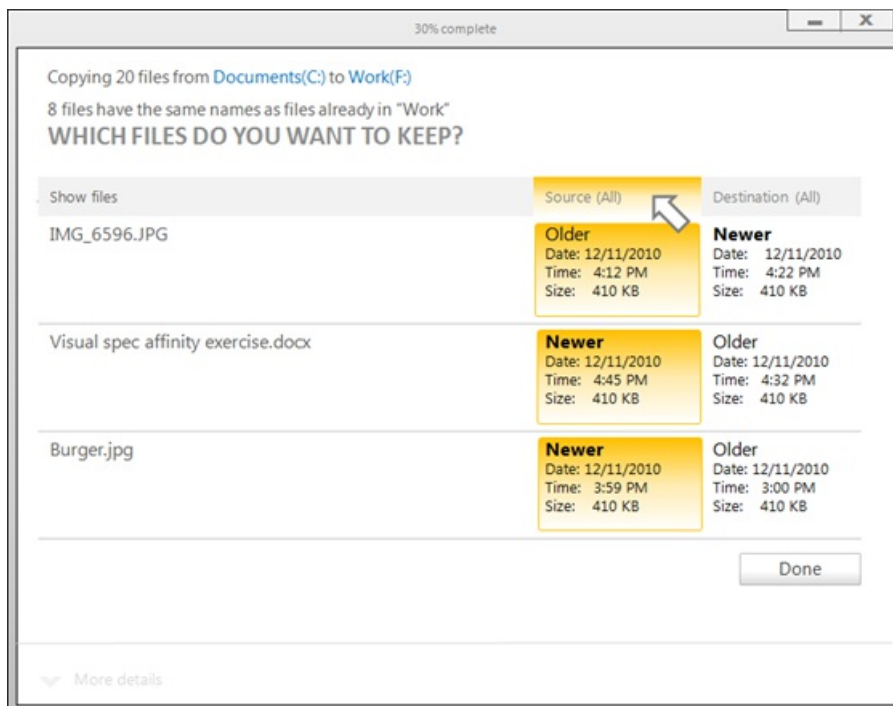
We started with two tiers:



Then we tried three-tiers:

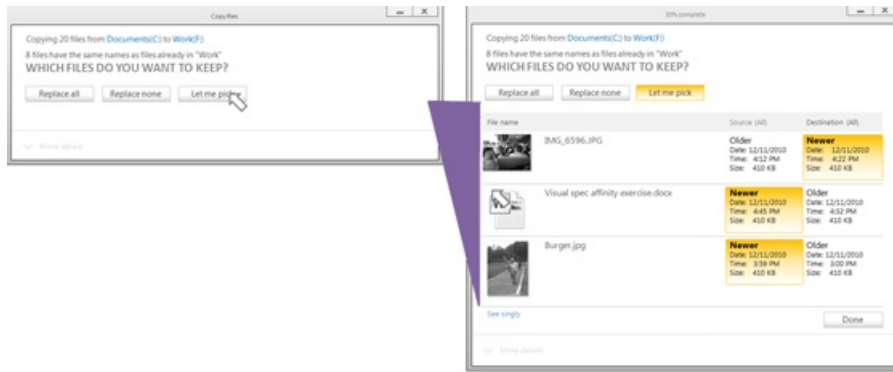


And ended up back at one-tier:



This design offers many positive attributes. It provides a lot of information. Since clicking on the headers selects everything in a column, it provides real power for managing conflicts. But it was a very complex piece of UI to be presenting as the initial experience.

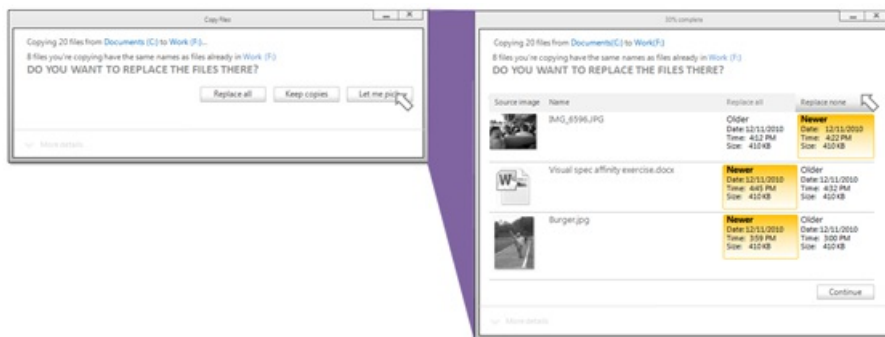
Instead, we combined the best of these options into the following:



Simple and detailed conflict resolution

It was clear that this design was heading toward a balanced combination of simplicity and power that would suit user patterns.

Unfortunately, we identified a real challenge with this design: when you select “Let me pick,” the result is confusing and overly complex because the simpler and advanced options are both available. This led us to a design where the “Simple Conflict Resolution dialog” and the “Detailed Conflict Resolution dialog” were discrete experiences.

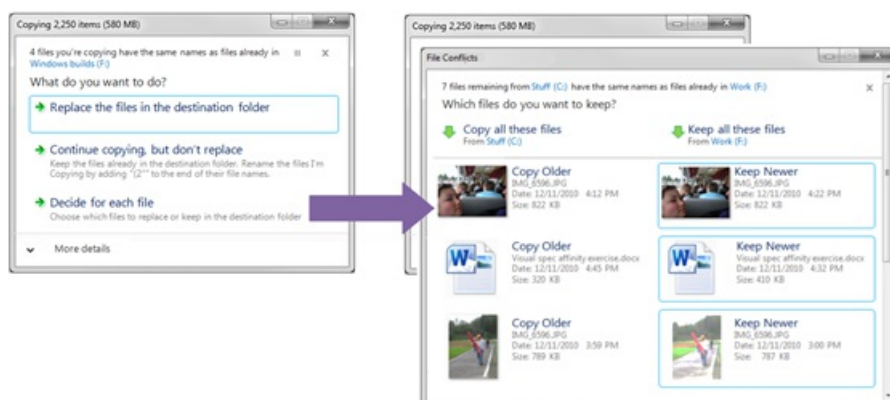


With this decision, our basic structure was in place.

Refinements

In preparation for testing with users, we iterated on the design.

- We cleared up the confusion caused by a single thumbnail.
- We made the source and destination (and their columns) more apparent.
- Our User Assistance team (the experts in authoring text we use in the product, assistance, and the web) helped us out with better text.



It's interesting to see the similarities between the Simple Conflict Resolution dialog and some of the earliest designs for dealing with single file conflicts. It's also interesting how similar they both are to the final design for the dialog.

First round of usability research

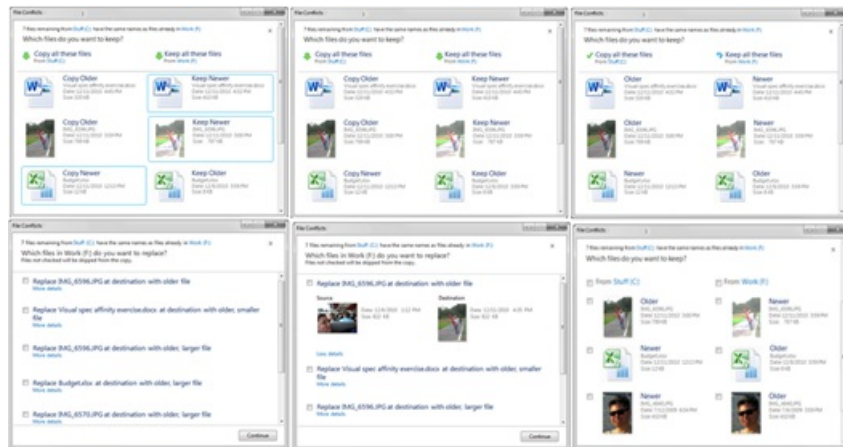
In our usability tests our researchers find a diverse set of subjects who don't work at Microsoft and represent a range of different skill levels and experiences. We show them the software and ask them to complete a set of tasks. By listening as they describe their thought process, using eye-tracking to watch how they see the UI and measuring successful task completion, we gain valuable insights into what works (or not) about a

design.

It is super important to understand that usability tests are one tool we use. Anyone who has ever used this tool knows that you have to be both an expert in the domain and also an expert at designing tests themselves as observer bias and test construction can easily lead you to a false sense of security or efforts to optimize an inherently flawed solution. To help us in that regard, our tests are designed by objective researchers who understand the limits of what can be tested and also make sure that the conclusions drawn from the test match what the test was meant to measure. Ultimately, design choices require the use of many different inputs both qualitative and quantitative as well as experience and intuition.

We knew that we'd learn a lot in our first round of usability tests and make many changes, so we used the [RITE method](#) as our protocol. Most usability studies test the same UI with all users, but with RITE, we make changes continuously between participants, based on what we learn. (We were testing with PowerPoint slides at this point, so change was cheap.)

We didn't end up needing to make many changes to the Simple Conflict Resolution dialog as it tested well, but we tested lots of different things for the Detailed Conflict Resolution dialog:



Our key lessons:

- Check boxes are necessary. As much as we preferred the cleaner no-checkbox look, it simply didn't test well. Users didn't know what to do when presented with the UI. Check boxes were much more effective in providing appropriate cues for selection. We made sure to retain a large click-target area, so users can click on the check box, thumbnail, or the text to select a file.

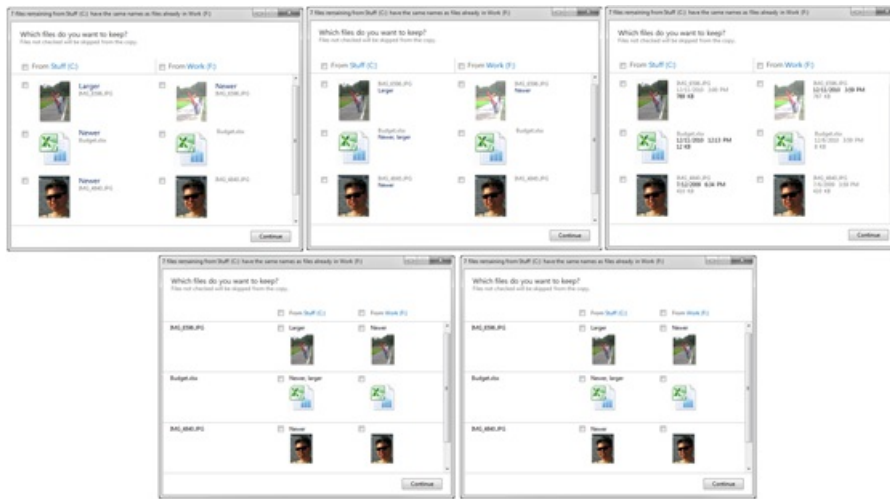


Click target area for file selection

- Mixing the adjectives (e.g. "newer," "larger") and the metadata was confusing. Users interpreted them as two different concepts. The adjectives were particularly problematic – people thought they were titles, or described the file location (for example, "older" was interpreted to mean the files in the destination because they were present prior to the copy.)
- Columns needed to be more distinct. At first glance, it looked like the Tiles view in Explorer, rather than a table.

More refinement

There was no simple solution to the adjective and column issues, so that led to more design explorations:



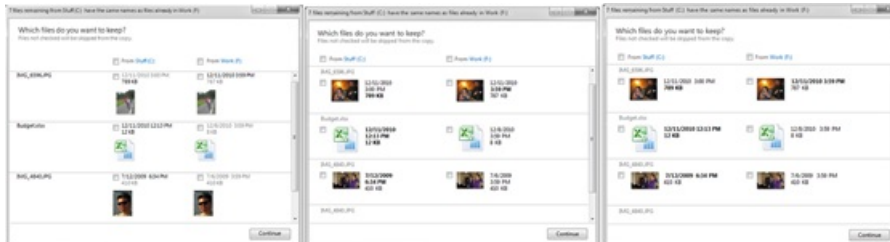
We really struggled with how best to define the hierarchy and importance of source/destination versus conflict rows. We tried vertical lines, which separated the source and destination too much. We ultimately landed on horizontal lines, combined with the file name as a header, to give the most prominence to the distinction between conflicts. The check boxes aided in distinguishing a choice between source and destination without interfering with this distinction.

Some of our earliest ideas were discarded at this point in the process:

- No default choices. With conflicts scrolling off the page, defaults posed too much of a risk of data loss. No selection in a row results in the copy of that file being skipped, so nothing is lost.
- No adjectives. We liked “Newer” and “Larger,” but they added confusion and users valued the concrete data. Instead, to help users make the choice, we chose a more subtle suggestion – the newer and larger metadata values are **bold** in the UI. This has proven to be surprisingly effective, without introducing new concepts or adding clutter.

More usability research

In our next round of usability tests, we were heading toward the final design, and tested fewer alternatives:



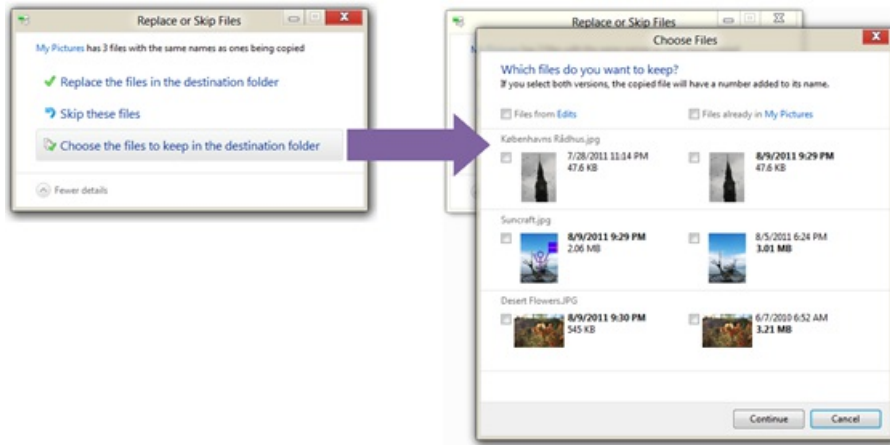
The third option was the clear winner. The two-column view is the most efficient use of space and moves the check boxes close to the question. Date and time need to be on the same line because these are primarily a single value.

The Detailed Conflict Resolution dialog also offers the following features to help when even more information is required to make the decision:

- Double-clicking the thumbnail opens the file.
- Right-clicking the thumbnail opens the standard context menu.
- The blue Source and Destination text are clickable, and open those locations in Explorer.
- Hovering on the thumbnail or link shows a tooltip with the full file path.

Continuing to iterate

We’ve continued to conduct more studies and make minor changes since the initial research, but the core design has remained basically the same. It has been very encouraging to witness the ease with which users complete usability tasks. Resolving file name collisions is a tricky problem, but users are efficient and successful.



Check out the video in our previous post on [file management basics](#) to see this design in action.

We love feedback and want to use it to make the best design we can, so we've been carefully reading all your comments, and look forward to you working with it in practice.

-- Ben Truelove, Matt Duignan, Jon Class, and Ilana Smith

(If you missed them, several of our team members made comments on the previous post that addressed some of the questions raised: [Alex](#), [Matt](#), [Jordi](#), [Jon](#).)

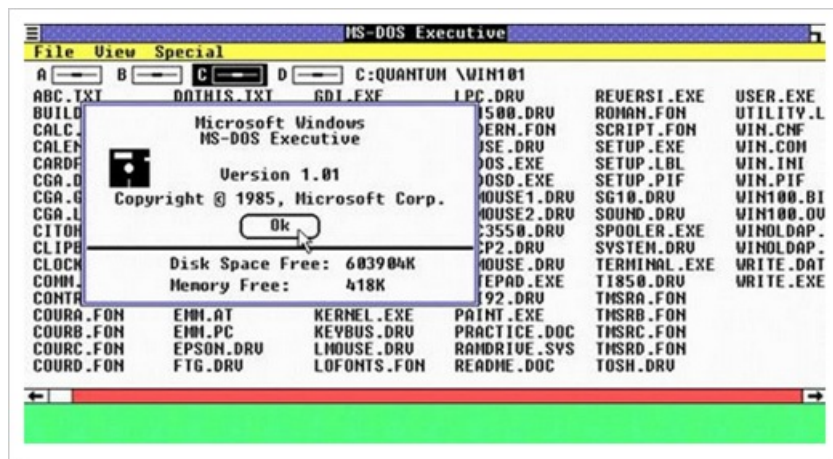
Improvements in Windows Explorer

Steven Sinofsky | [2011-08-29T08:15:00+00:00](#)

Windows Explorer is a foundation of the user experience of the Windows desktop and has undergone several design changes over the years, but has not seen a substantial change in quite some time. Windows 8 is about reimagining Windows, so we took on the challenge to improve the most widely used desktop tool (except maybe for Solitaire) in Windows. Alex Simons on the program management team authored this post with a detailed look at the evolution of Explorer and the major improvements to its interface and functionality for Windows 8. Judging by the passion on file operations and user interface design, we know this is an important subject so we expect a pretty engaged dialog on the topic. We put this in one lengthy post, will watch the comments and dialog, and down the road we'll continue the discussion.

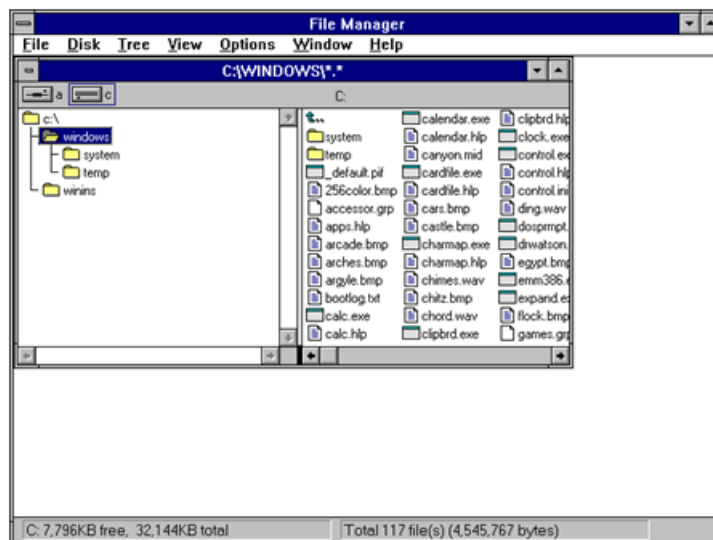
-- Steven

It's exciting to have this opportunity to share the improvements we're making to the file management capabilities of Windows Explorer. Explorer is one of the most venerable parts of Windows with a heritage you can trace back to the "MS-DOS Executive" in Windows 1.0!

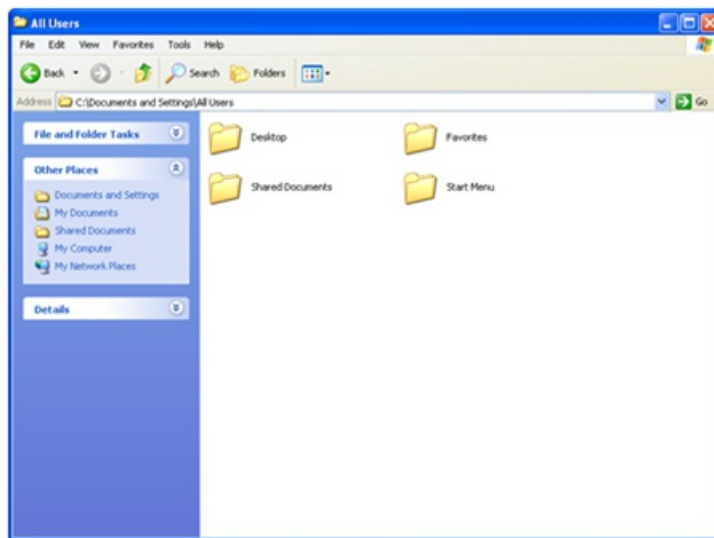


MS-DOS Executive in Windows 1.0

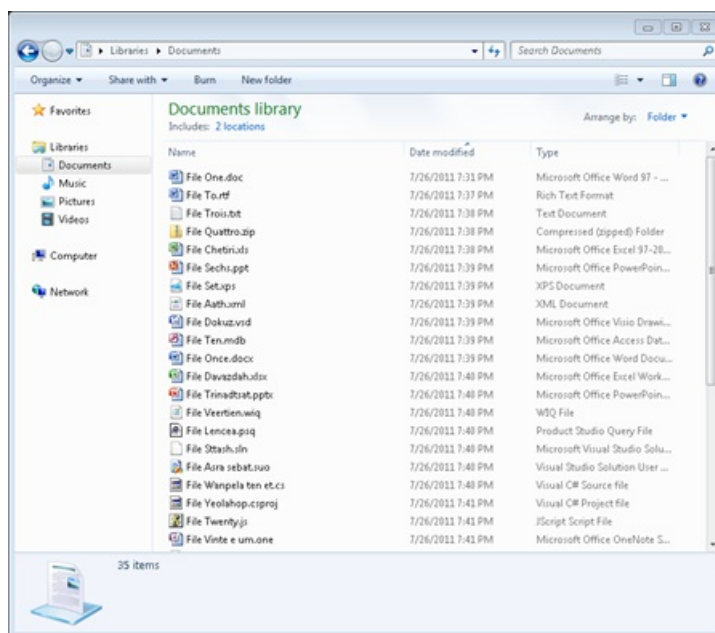
Over the years, Explorer and its forerunners have gone through several major iterations:



File Manager in Windows 3.1



Explorer in Windows XP



Explorer in Windows 7

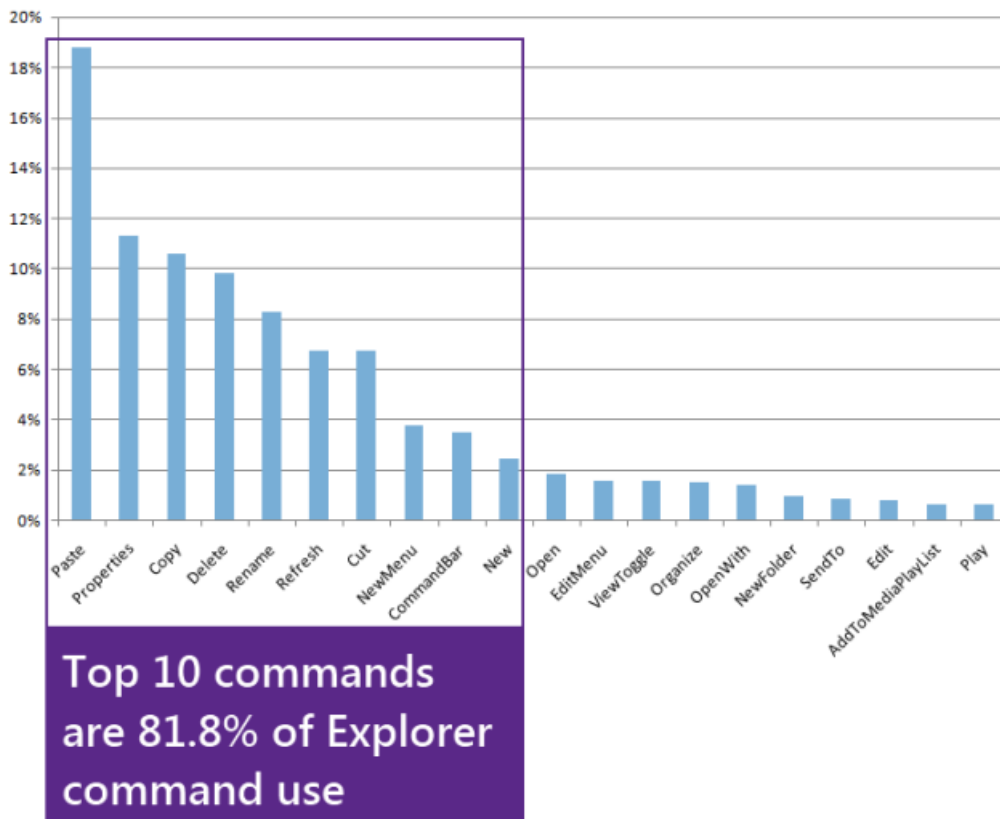
It's a bit daunting but also pretty exciting to have the opportunity to revisit and rethink this cornerstone of our product. Many of you who are reading this (and most of us on the development team) are among the most extreme "power users" of the file management tools in Explorer and likely start from a different perspective than the broad base of customers. As we approach the work to improve file management in Windows, we do so knowing many of you have long ago "given up" on Explorer and are using some of the wide variety of add-ons or alternatives.

As we mentioned in our post on improvements in the copy function, telemetry data indicates these add-ons and alternatives are mostly used by us power-users and we represent a small but influential group of people. The most popular add-ons and replacements (programs like TeraCopy, FastCopy, xplorer² & QTabBar) are installed (note that does not mean used) on about 0.45% of PC's. Our goal is to improve the usage experience for a majority of customers while recognizing that, with such a long history and variety of depth usage, we cannot possibly provide all of the power everyone might want. We expect that there will be a vibrant third-party toolset for some time to come. Windows 8 is an opportunity to substantially improve the experience for everyone.

How Explorer is used today

Over the years, Explorer has grown to support a number of different scenarios, many unrelated to file management – launching programs, viewing photos, playing videos, and playing music, to name just a few. We wanted to know which of these capabilities customers were really using. Using telemetry data, we were able to answer the question of how the broadest set of customers use Explorer in aggregate. As a reminder, the telemetry data is opt-in, anonymous, and private, but it does represent hundreds of millions of sessions from all customer types.

Command usage in Windows Explorer

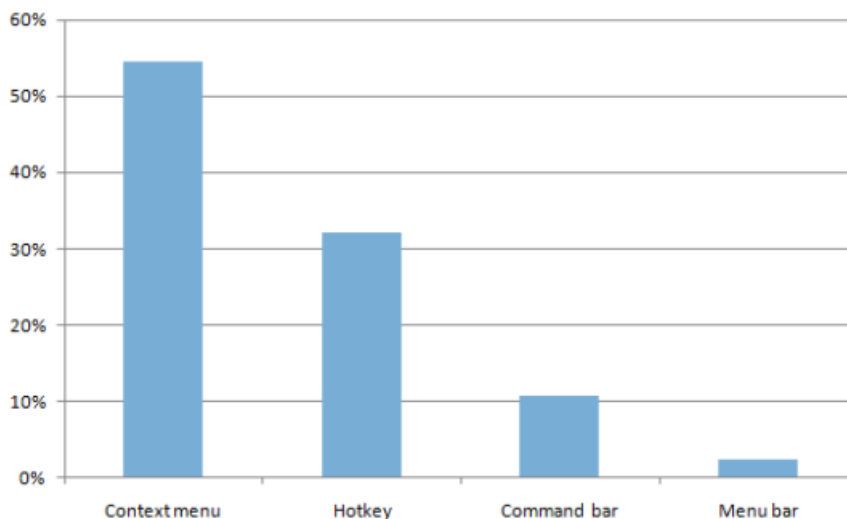


This data is pretty interesting. First it shows that even though there are over 200 commands in Explorer, customers use a small number of them with any real frequency: the top 10 commands represent 81.8% of total usage. Additionally it shows us that people overwhelmingly use Explorer for core file management tasks - the top 7 commands (72.2% of usage) are all for managing/manipulating files.

This data represents the total usage of Explorer and includes cases where a person has a third-party add-on installed that uses one of our built-in commands (i.e. "play," "open," "edit," "email," etc.) A good example would be that a customer might have a third-party music app installed, which is the default player for all their music formats. The command usage of this third-party add-in from within Explorer is included in the data above. There are a class of add-ons that add their own custom commands (i.e. "rotate") and we don't get telemetry data for those, though we do know how often they are installed and get invoked (<2% of user sessions). This data is pretty solid and given the hundreds of millions of data points, it gives us a very clear picture of average usage across the population as a whole, and also of the spectrum of usage patterns (depth and breadth, frequency, etc.).

We also wanted to know how people most frequently invoke commands in Explorer.

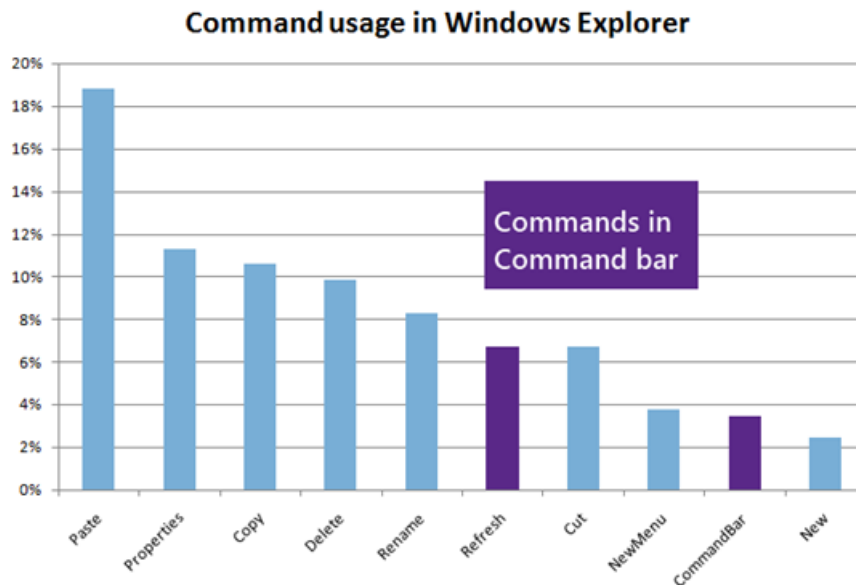
Command entrypoint usage



The telemetry data here shows that 54.5% of commands are invoked using a right-click context menu, and another 32.2% are invoked using keyboard shortcuts ("Hotkey" above) while only 10.9% come from the Command bar, the most visible UI element in Explorer in Windows 7 and

Vista. With greater than 85% of command usage being invoked using a method other than the primary UI, there was clearly an opportunity to improve the Explorer user experience to make it more effective—more visible and uniformly accessible. While context menus are convenient, the features in them can be overlooked if you don't condition yourself to "search" via a context menu for the feature (a well-known challenge with the mechanism).

We also did an analysis of which of the commands that customers used were available in the Command bar:



Only 2 of the top 10 commands customers invoke in Explorer are available in the Command bar, the main UI element for invoking commands. This further reinforced our thinking that there was a big opportunity here to improve Explorer by making common commands more readily available. A clear user interface design principle is that frequently used commands should be easy to get to—clearly we had not yet accomplished that with existing designs.

Next, we turned to customers and community feedback. Customers have a lot of suggestions for how they'd like to see Explorer evolve. Many of these suggestions are for things that after-market add-ons like TeraCopy, QTTabBar, DMEXBar, & StExBar, or Explorer replacements like xplorer², XYplorer or FreeCommander already offer.

The biggest category of feedback was requests to bring back features from Windows XP that were removed in Windows Vista, especially things like bringing back the "Up" button from Windows XP, adding cut, copy, & paste back into the top-level UI, and for providing a more customizable command surface. Also frequently requested is the need for more keyboard shortcuts. As you'll read below, we've addressed many of these top requests in the redesigned Explorer. Each of these "removed" commands has a long history rooted in the changes to the Windows architecture and/or design philosophy.

Goals of the new Windows Explorer

We set out to accomplish three main goals with this new version of Explorer.

1. **Optimize Explorer for file management tasks.** Return Explorer to its roots as an efficient file manager and expose some hidden gems, those file management commands already in Explorer that many customers might not even know exist.
2. **Create a streamlined command experience.** Put the most used commands in the most prominent parts of the UI so they are easy to find, in places that make sense and are reliable. Organize the commands in predictable places and logical groupings according to context, and present relevant information right where you need it.
3. **Respect Explorer's heritage.** Maintain the power and richness of Explorer and bring back the most relevant and requested features from the Windows XP era when the current architecture and security model of Windows permits.

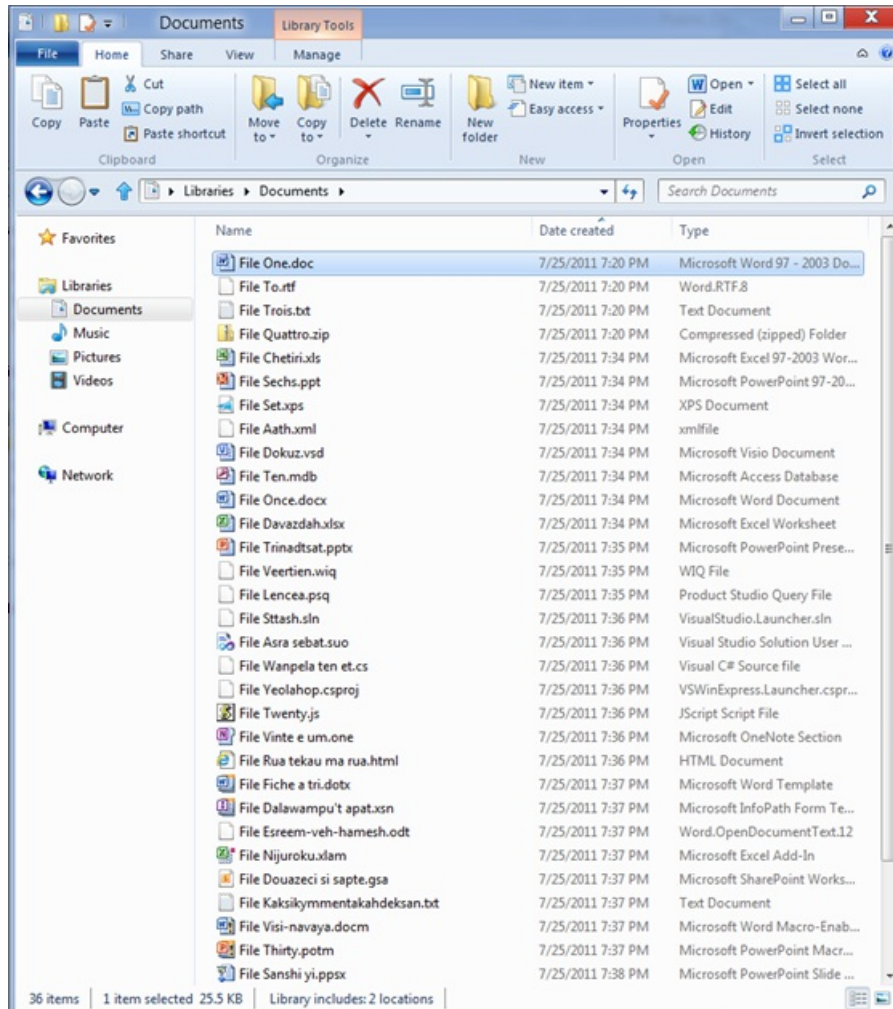
We evaluated several different UI command affordances including expanded versions of the Vista/Windows 7 command bar, Windows 95/Windows XP style toolbars and menus, several entirely new UI approaches, and the Office style ribbon. Of these, the ribbon approach offered benefits in line with our goals:

- Provides the ability to put the most important commands in very prominent, front and center locations.
- Makes it easy to find commands predictably and reliably. Every important file management command could be given a home in the ribbon, and customers would always know where to look for them.
- Exposes a large set of commands (~200) in one easy and consistent experience and organizes commands into scenario-focused groups without the use of nested menus, popups, dialogs, and right-click menus.
- Aids command identification with support for grouping, a variety of button sizes and icons, and aids deeper investigation with live previews and expanded tooltips.
- Takes a similar approach to Office, Microsoft Paint, and Windows Live Essentials, which means that many of our customers will be familiar

with the model and not have a lot to learn.

- Provides a consistent, reliable UI that doesn't degrade over time like traditional toolbar and menu-based user interfaces do. [See Jensen's earlier blog on this topic](#) from the development of the ribbon.

These strengths fit well with our three goals – the ribbon would allow us to create an optimized file manager where commands would have reliable, logical locations in a streamlined experience. The flexibility of the ribbon with many icon options, tabs, flexible layout and groupings also ensured that we could respect Explorer's heritage. We could present a rich set of commands without removing access to previously top-level commands, something we knew was really important to our customers. As it so happens, while not primarily a touch interface, the ribbon also provides a much more reliable and usable touch-only interface than pull-down menus and context menus (we'll have lots more to say on the topic of touch, of course—as a reminder, [check out this Windows 8 video](#)--we definitely know there is a lot of interest but also want to make clear that we know how important keyboard and mouse scenarios are to power-user scenarios of file management).



Explorer in "Windows 8"

We knew that using a ribbon for Explorer would likely be met with skepticism by a set of power-users (like me), but there are clear benefits in ways that the ribbon:

- Exposes hidden features that they already use but which require third party add-ons to use in the Explorer UI today.
- Provides keyboard shortcuts for every command in the ribbon, something many people have been asking for.
- Provides UI customization with the quick access toolbar, taking us back to a customization level that is basically equivalent to Windows XP.

We also knew that, similar to when we added the ribbon into Office, there would be concerns about reduced screen real estate. We worked hard to mitigate this issue, and I'll tell you what we did here a little later in this blog post.

Finally, there are quite a few third-party add-ons that some of our more advanced customers use with Explorer today. These add-ons will continue to work in the right-click context menus in Windows 8, which is by far the most common access point for experienced customers running these add-ins (where discovery and occasional usage are not the primary design points). However, add-ins will not be able to plug into the ribbon UI. This was a difficult engineering choice for us and we expect that many of you will read this and suggest we add the capability--of course if we could get it right this time around we would have done that. A big part of this blog is sharing these choices--tradeoffs--between new features and adding everything we can dream up and finishing. We also think the customization we provide and the improvements are worthwhile *this time around*.

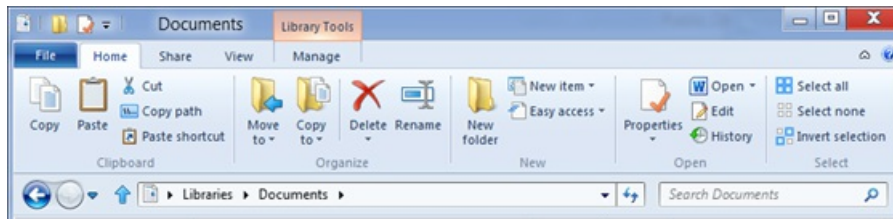
In a related note, one of the most common requests we get in any redesign is to continue to provide the old user interface along with the new. Sometimes this is suggested as a "transitional" benefit, and other times as a "compatibility mode." We've learned over many product cycles that the

work to provide this significantly impacts the evolution of the product. The most immediate challenge is that any new commands added to the ribbon then need to be added in the old UI, even if there is no logical place for them. And of course as the new UI evolves, backward compatibility proves doubly challenging. Each time we change we double the number of "old" experiences we carry forward. Our hope is that those who maintain software understand that these are tradeoffs we make in a thoughtful and deliberate manner, and are not meant to be forceful or painful in any way. We are fully aware of the responsibility that comes from changing an interface used by so many people.

A ribbon gave us a lot of layout options and we explored a number of different approaches to tabs and grouping. We decided to go with three main tabs: Home, Share, and View, plus a File menu and a variety of contextual tabs.

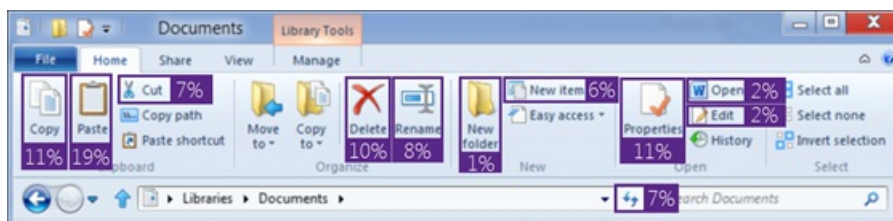
The new ribbon

The Home tab is focused on the core file management tasks, and we've put all the major file management commands there in prominent locations: Copy, Paste, Delete, Rename, Cut, and Properties. We've also given new prominence to two popular heritage features, Move to and Copy to, along with exposing a hidden gem, Copy path, which is really useful when you need to paste a file path into a file dialog, or when you want to email someone a link to a file on a server.



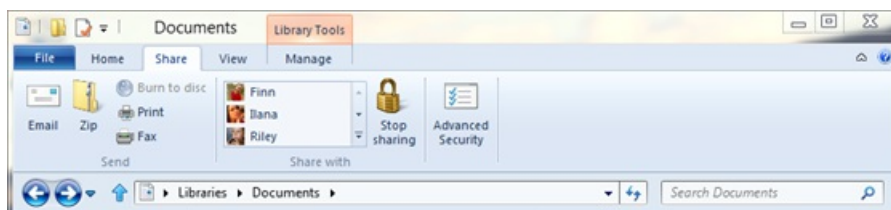
The new Home tab

The Home tab is the heart of our new, much more streamlined Explorer experience. The commands that make up 84% of what customers do in Explorer are now all available on this one tab:



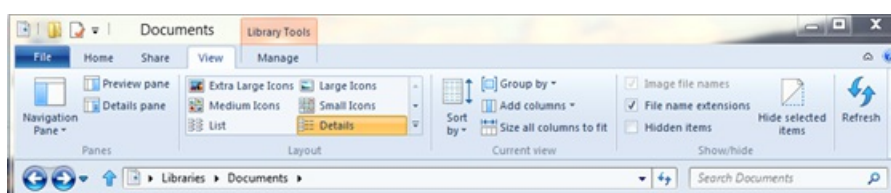
Overlay showing Command usage % by button on the new Home tab

The Share tab is for sharing files by typical methods like zipping them up and emailing them to a friend, or burning them to optical media. Or you can quickly share files with other people in your home group or your network domain. It also provides one-click access to the ACLs for the currently highlighted file.



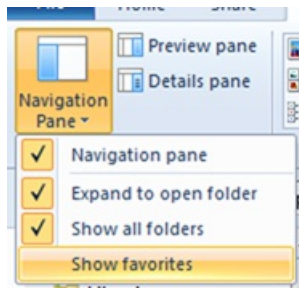
The new Share tab

The View tab provides access to options for view customization. We've enabled one-click access for turning on/off the Navigation pane, Preview pane, and Details pane, a live preview gallery for the different icon display sizes, quick access to sorting and grouping by column, the ability to quickly add columns, plus easy access to three hidden features: show file name extensions, show hidden items, and hide selected items.



The View tab

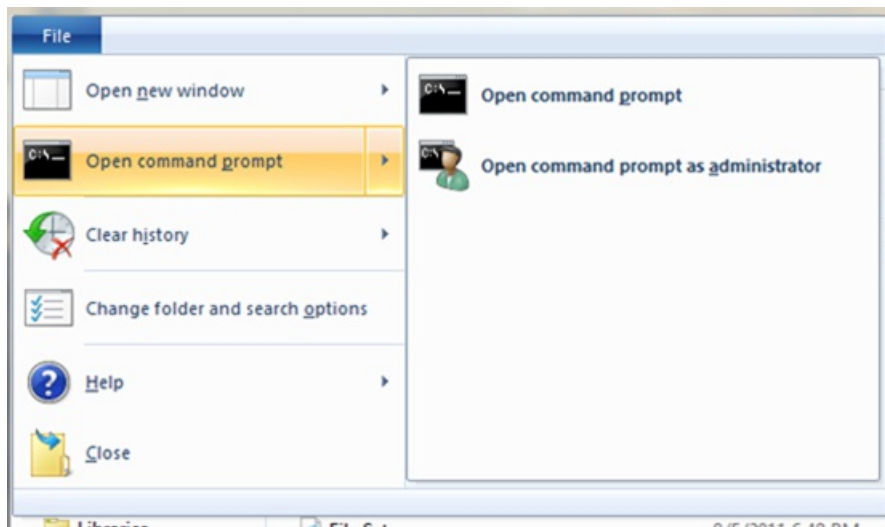
The customization options for the Navigation pane are also much easier to access – in the drop-down menu, you get one-click access to them, including a new option to show or hide favorites.



Navigation pane options

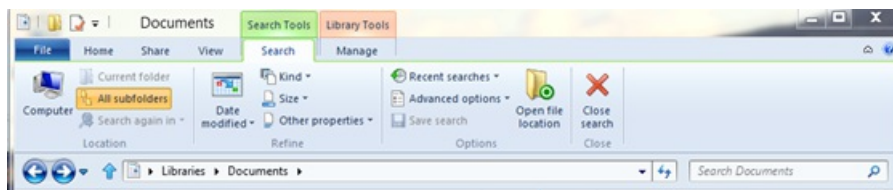
The file menu and other tools

The file menu lets you quickly open new Explorer windows, access your shortcuts, and change folder and search options. It also includes a hidden feature that we love, Open command prompt, and a really useful new command, Open command prompt as administrator, both of which launch a command prompt with the path set to the currently selected folder.



File menu

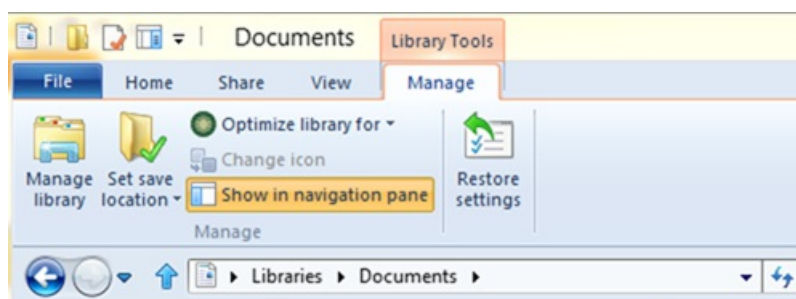
We've provided a variety of contextual tabs that activate in the context of specific files and folders, and for tasks like searching, managing libraries, viewing pictures, and playing music. One of the best examples is the new Search Tools contextual tab which launches when you click in the search box.



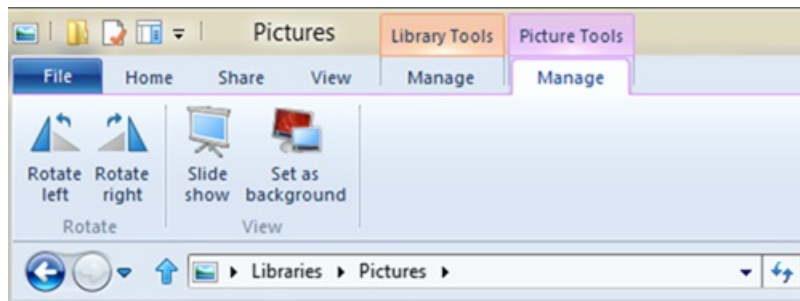
Search tab

The Search tab surfaces a bunch of hidden gems that most people are not aware of, but that could solve some common problems for them. You can quickly adjust the scope of any search, filter by common date ranges, file type, file size, and other properties like the author or name. Then you can save these searches for future use.

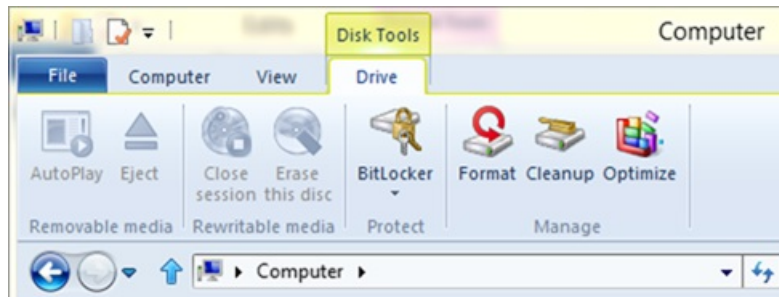
Here are examples of some of the other Explorer context tabs:



Library Tools



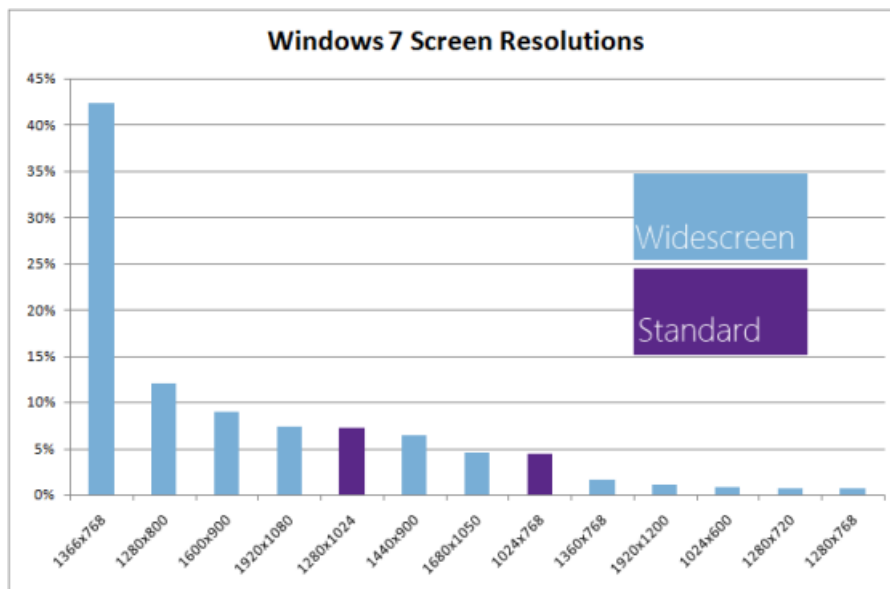
Picture Tools



Disk Tools

Designing for a wider screen

When considering the ribbon UI, we knew we had to be conscious of one of the primary customer concerns we hear about: screen real estate. As we looked at ways to mitigate this issue, we dug up some more telemetry data for Windows 7:



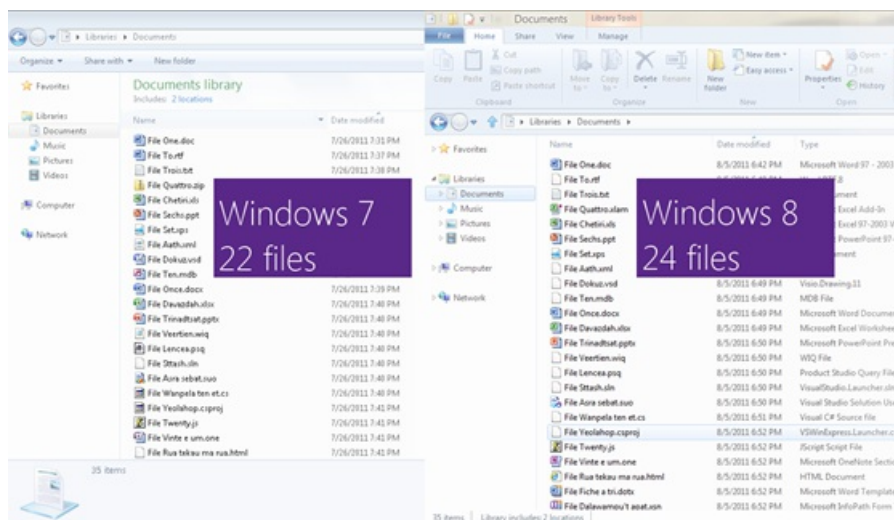
As this data shows, widescreen formats (those with a resolution ratio > 1.3) have become the standard. Of the top 20 screen resolutions, 17 of them are widescreen formats and they account for 83% of the total Windows 7 PC base. This should make sense to everyone because the majority of PCs are laptops and almost all laptops are wide screen. The two common standard resolutions are almost exclusively desktop PCs. We had a lot of good discussion about display resolution in Engineering Windows 7 and likely this will be an interesting topic again.

Knowing this, we investigated a number of options for using widescreen formats more effectively with the goal that the total vertical space available for content was the same after we added the ribbon as it had been in Windows 7. We removed the header at the top of the main view and moved the Details pane to the right side (and also did a visual revamp of the pane) while keeping a one-line status bar at the bottom of the window where we show you critical information.



Details pane for images

This approach gives you a new Details pane that is much easier to read, makes better use of widescreen formats, and preserves screen real estate for the main file/folder pane. The exact number of lines might vary a bit from PC to PC depending on what add-ins you have, but for the out-of-the-box configuration running full screen at 1366 X 768, you can actually fit two more lines on the screen than you could in Windows 7.

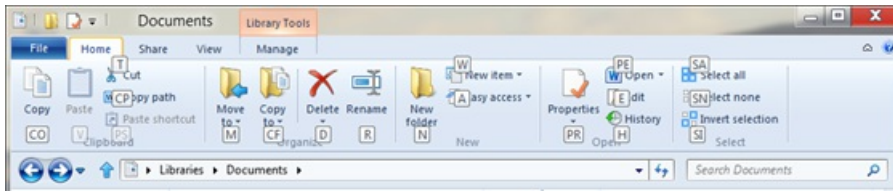


And this comparison assumes you have the ribbon open. If you collapse the ribbon (double-click the tab, or click the Minimize arrow on the right side of the ribbon), you get even more vertical real estate with our new approach.

Making it work well for power users too

Finally, while most of the work we've done is focused on making Explorer work for everyone, we also wanted to make sure we were giving our more sophisticated users a good experience as well.

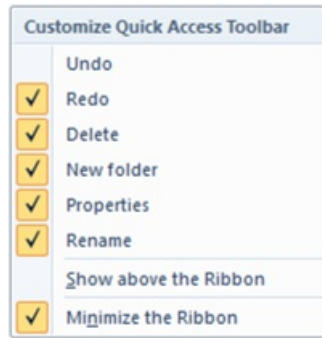
One of the top requests from more advanced users is for more keyboard shortcuts. All of the existing Windows Explorer shortcuts work in this version of Explorer, but with our new approach, all of the approximately 200 commands in the ribbon now have keyboard shortcuts as well. (Note that we haven't finalized the exact number of commands in the ribbon yet. It will likely end up between 198 and 203 when we're done.)



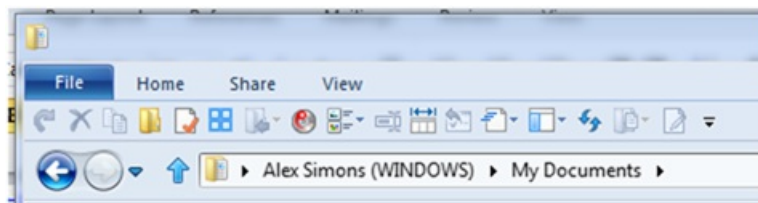
Keyboard shortcuts

Advanced customers have also traditionally asked for the ability to customize Explorer more. The Explorer in Windows XP was probably the most customizable version to date (you could add or remove a pre-specified set of buttons from the toolbar and customize the layout) but the Explorer UI in Windows 7 and Vista had very limited customization options beyond installing third-party add-ons.

The new Quick Access Toolbar (QAT) in Explorer provides a lot of customization opportunities. Similar to Office, by right-clicking any button in the ribbon, you can add it to the QAT. Additionally, you can choose to have the QAT display above or below the ribbon, and to display the ribbon in an open or minimized state. This is a big increase in the level of customization available in Explorer (you can choose approximately 200 commands to add to the QAT) and returns it to a level equal to or greater than we had in Windows XP.



QAT customization options



A customized QAT with a minimized ribbon

Finally, as you may have noticed in several of the screen shots, we just had to bring back the "Up" button.



The return of the "Up" button!

This is far and away the most requested improvement to Explorer, and a great opportunity to bring back some of Windows Explorer's heritage features.

I'll leave you with this quick demo where I walk you through the main features of the new Windows Explorer.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

--Alex Simons

Accessing data in ISO and VHD files

Steven Sinofsky | [2011-08-30T00:01:00+00:00](#)

In continuing with the improvements in core Windows functionality and also oft-requested features, we are adding native Explorer support for ISO and VHD files in Windows 8. While terabytes of storage are available to all of us, managing disk (or disc) image formats remains important for a number of mission-critical operations in many organizations and among power users. We know even more support for VHD is a big request, so stay tuned. Rajeev Nagar authored this post. He is a group program manager on our Storage & File Systems team. --Steven

The trend of incredibly large and small form-factor hard disks means we can store ever increasing amounts of data without worrying about running out of capacity. Windows 8 enables easy access to the contents of two important storage formats, ISO and VHD files. While we generally think of these formats when they appear on media, they are also very useful as files within a file system and that is where native support in Explorer comes in handy.

Working with ISO files

While optical discs continue to be useful in many situations, large hard disks allow us to decrease our dependence on them. Personally, I've spent a load of my time (legally) ripping about 900 GB worth of music, and more recently almost 1TB of home video DVDs into my collection. I know that my backup of our photos and home movies is probably the most important data in my house. Together with backups, storing the most basic things in my house now requires terabytes of space. Just a couple of years ago that was an unimaginable amount of storage. These days, however, I know I can buy a 3TB hard disk for less than \$200.

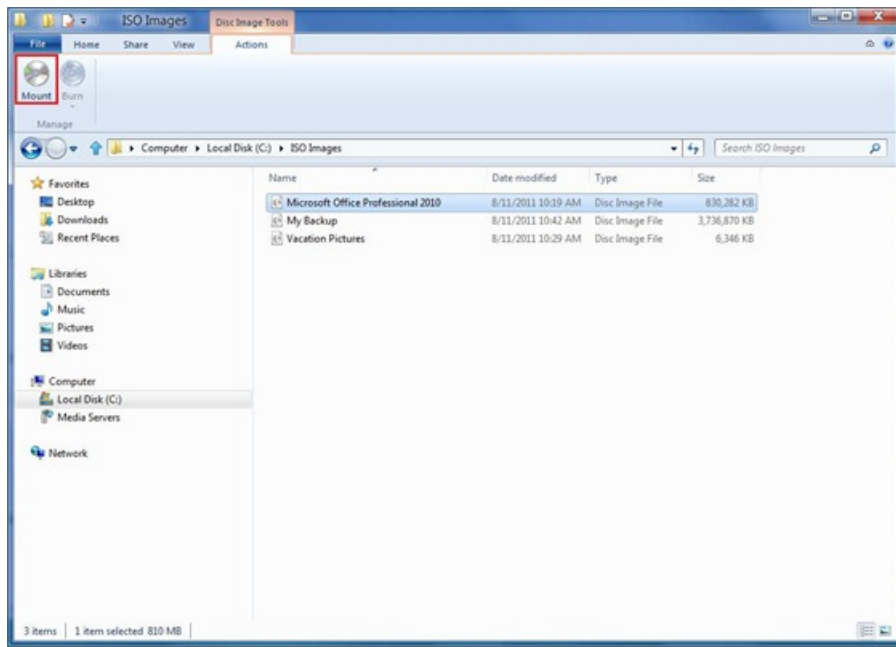
Given cheap hard disks and our mobile lifestyle, we have little interest in carting around collections of discs. Also, we expect to be able to receive content as well as share and collaborate with friends, family, and colleagues in an instant – typically through online file transfers. Last but not least, our desire for thin and light form factors such as slates and ultra-mobile laptops often leaves no room for vendors to add optical disc drives. This is exactly the feedback we received from many of you who used Windows 7 – the ability to directly use *ISO files* (also known as *ISO images*) without requiring a physical CDROM or DVD drive is very important.

A quick refresher on ISO files might be helpful. ISO refers to the [International Organization for Standardization](#) which is an international standard-setting body, and a world leader in developing and publishing international standards. For the purpose of this blog entry, our interests lie in a couple of standards published by ISO, namely [ISO-9660](#) and [ISO-13346](#). Simply stated, these two standards each describe a method by which photos, video, applications, documents or other content (excluding CD audio) are organized on CDROM or DVD optical media. The reason for the popularity of these standards is they allow CDROM and DVD media content to be easily interchanged across systems from different vendors e.g. you can create a DVD on a Windows PC and read it in your living room DVD player. An ISO file is simply a disc image stored as a file, composed of all of the contents of a CDROM or DVD disc. You can also think of an ISO file as a full-fidelity image (digital copy) of the optical disc.

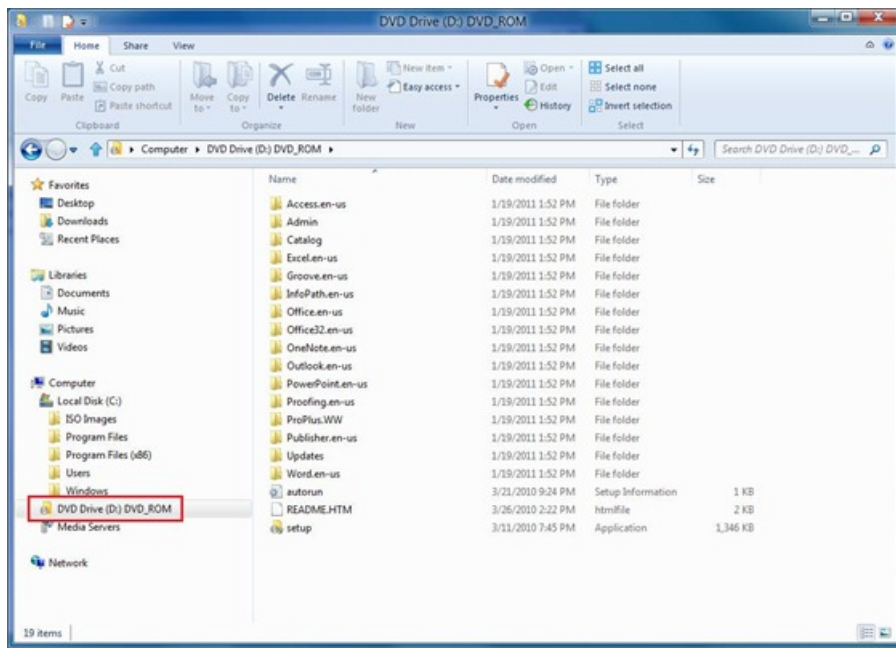
ISO files are used by vendors to distribute software. Backup applications also store content in the ISO format and many utilities allow creation of an ISO file from existing CDROM or DVD media. Once created, these files can be sent around, downloaded, and stored just like any other file – however, before you can access the photos, video, applications, documents, or other content contained *within* the ISO file, you either have to “burn” the ISO file to a writable optical disc or download and install software that allows you to “mount” and access the ISO file contents directly (i.e. without burning). With Windows 8, we have eliminated this last step – you can simply access the contents of the ISO file without needing either needing to burn a new disc or needing to find/download/install additional software just to logically access the ISO.

So how does this work in Windows 8? It's quite simple – just “mount” the ISO file (you can select mount from the enhanced Explorer ribbon or double-click or right-click on the file), and a new drive letter appears, indicating that the contents are now readily accessible. Underneath the covers, Windows seamlessly creates a “virtual” CDROM or DVD drive for you on-the-fly so you can access your data. Let's walk through the flow that will enable you to access such an ISO file.

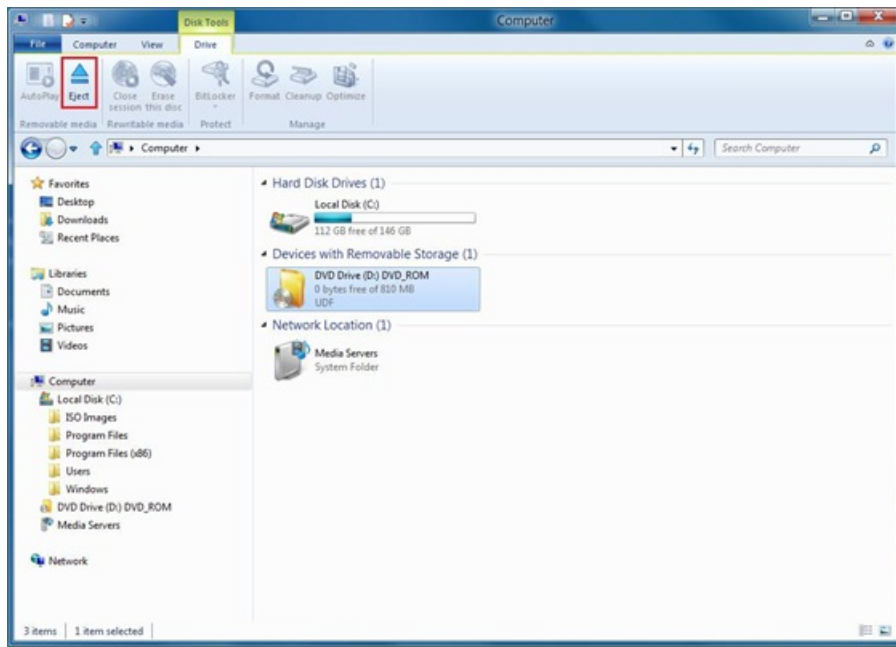
As you see in the figure below, we have three ISO files in a local folder. The one we will work with contains the (legally obtained) Office application suite. To mount the ISO, you can either double click the file or click Mount on the Actions tab.



Once you mount the ISO, a new drive letter appears for the virtual CDROM/DVD drive that Windows seamlessly creates. The contents of the ISO are accessible just as they would have been had you inserted the CD/DVD media into a physical optical drive. Only, operating on the contents happens at the speed of your hard drive, not an optical drive.



Once you are done using the ISO, you can (virtually) "eject" it, and the virtual drive disappears.



In case you need a utility to create ISO images from existing optical media, there are many tools that give you that capability. One I use is the [Oscding command line tool](#) that is available as part of our [automated deployment kit](#).

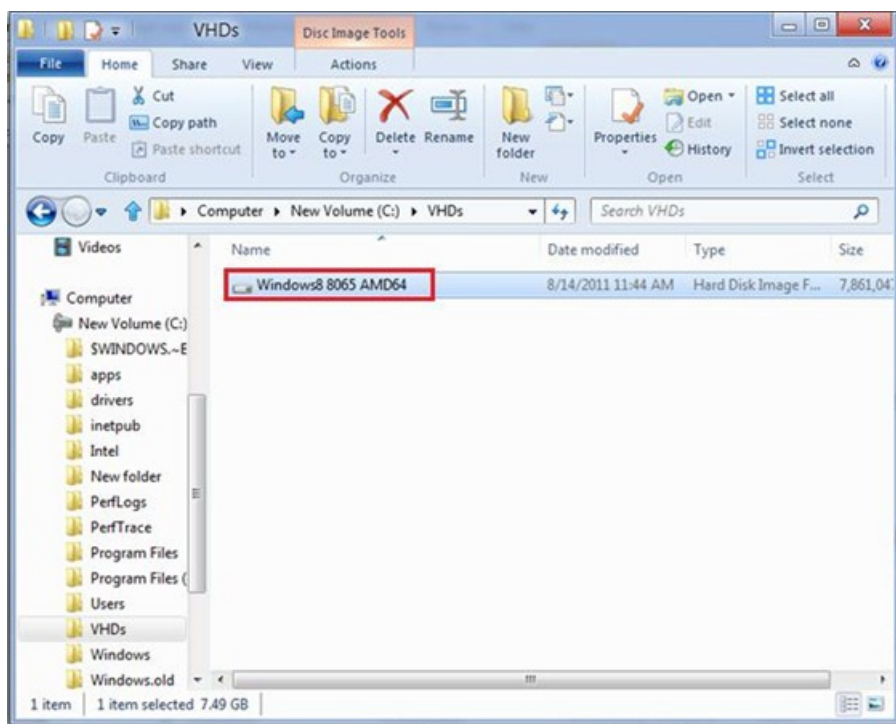
That is it! Accessing ISO files has now become a snap with Windows 8. Regardless of whether you have an optical drive accessible to you or not, accessing your data is never a problem.

Working with VHD files

Another place we've simplified access is with Virtual Hard Disk files. Virtual Hard Disks are the format used by Virtualization software Hyper-V or Virtual PC. In a future blog post, we'll talk more about the enhancements to Windows Virtualization technology, Hyper-V.

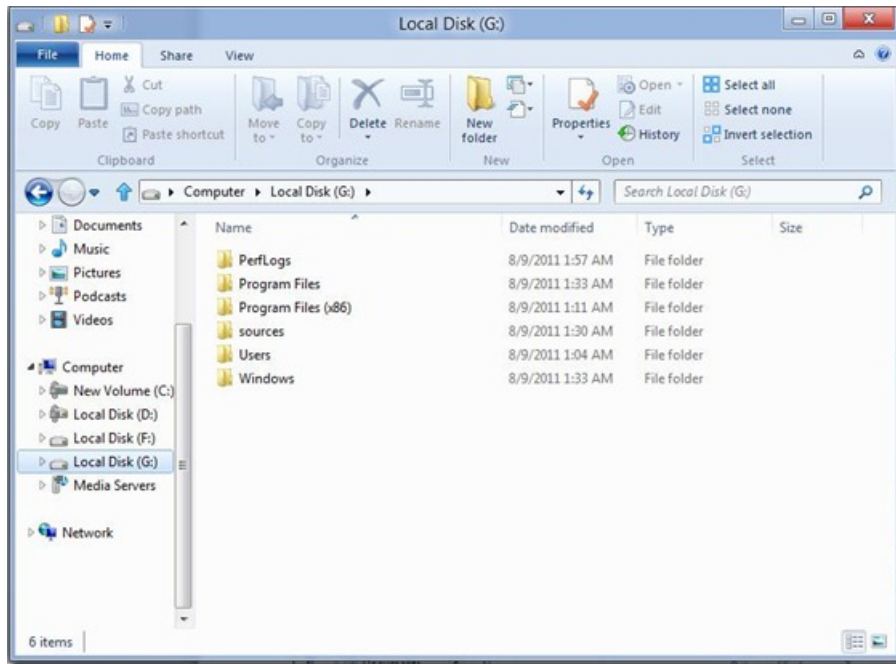
The [Virtual Hard Disk \(VHD\)](#) format is a publicly-available image format specification that allows encapsulation of the hard disk into an individual file for use by the operating system as a virtual disk in all the same ways physical hard disks are used. The VHD format is used by Hyper-V to store information for Virtual Machines. In Windows 7 & Windows Server 2008 R2 we have the ability to boot the system off a VHD file, and we had command line and MMC plugins for managing them. VHDs are handy for portability of system settings or to play back what has been saved as a snapshot of a system.

Accessing a VHD in Windows 8 is as simple as what we've done with ISO files, but there is one important difference: rather than appearing as a removable drive (as is the case with ISO), VHDs appear as new hard drives.

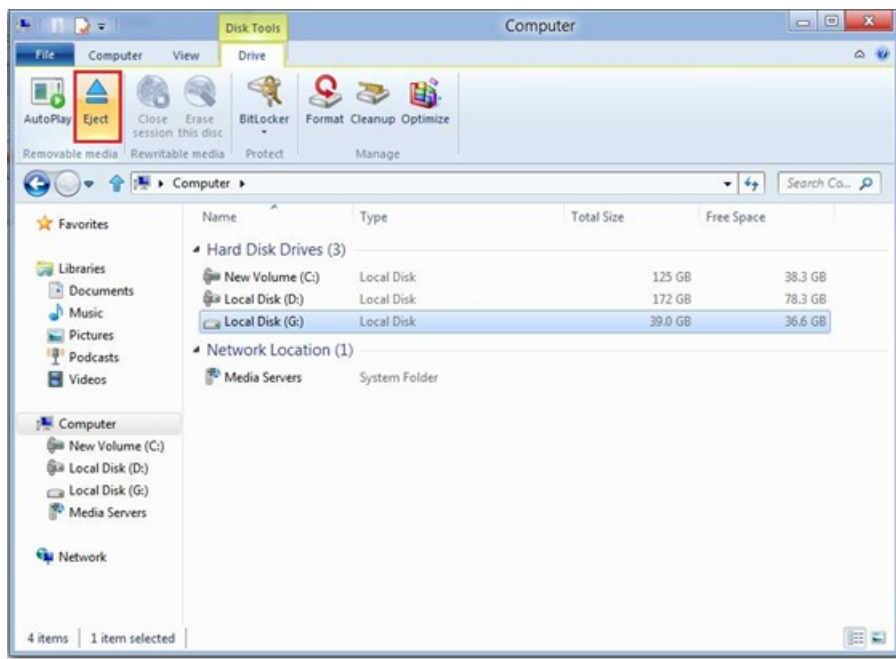


Underneath the covers, Windows provides a virtual drive letter pointing to the volume within the VHD. You'll notice that the icon for the drive G:

below is the same as the icon for a VHD file.



You can then work with the virtual hard disk just like any other file storage in your system, whether you are modifying, adding or removing files.



Once you've finished working with the VHD, like an ISO, you can right-click it and click Eject (or just use the Eject button on the ribbon). Any changes you've made remain saved within the file.

Here's a quick demo to show you what it looks like to mount ISOs and VHDs on a new "Windows 8" system.

Your browser doesn't support HTML5 video.
Download this video to view it in your favorite media player:
[High quality MP4](#) | [Lower quality MP4](#)

--Rajeev

Designing for Metro style and the desktop

Steven Sinofsky | [2011-08-31T08:00:00+00:00](#)

We thought it would be good to take a moment to talk about where we are heading in terms of the user interface of Windows 8.

By now you've seen two different elements of the Windows 8 design—first, a Metro style user interface we showed previously and in a video seen by millions of folks. And recently, we have described in this blog some of the enhancements we're making to familiar Windows desktop tools such as Explorer and the copy file dialog. We've seen a lot of dialog about these changes.

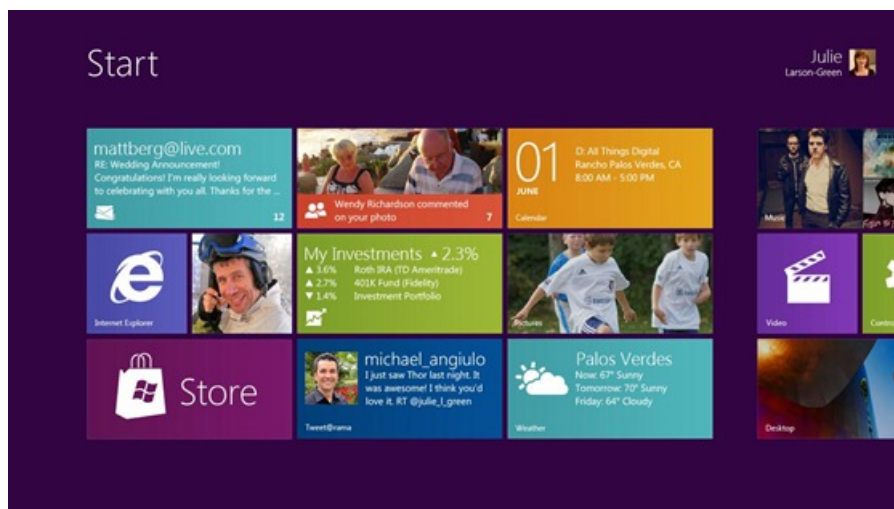
Some of you are probably wondering how these parts work together to create a harmonious experience. Are there two user interfaces? Why not move on to a Metro style experience everywhere? On the other hand, others have been suggesting that Metro is only for tablets and touch, and we should avoid “dumbing down” Windows 8 with that design.

This is a balancing act, and one we'll be talking quite a lot about in this blog in the coming months. Having both of user interfaces together harmoniously is an important part of Windows 8. As a starting point for the discussion, here is how we approached the design of Windows 8 from the very beginning.

We started planning Windows 8 during the summer of 2009 (before Windows 7 shipped). From the start, our approach has been to reimagine Windows, and to be open to revisiting even the most basic elements of the user model, the platform and APIs, and the architectures we support. Our goal was a no compromise design.

This is an ambitious undertaking—it involves tools, APIs, languages, UI conventions, and even some of the most basic assumptions about a PC. For example, how do you isolate applications from each other, or prevent applications from stealing all your battery power? How can installing (and removing) apps be as quick and painless as changing the channel on the TV? How do you attract the broadest set of developers possible to a new platform? How do you build a touch-first interface with a unique point of view?

When we showed the first demos of Windows 8, we introduced our new Metro experience—fast and fluid, immersive, beautiful, and app-centric. We are certain that as we show you more in the coming months you will see just how deeply we have reimaged Windows. Metro style is much more than the visual design as we shall see.



At the same time, we recognized that Windows 7 has been a huge success. Not just as measured by sales figures or by the number of people using it, but also by the depth of usage. Hundreds of millions of people rely on the Windows 7 UI and existing Windows apps and devices every day, and would value (and expect) us to bring forward aspects of that experience to their next PCs.

In this light, the role of the Windows desktop is clear. It powers the hundreds of thousands of existing apps that people rely on today, a vast array of business software, and provides a level of precision and control that is essential for certain tasks. The things that people do today on PCs don't suddenly go away just because there are new Metro style apps. The mechanisms that people rely on today (mice, physical keyboards, trackpads) don't suddenly become less useful or “bad” just because touch is also provided as a first-class option. These tools are quite often the most ergonomic, fast, and powerful ways of getting many things done.

We knew as we designed the Windows 8 UI that you can't just flip a bit overnight and turn all of that history into something new. In fact, that is exactly what some people are afraid of us doing. Some have said that is the only path to take. Yet, even those who have fully embraced tablets also own a laptop for those times when they need more precise control or need to use one of the apps that are mission critical (and are still being developed). In people's desire not to carry around two different devices, “remote desktop” programs for tablets and phones have become popular but extremely awkward attempts to harness the usefulness of the Windows 7 desktop within a new form factor.

Why not just start over from scratch? Why not just remove all of the desktop features and only ship the Metro experience? Why not “convert” everything to Metro? The arguments for a “clean slate” are well known, both for and against. We chose to take the approach of building a design

without compromise. A design that truly affords you the best of the two worlds we see today. Our perspective rests on the foundation of the open PC architecture that has proven flexible and adaptable over many significant changes in hardware capabilities and software paradigms. This is the flexibility that has served as a cornerstone through transitions in user interface, connectivity, programming models, and hardware capabilities (to name a few).

We believe there is room for a more elegant, perhaps a more nuanced, approach. You get a beautiful, fast and fluid, Metro style interface and a huge variety of new apps to use. These applications have new attributes (a platform) that go well beyond the graphical styling (much to come on this at Build). As we showed, you get an amazing touch experience, and also one that works with mouse, trackpad, and keyboard. And if you want to stay permanently immersed in that Metro world, you will never see the desktop—we won't even load it (literally the code will not be loaded) unless you explicitly choose to go there! This is Windows reimagined.

But if you do see value in the desktop experience—in precise control, in powerful windowing and file management, in compatibility with hundreds of thousands of existing programs and devices, in support of your business software, those capabilities are right at your fingertips as well. You don't need to change to a different device if you want to edit photos or movies professionally, create documents for your job or school, manage a large corpus of media or data, or get done the infinite number of things people do with a PC today. And if you don't want to do any of those "PC" things, then you don't have to and you're not paying for them in memory, battery life, or hardware requirements. If you do want or need this functionality, then you can switch to it with ease and fluidity because Windows is right there. Essentially, you can think of the Windows desktop as just another app.

Windows 8 brings together all the power and flexibility you have in your PC today with the ability to immerse yourself in a Metro style experience. You don't have to compromise! You carry one device that does everything you want and need. You can connect that device to peripherals you want to use. You can use devices designed to dock to large screen displays and other peripherals. You can use convertible devices that can be both immersive tablets and flexible laptops.

Which brings us back to the improvements we're making to the desktop experience: we believe in the Windows desktop. It powers the experiences today that make a Windows 7 PC the most popular device in the world. So, even if we believe that over time many scenarios will be well-served by Metro style apps, for the foreseeable future, the desktop is going to continue to play a key role in many people's lives. So we are going to improve it. We're having a good dialog about what folks might think about our design choices but also wanted to put these choices in a broader context of the unmatched utility of the desktop.

Our design goal was clear: no compromises. If you want to, you can seamlessly switch between Metro style apps and the improved Windows desktop. Existing apps, devices, and tools all remain and are improved in Windows 8. On the other hand, if you prefer to immerse yourself in only Metro style apps (and platform) and the new user experience, you can do that as well! Developers can target the APIs that make sense for the software they wish to deliver. People can debate how much they need or don't need different aspects of the product, but that has always been the case. All of this is made possible by the flexibility of Windows.

This is just the beginning of the discussion. There's so much more to talk about as we dive into details about the Windows 8 UI. We're delivering a whole new experience, reimagined from the chips all the way to the user experience, to enable new scenarios, new apps, and new ways of using a PC.

--Steven

Reflecting on our first conversations (part 1)

Steven Sinofsky | [2011-09-01T12:01:00+00:00](#)

When we kicked off this blog, the premise was a dialogue – a two-way conversation about building Windows 8. As we said we intended to do, we’ve started a discussion about how we build the product and have had a chance to have some back and forth in comments and in posts about topics that are clearly important to you. To put some numbers on things, I’ve personally received about 300 email messages (and answered quite a few) and in total, we have had just over 3,000 English language comments from about 1,700 readers. In terms of Twitter followers, we have leveled off at about 15,000, (which seems to be about the size of the “market” for following a blog like this based on comparable handles). Just as with the Engineering Windows 7 blog, early in the process I wanted to take a step back to [reflect](#) on the dialogue we’re having and focus on a few of the themes. This is a normal part of starting up a new blog—lots of early energy and everyone finding their voice and rhythm.

We knew talking about Windows 8 would be different than talking about Windows 7. Whereas Windows 7 was about returning to roots, Windows 8 is about maintaining those roots while moving forward in a big and new way. Moving in a new direction always brings engineering challenges as well as challenges in just talking about what we’ve done. This is especially the case for Windows 8 for two reasons.

First, we’re talking about a product used by a billion people. No matter how you slice it, that is going to create a very, very large number of perspectives and customers to serve. Of course there is very high value in serving a broad set of customers with one very open product. We’re seeing a lot of this play out in the comments where people state emphatically their perspectives, reassert them frequently, and are diametrically opposed to one another. Our job, which we view with deep responsibility, is delivering a product that spans customer scenarios and builds on the value of having a single product (for developers, IT administrators, PC manufacturers, hardware vendors, etc.).

Second, we’re changing the user experience model of Windows 8. As anyone who has done user interface work (and more importantly as anyone who has used an interface) knows, having an opinion on user interface is not difficult. Even mocking up static images of how things could look is not that hard. My inbox is filled with mock-ups and proposals of dialog boxes and toolbars. But it already was—we’ve been doing this process for a long time. The difficulty in talking about UI through static images is much like trying to summarize or review a movie based on only viewing a still. Our own testing uses dozens of images in sequence when we evaluate designs.

We went back and forth quite a bit over how to begin blogging. There’s obviously a strong desire to know more. At the same time, we think that when we try to do big things there should be an opportunity to engage in a stepwise discussion of the story. Movies don’t start with the end and you get to meet the characters and motivations behind them (in a well done screenplay). There’s always learning for us in how we approach this, as the combination of the environment and work we’re doing is unique at each juncture.

In that sense, we learned one very valuable lesson early on, which is that discussing user interface is something a lot of people want to do, but doing so through static images very quickly misses the point. Very much like zooming in too far with a microscope, the big picture is lost. It also surfaces the least actionable sorts of feedback to wade through of the “love it” / “hate it” variety. Even with short videos we have not found the right way to put context around the overall experience. Given enough focus, light, and magnification, anything can become important and the subject of a big debate. We certainly contributed to that.

In this, and a subsequent post, I want to talk about four topics in particular: Feedback (which I’ll cover today), the Ribbon, Metro, and Media Center. I hope to add a bit of additional “focus, light, and magnification” without distorting the bigger picture here. Based on the comments and the dialogue, I do feel that each of these deserves some further discussion. One additional topic folks want to talk more about that we will cover at BUILD is the overall programming model. Early on it was clear this is the sort of topic that will take more than a blog post because we have so much to say and to demonstrate.

Feedback

The blog is a feedback mechanism for sure. It is one of many. We’re committed to absorbing and internalizing the feedback. It is fair to say that no other product is both used by so many but also has this channel of dialogue, and certainly not before it is released. How we use this channel is certainly an interesting discussion.

I’ve certainly received my share of extremely warm messages telling me to ignore “those trolls and fanboys” and “what you’re saying resonates.” Those are nice to read in the face of an equal number of messages telling me how poor a job we’re doing. We also receive a great many very specific questions and suggestions.

The importance of Windows to so many is clear when we receive such suggestions. It tells us how big a difference Windows makes in people’s lives at home, work, and school. Small changes in the product can make things much easier. Big changes have the opportunity to dramatically improve things, or perhaps not. Our job—what we come to work to do every day—is to figure out how to make changes to the product so that it is better at doing what you expect it to do, and at doing new things that you might not be expecting it to do.

We would love to answer each comment or comment on each proposal, but we are outnumbered, literally. And that is only talking about the blog. Our approach is to listen carefully. Respond to comments that are representative of a theme or represent a topic we think will shed some light on the dialogue. Members of the team have been part of the dialogue. At least 20 senior members of the team have posted comments so far. That will gradually increase as the dialogue evolves.

I just want to reiterate that we are actively participating. Believe me, this blog is the “talk of the town” here in Redmond. ?? We look forward to

the continued exchanges – the good feedback, the critique, and the constructive comments. It helps us deliver to you all a product that meets our stated goal of Windows, reimagined.

...more to come

-- Steven

Reflecting on our first conversations (part 2)

Steven Sinofsky | [2011-09-02T12:01:00+00:00](#)

As mentioned at the start of the last post, I wanted to pause for a moment and look back at the dialogue we've had via this blog, further exploring a few of the exchanges and inquires, just as we did with the Engineering Windows 7 blog. We'll pick up where we left off the last post, discussing the importance of feedback, and then we'll look at the discussions around the Ribbon, Metro style, and Media Center availability.

Ribbon

We thought the amount of energy around the redesign of copying files was immense. And then we posted about Windows Explorer. We even went into this knowing that there would be a lot of "energy." To anyone who has hit on a controversial blog topic, the patterns are familiar. Rather than focus on the number of referrals from Slashdot (a lot more than other posts) or blog server performance (we tweaked our site layout for efficiency), let's focus on the design choices.

Most importantly, this is one element of the product. Just as with the copy conflict dialog when you shine a light on it, brightly, there is a tendency to miss some things (on both sides) and to place more emphasis than it deserves. To return to our movie analogy, it is like when a specific scene is picked to market a movie and it inadvertently shifts the dialogue about the film (or even the target market). The good news is we have lots to talk about—a lot.

Without repeating the first post, I would add that we do believe we have taken into account many of the criticisms we were certain would surface. We chose the ribbon mechanism, and to those that find that a flawed choice, there isn't much we can do other than disagree. We were certain, and this proved out, that the dislike of the ribbon is most intense in the audience of this blog. Said dislike, we assumed, would produce a high level of commentary, much the way some topics during Windows 7 blogging did. That assumption was correct.

There was a lot of back and forth over the role of the mechanism for different customers—is it advanced or beginner targeted? There's irony here in that menus were once for beginners (the keyboard was what power users used), which then were simplified with toolbars. Context menus were originally shortcuts for advanced users, but ended up being used more by everyone. Now we are hearing (and seeing) that menus and toolbars are being touted for advanced users. Of course, we have been trying to unify these disparate mechanisms in an effort to have a simpler experience—fewer mechanisms means less UI surface area, by definition. While there are a lot of opinions, the one thing we know is that the satisfaction with our products that use the ribbon is much higher and the usage much broader and deeper. We also know a very small set of people remain unhappy. That was true in versions before the introduction of the Ribbon mechanism, though obviously for different reasons. It might be the case that no matter what we do, there will be a small set of people that are not satisfied?

To me the most interesting feedback has been about the visual overhead. The role of "Metro" came up and how we should use a lighter graphical treatment and also just expose fewer commands because people want minimalism now. Obviously we all want less—fewer exposed features means less surface area which means less code to write, test, maintain. Minimalism is not hiding features or making useful things hard to get to. Minimalism is about stripping things down to fundamental features. The question then is really defining that set of features. Our approach to minimalism is to avoid layers of commands or hidden pockets of features (those mechanisms themselves become conceptual and code overhead—bloat can come from UI itself, not just what UI exposes), and to reduce the number of mechanisms in the UI. In doing so, we aim to present the capabilities of the product in one manner. We also know that minimalism is not about doing less stuff, especially in light of all the feedback over features people want to see added to explorer.

The progressive or hierarchical rendering of features is the world we came from—some features are keyboard only, some context menu only, some top-level toolbars, some on toolbars you had to show/hide, some on menus or submenus, etc. This collection of mechanisms doesn't work well for anyone except those who invest a lot of time. Of course if you invest a lot of time you become a very outspoken opponent of change. Perhaps there is some of that? I was a strong proponent of the Office 2000 "adaptive menus" that literally drove people crazy, and those were a deliberate attempt to have less clutter and less surface area. One failure is not a trend but the lesson that "hiding is not simplifying" is valuable I think.

We have work to do as we continue to refine the way we have organized commands and what commands we should organize (map network drive, powershell), as well as the default settings and graphical treatment. We are actively considering the feedback in this regard. We share the goal in having a clean user experience. We also have the goal of making sure people can get done the things they do want to get done. The role of data here is important when used correctly and it also helps us to avoid the use of small data sets or anecdotes driving the choices.

As the role of "Metro" came up, it became clear that for some, Metro is symbolic of a particular "palette" of colors and fonts, and perhaps some notion of controls. Several of the screen shots proposed were ones where some set of (less favored?) commands were dropped but primarily the change was to reduce the overall palette. We found the comparisons to some Metro apps like Zune interesting when you consider the amount of usage of competing products that are so much more "dense" and the requests for features that are in neither of the media players (CODECs, tags, etc.)

We've been looking at this and also trying to reconcile the feedback in this very forum around Windows 7 feeling washed out or "pale." We actually added intensity and pixels to what we had in the Windows 7 design because of feedback we received via the blog. We'll continue to look at this area of course, but want to avoid "churn" at a stylistic level because so many third-party products tend to mimic the Windows experience without utilizing the built-in metrics and system settings to obtain the palette (so things can look awkwardly different). This leads to the discussion of

Metro styling

Metro style

The challenge with the discussion of Metro style was definitely due to the ordering of our blog posts. We were not sure whether to speak abstractly at first or concretely. Given that nearly 6 million people viewed the video demonstration of the Windows 8 user experience (a rather in-depth demo) we felt that people already had the context for the user experience. That probably wasn't a good assumption on our part. We also know that even with that much background, until you can touch the software it is going to be difficult to develop a complete picture. Many products look better or worse until you can use them. I'm fairly certain that in this case we have a lot of upside.

In many of the comments, people primarily focused on Metro as what I would say are the graphical elements of the user interface—it was Metro v. Aero. We've seen a clear turn where Aero is the past and Metro is the future. And with that a strong desire for the existing Windows experience to take on a new look or a Metro redesign. These comments are usually focused on style and looking "old" or "new." Generally, those details of the visual styling come later in the engineering process, but we wrongly assumed that this was known. Stating that, we could have short-circuited this concern.

A lot of this discussion will depend ultimately on what Metro comes to mean for folks. As we looked at Metro style for Windows 8, as we talked about in an earlier post, we see much more than a more monochrome set of visuals and fewer controls (when there are fewer commands). We see a new platform, a reimagining of Windows. For Windows 8, Metro style means a new type of app—an app that learns from and improves upon the current (and most popular) platform. This is a lot of what we'll talk about at BUILD. If you watch [Video #1](#) you can see Metro style apps at work. At BUILD we'll talk about the attributes of those apps, and the tools and languages you can use to create those apps. What we've said is that there is a very deep platform that provides for a rich opportunity for apps of all types—from media to social to games to productivity. We don't see any limits to where this will go.

The other part of this dialogue is about the desktop. The desktop is many things to many people. To some it is literally the place where the important documents are stored (the most important folder). To some it is the Explorer window for managing files (it is an app to some). To some it is a metaphor or even Windows itself (toolbars/ribbons, menus, MDI/SDI, etc.). To some, their view of the desktop is an app they run "all the time" and they experience Windows only through File Open or maybe the Start menu (for example, people who for the most part use Outlook or Word, or Photoshop, or AutoCAD, or a line of business app). The desktop might even be relatively invisible to someone who does a lot of web browsing.

The unique element of Windows has always been the "open market" approach to interface. We embraced how people used and adapted the Windows APIs to bring unique experiences to market. Within any context there really isn't a single "desktop" experience. Certainly some have been critical in the past that "Aero" did not achieve a uniformity or consistency, even within Windows.

We said the desktop is like an app in Windows 8—you can use it or not, as much or as little as you want. Some have said "it feels jarring" to go to the desktop. My perspective is that it is no more or less jarring that switching between any other apps if you embrace diversity or experiences that are built for a specific task or purpose. Today's websites (and mobile apps) do not strive for consistency across disparate properties or apps, and the shell of a browser does little to prevent a jarring effect as you switch tabs (or apps). We've long embraced apps that have palettes or toolbars, full screen / windowed / MDI, built-in controls or custom controls. The mechanisms to implement this variety are part of the desktop heritage. Some wish for more uniformity or policing. As a member of the team that built our early Windows tools, I know we tried. Even in the most homogeneous platform, developers will strive to differentiate and build their user experiences for specific purposes and experiences will diverge from commonality. Commonality was an answer to complexity in another era--today we are surrounded by digital experiences of all different types and we readily adapt (just as people adapted to a wide variety of print formats or video formats as these technologies improved from their first generations). The answer today is whether the design works in the context for which it was intended.

That diversity allows us to say with confidence that going from Metro style to the desktop will be harmonious—as harmonious as switching apps or sites is today. It will take orchestration at the top level to make moving seamless—that's why you see things like switching between apps, snapping apps, or even using ALT+TAB between apps, and the desktop itself, all mechanisms that just work. The animations will work. Copy / Paste will work. Even bridging between "legacy" control panel applets will work.

There's a lot more to this topic, as I said. I wanted to touch on some of the feedback and play back some of what has occurred to me and other folks on the team as we read through the dialogue. I think we need to do more in terms of showing the product and maybe we erred on the side of too much transparency too soon, but we're on board and moving forward. BUILD is just a few days away.

Media Center

While not a central topic of feedback, I received about 50 emails about Media Center. I want to reassure customers that Media Center will definitely be part of Windows 8. No doubt about it. Knowing how strong the support for Media Center is among pre-release testers, we still have work to do to make sure the quality and compatibility with add-ins is what you would expect even in pre-release (as with any release of Windows, compatibility is a major effort and when we work on the underlying video engine, as one example, we have to make sure features that push these areas receive adequate coverage).

In the coming months, many folks will be testing pre-release builds of Windows 8. As everyone knows, two things are always the case early on. First, the software is not done and things will change—features will be added and removed. Second, the different editions or SKUs are not developed or announced until late in the development process (closer to market availability).

Media Center will not be part of the first pre-release builds. Some other features/capabilities will not be in the first pre-release builds including: Windows 7 games, DVD Creator, upgrade setup, Dot Net 3.5 (Note there are perhaps a couple of other relatively low profile items but just wanted to hit the major ones here). These are engineering decisions as well as business decisions.

As we get closer to market availability, we will make sure to explain how not just these specifically but all features of the product will be made available. As an aside, it is early to start the dialogue about a preference for one SKU with Windows. We're well aware of this feedback and we always need to balance it with the feedback from our business partners who value a different approach. We'll cross that bridge when we come to it. Interestingly, the feedback about Media Center was predominantly "we will pay extra, just include it" based on the input directly to me. Today Media Center is part of "premium" SKUs for Windows, which means that is the case today.

In addition a lot of folks said "everyone I know uses it." As I said, we're completely committed to delivering Media Center in Windows 8, but I wanted to share some of the usage data. This data in no way influenced not delivering it as part of the first pre-release build—we are as committed as ever to Media Center.

Our opt-in usage telemetry shows that in July, Windows Media Center was launched by 6% of Windows 7 users globally with the heaviest usage in Russia, Mexico, and Brazil (frequency and time). However, most people are just looking around; only one quarter (25% of 6%) of these people used it for more than 10 minutes per session (individual averages), and in 59% of Media Center sessions (by these 6% of users) we see almost no activity (less than a minute or two of usage). TV was the most common scenario we observed, and not surprisingly, traditional media (DVD and CD) are less common (and declining over time) than streaming and file-based content. By comparison, Media Player (66% of Windows users in July) and IE (88%) are popular rendering engines for all types of media content, including an increased volume of "premium" and streaming content. This is another place we're reminded of the tremendous diversity of Windows activity.

Those were a few of the major topics raised in comments in our first posts. Just as we did for Engineering Windows 7, we took a step back to calibrate our dialogue and reflect on the conversation. It is hard for us to find negatives in the feedback and passion around what we're doing—what could possibly be better than to work on something that so many people care so deeply about. That you are investing so much in what we write and create, even before you have a chance to use the software, is certainly humbling. We're focused on BUILD and making sure we do a great job showing off the work we've done, and are definitely excited to continue the dialogue. I look forward to those conversations in person at BUILD and here on the blog as well.

--Steven

Bringing Hyper-V to “Windows 8”

Steven Sinofsky | [2011-09-07T09:00:00+00:00](#)

In this post we talk about how we will support virtualization on the Windows "client" OS. Originally released for Windows Server where the technology has proven very popular and successful, we wanted to bring virtualization to a core set of scenarios for professionals using Windows. The two most common scenarios we focused on are for software developers working across multiple platforms and clients and servers, and IT professionals looking to manage virtualized clients and servers in a seamless manner. Mathew John is a program manager on our Hyper-V team and authored this post. One note is that, as with all features, we're discussing the engineering of the work and not the ultimate packaging, as those choices are made much later in the project. --Steven PS: We didn't plan on doing so many posts in a row so we'll return to more sustainable pace -- sorry if we inadvertently set expectations a bit too high. We're getting ready for BUILD full time right now!!

Whether you are a software developer, an IT administrator, or simply an enthusiast, many of you need to run multiple operating systems, usually on many different machines. Not all of us have access to a full suite of labs to house all these machines, and so virtualization can be a space and time saver.

In building Windows 8 we worked to enable Hyper-V, the machine virtualization technology that has been part of the last 2 releases of Windows Server, to function on the client OS as well. In brief, Hyper-V lets you run more than one 32-bit or 64-bit x86 operating system at the same time on the same computer. Instead of working directly with the computer's hardware, the operating systems run inside of a *virtual machine* (VM).

Hyper-V enables developers to easily maintain multiple test environments and provides a simple mechanism to quickly switch between these environments without incurring additional hardware costs. For example, we release [pre-configured virtual machines](#) containing old versions of Internet Explorer to support web developers. The IT administrator gets the additional benefit of virtual machine parity and a common management experience across Hyper-V in Windows Server and Windows Client. We also know that many of you use virtualization to try out new things without risking changes to the PC you are actively using.

An introduction to Hyper-V

Hyper-V requires a 64-bit system that has Second Level Address Translation (SLAT). SLAT is a feature present in the current generation of 64-bit processors by Intel & AMD. You'll also need a 64-bit version of Windows 8, and at least 4GB of RAM. Hyper-V does support creation of both 32-bit and 64-bit operating systems in the VMs.

Hyper-V's dynamic memory allows memory needed by the VM to be allocated and de-allocated dynamically (you specify a minimum and maximum) and share unused memory between VMs. You can run 3 or 4 VMs on a machine that has 4GB of RAM but you will need more RAM for 5 or more VMs. On the other end of the spectrum, you can also create large VMs with 32 processors and 512GB RAM.

As for user experience with VMs, Windows provides two mechanisms to peek into the Virtual Machine: the VM Console and the Remote Desktop Connection.

The VM Console (also known as VMConnect) is a console view of the VM. It provides a single monitor view of the VM with resolution up to 1600x1200 in 32-bit color. This console provides you with the ability to view the VM's booting process.

For a richer experience, you can connect to the VM using the Remote Desktop Connection (RDC). With RDC, the VM takes advantage of capabilities present on your physical PC. For example, if you have multiple monitors, then the VM can show its graphics on all these monitors. Similarly, if you have a multipoint touch-enabled interface on your PC, then the VM can use this interface to give you a touch experience. The VM also has full multimedia capability by leveraging the physical system's speakers and microphone. The Root OS (i.e. the main Windows OS that's managing the VMs) can also share its clipboard and folders with the VMs. And finally, with RDC, you can also attach any USB device directly to the VM.

For storage, you can add multiple hard disks to the IDE or SCSI controllers available in the VM. You can use Virtual Hard Disks (.VHD or .VHDX files) or actual disks that you pass directly through to the virtual machine. VHDs can also reside on a remote file server, making it easy to maintain and share a common set of predefined VHDs across a team.

Hyper-V's "Live Storage Move" capability helps your VMs to be fairly independent of the underlying storage. With this, you could move the VM's storage from one local drive to another, to a USB stick, or to a remote file share without needing to stop your VM. I've found this feature to be quite handy for fast deployments: when I need a VM quickly, I start one from a VM library maintained on a file share and then move the VM's storage to my local drive.

Another great feature of Hyper-V is the ability to take *snapshots* of a virtual machine while it is running. A snapshot saves everything about the virtual machine allowing you to go back to a previous point in time in the life of a VM, and is a great tool when trying to debug tricky problems. At the same time, Hyper-V virtual machines have all of the manageability benefits of Windows. Windows Update can patch Hyper-V components, so you don't need to set up additional maintenance processes. And Windows has all the same inherent capabilities with Hyper-V installed.

Having said this, using virtualization has its limitations. Features or applications that depend on specific hardware will not work well in a VM. For

example, Windows BitLocker and Measured Boot, which rely on TPM (Trusted Platform Module), might not function properly in a VM, and games or applications that require processing with GPUs (without providing software fallback) might not work well either. Also, applications relying on sub 10ms timers, i.e. latency-sensitive high-precision apps such as live music mixing apps, etc. could have issues running in a VM. The root OS is also running on top of the Hyper-V virtualization layer, but it is special in that it has direct access to all the hardware. This is why applications with special hardware requirements continue to work unhindered in the root OS but latency-sensitive, high-precision apps could still have issues running in the root OS.

As a reminder, you will still need to license any operating systems you use in the VMs.

Here's a quick run-through of how the Hyper-V works in Windows 8.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

Supporting VM communication through wireless NICs

As you saw in the demo, creating an external network switch is as simple as selecting a physical network adapter (NIC) from a drop-down list and clicking OK. This already worked well for Windows Server Hyper-V, but to have similar results in Windows 8, we needed to get it working with wireless NICs, a new challenge.

The problem

The virtual switch in Hyper-V is a "layer-2 switch," which means that it switches (i.e. determines the route a certain Ethernet packet takes) using the MAC addresses that uniquely identify each (physical and virtual) network adapter card. The MAC address of the source and destination machines are sent in each Ethernet packet and a layer-2 switch uses this to determine where it should send the incoming packet. An *external* virtual switch is connected to the external world through the physical NIC. Ethernet packets from a VM destined for a machine in the external world are sent out through this physical NIC. This means that the physical NIC must be able to carry the traffic from all the VMs connected to this virtual switch, thus implying that the packets flowing through the physical NIC will contain multiple MAC addresses (one for each VM's virtual NIC). This is supported on wired physical NICs (by putting the NIC in promiscuous mode), but not supported on wireless NICs since the wireless channel established by the WiFi NIC and its access point only allows Ethernet packets with the WiFi NIC's MAC address and nothing else. In other words, Hyper-V couldn't use WiFi NICs for an external switch if we continued to use the current virtual switch architecture.

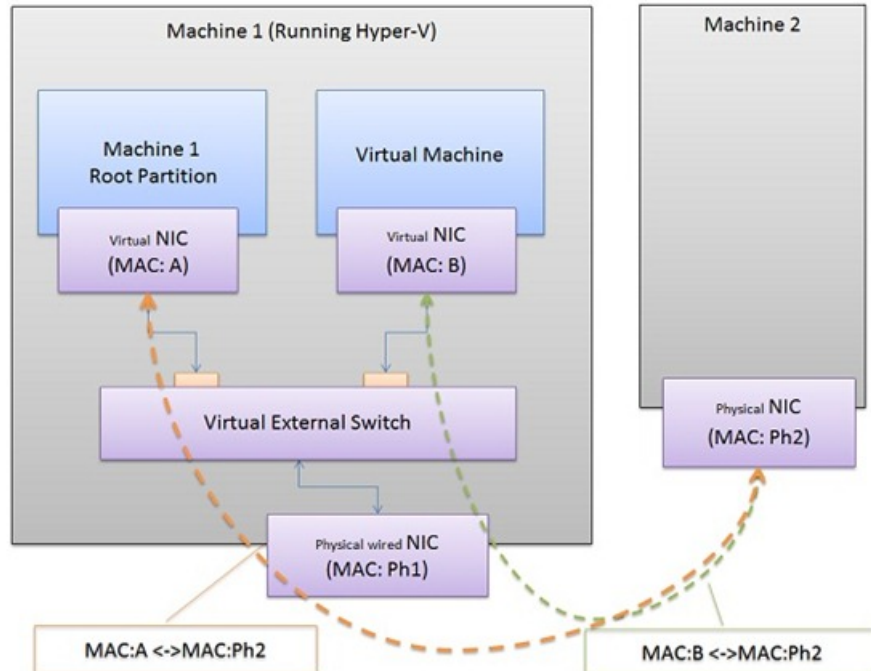


Figure 1: Networking between VM and external machine using wired connection

The solution

To work around this limitation, we used the Microsoft Bridging solution, which implements ARP proxying (for IPv4) and Neighbor Discovery proxying (for IPv6) to replace the *virtual* NICs' MAC address with the WiFi adapter's MAC address for outgoing packets. The bridge maintains an internal mapping between the virtual NIC's IP address and its MAC address to ensure that the packets coming from the external world are sent to the appropriate virtual NIC.

Hyper-V integrates the bridge as part of creating the virtual switch such that when you create an external virtual switch using a WiFi adapter, Hyper-V will:

1. Create a single adapter bridge connected to the WiFi adapter
2. Create the external virtual switch
3. Bind the external virtual switch to use the bridge, instead of the WiFi adapter directly

In this model, Ethernet switching still happens in the virtual switch, and MAC translation occurs in the bridge. For the end user who is creating an external network, the workflow is the same whether you select a wired or a wireless NIC.

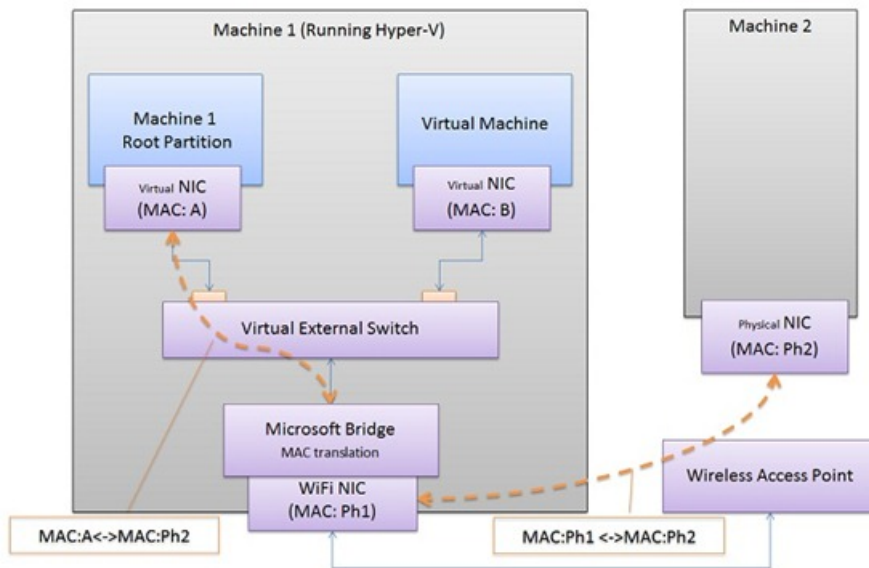


Figure 2: Networking between VM and external machine using WiFi connection

In conclusion, by bringing Hyper-V from Windows Server to Windows Client, we were able to provide a robust virtualization technology designed for the scalability, security, reliability, and performance needs of most data centers. With Hyper-V, developers and IT professionals can now build a more efficient and cost-effective environment for using and testing across multiple machines.

--Mathew John

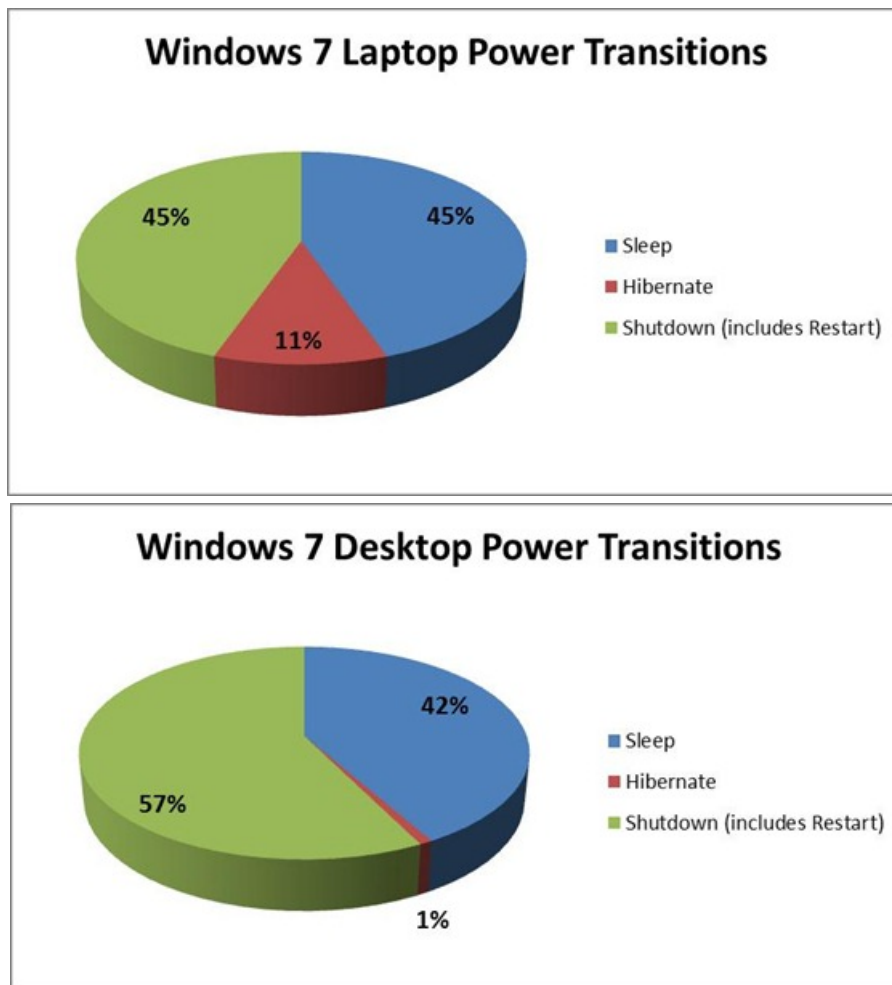
Delivering fast boot times in Windows 8

Steven Sinofsky | [2011-09-08T19:00:00+00:00](#)

When it comes to talking about "fundamentals" we want to start with boot time – no feature gets talked about and measured more. We designed Windows 8 so that you shouldn't have to boot all that often (and we are always going to work on reducing the number of required restarts due to patching running code). But when you do boot we want it to be as fast as possible. This is a very deep topic and we have a lot of folks focused on it. We made a bigger leap in this area with Windows 8 than we have in a long time due in no small part to cooperation across the whole ecosystem. Gabe Aul, a director of program management in Windows, authored this post (a first in what will be a series of posts on fundamentals).

--Steven

Few operations in Windows are as scrutinized, measured, and picked apart as boot. This is understandable—boot times represent an effective proxy for overall system performance and we all know the boot experience is an incredibly important thing for us to get right for customers. Data shows that 57% of desktop PC users and 45% of laptop users shut down their machines rather than putting them to sleep. Overall, half of all of users shut down their machines rather than putting them to sleep.



Qualitatively, people say they prefer to shut down because they want to have their PC completely “off” so that it uses no power – either to preserve battery life or to reduce their energy use. Hibernate is also a good option for this since it similarly has no power draw, and many people really like it. However, it’s clearly not for everyone, since one of the other things we’ve heard is that many people want to turn their PCs on and have it be a “fresh start” rather than running all of the stuff from their previous session. Sleep/resume is the best option for fast on/off transitions on today’s PCs, but it still consumes some power in order to preserve the contents of RAM, which means battery drain – even if it’s only a little bit on a well optimized system. All of this is happening with the backdrop of how we all use our mobile phones today, which is almost never restarting them, and always using what feels closest to a sleep-like state.

Our challenge then, was to design a way to meet all of these desires on today’s PCs without requiring some special new hardware. These were our goals:

- Effectively zero watt power draw when off
- A fresh session after boot
- Very fast times between pressing the power button and being able to use the PC.

In Windows 7 we made many improvements to the boot path, including parallel initialization of device drivers, and trigger-start services, but it was

clear we'd have to get even more creative (and less incremental) if we hoped to get boot performance anywhere close to fast enough to meet all of these needs.

Our solution is a new fast startup mode which is a hybrid of traditional cold boot and resuming from hibernate.

Before I go into exactly how it works though, a little background is probably helpful on how shutdown and boot works today in Windows 7.

Shutdown entails:

1. The user initiates a shutdown by selecting "shut down" from the Start menu, or by pressing the power button; or an application initiates shutdown by calling an API such as [ExitWindowsEx\(\)](#) or [InitiateShutdown\(\)](#).
2. Windows broadcasts messages to running applications, giving them a chance to save data and settings. Applications can also request a little extra time to finish what they're doing.
3. Windows closes the user sessions for each logged on user.
4. Windows sends messages to services notifying them that a shutdown has begun, and subsequently shuts them down. It shuts down ordered services that have a dependency serially, and the rest in parallel. If a service doesn't respond, it is shut down forcefully.
5. Windows broadcasts messages to devices, signaling them to shut down.
6. Windows closes the system session (also known as "session 0").
7. Windows flushes any pending data to the system drive to ensure it is saved completely.
8. Windows sends a signal via the ACPI interface to the system to power down the PC.

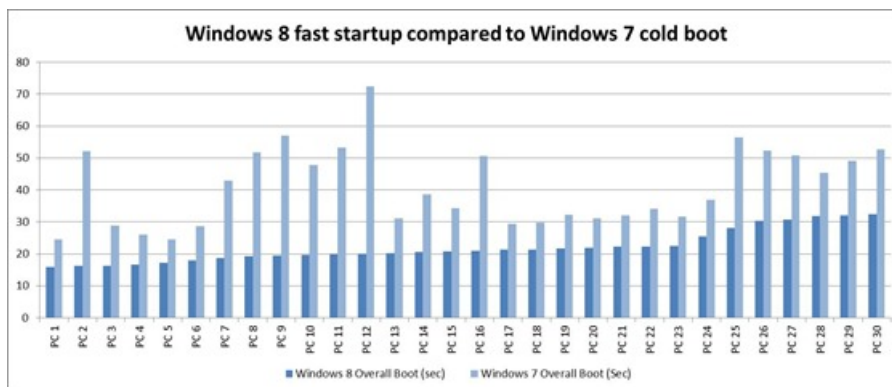
And boot entails:

1. After pressing the power button, the PC's firmware initiates a Power-On Self Test (POST) and loads firmware settings. This pre-boot process ends when a valid system disk is detected.
2. firmware reads the master boot record (MBR), and then starts Bootmgr.exe. Bootmgr.exe finds and starts the Windows loader (Winload.exe) on the Windows boot partition.
3. Essential drivers required to start the Windows kernel are loaded and the kernel starts to run, loading into memory the system registry hive and additional drivers that are marked as BOOT_START.
4. The kernel passes control to the session manager process (Smss.exe) which initializes the system session, and loads and starts the devices and drivers that are not marked BOOT_START.
5. Winlogon.exe starts, the user logon screen appears, the service control manager starts services, and any Group Policy scripts are run. When the user logs in, Windows creates a session for that user.
6. Explorer.exe starts, the system creates the desktop window manager (DWM) process, which initializes the desktop and displays it.

There are a lot more specific details here, if anyone wants to go deeper: <http://msdn.microsoft.com/en-us/windows/hardware/gg463386>

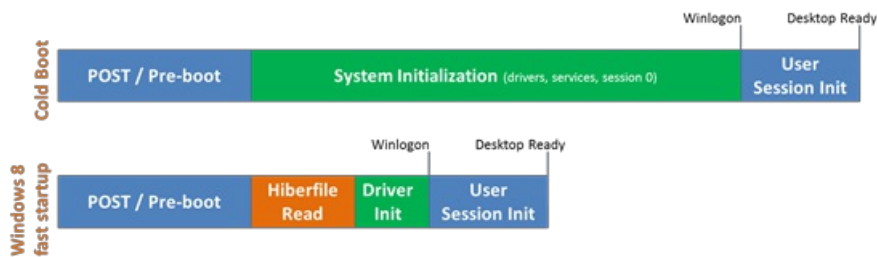
The key thing to remember though is that in a traditional shutdown, we close all of the user sessions, and in the kernel session we close services and devices to prepare for a complete shutdown.

Now here's the key difference for Windows 8: as in Windows 7, we close the user sessions, but instead of closing the kernel session, we hibernate it. Compared to a full hibernate, which includes a lot of memory pages in use by apps, session 0 hibernation data is much smaller, which takes substantially less time to write to disk. If you're not familiar with hibernation, we're effectively saving the system state and memory contents to a file on disk (hiberfil.sys) and then reading that back in on resume and restoring contents back to memory. Using this technique with boot gives us a significant advantage for boot times, since reading the hiberfile in and reinitializing drivers is much faster on most systems (30-70% faster on most systems we've tested).



*Sample of test results from System Integration Test lab systems.
Click to see a bigger version of this chart.*

It's faster because resuming the hibernated system session is comparatively less work than doing a full system initialization, but it's also faster because we added a new multi-phase resume capability, which is able to use all of the cores in a multi-core system in parallel, to split the work of reading from the hiberfile and decompressing the contents. For those of you who prefer hibernating, this also results in faster resumes from hibernate as well.



*Representation of different phases between cold boot and fast startup.
Click to see a bigger version of this chart.*

It's probably worth mentioning quickly how we treat the hiberfile—if you read this and immediately went and did a `dir /s /ah hiberfile.sys` you would have found that it's a pretty big file on disk. The hiberfile is sized by default at 75% of physical RAM. The file is essentially a reservation for hibernation data that will be written out as the system is dropping into hibernation. Typically much less space is actually used, and in the case of our fast startup usage, it's typically ~10-15% of physical RAM but varies based on drivers, services, and other factors. The system also treats the hiberfile slightly differently than other files on disk, for example, the Volume Snapshot service ignores it (a small performance benefit.) You can disable hibernation and reclaim this space by running `powercfg /hibernate off` from an elevated command prompt. But be aware that if you do this, it will disable hibernation completely, including some nice capabilities like fast startup as well as hybrid sleep, which allows desktop systems to do both a sleep *and* hibernate simultaneously so if a power loss occurs you can still resume from the hibernated state. You can also run `powercfg /hibernate /size` and specify a value between 0 and 100 for the percentage of physical RAM to reserve for the hiberfile – but be careful! Specifying too small a size can cause hibernation to fail. In general, I recommend leaving it enabled at the default value unless you're working on a system with extremely limited disk space.

Another important thing to note about Windows 8's fast startup mode is that, while we don't do a full "Plug & Play" enumeration of all drivers, we still do initialize drivers in this mode. Those of you who like to cold boot in order to "freshen up" drivers and devices will be glad to know that is still effective in this new mode, even if not an identical process to a cold boot.

This new fast startup mode will yield benefits on almost all systems, whether they have a spinning HDD or a solid state drive (SSD), but for newer systems with fast SSDs it is downright amazing. Check out the video below to see for yourself:

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

One thing you'll notice in the video was how fast the POST handoff to Windows occurred. Systems that are built using Unified Extensible Firmware Interface (UEFI) are more likely to achieve very fast pre-boot times when compared to those with traditional BIOS. This isn't because UEFI is inherently faster, but because UEFI writers starting from scratch are more able to optimize their implementation rather than building upon a BIOS implementation that may be many years old. The good news is that most system and motherboard manufacturers have begun to implement UEFI, so these kinds of fast startup times will be more prevalent for new systems.

Of course, there are times where you may want to perform a complete shutdown – for example, if you're opening the system to add or change some hardware. We have an option in the UI to revert back to the Windows 7 shutdown/cold boot behavior, or since that's likely a fairly infrequent thing, you can use the `new /full` switch on shutdown.exe. From a cmd prompt, run: `shutdown /s /full /t 0` to invoke an immediate full shutdown. Also, choosing Restart from the UI will do a full shutdown, followed by a cold boot.

Boot work is mostly owned by our Kernel Platform Group, but a number of teams came together in Windows 8 to make changes across the OS to support this new mode, plus other exciting boot changes that we'll talk about very soon. We're really enjoying the boot performance of Windows 8 in our internal use, and are looking forward to you being able to try it for yourselves so you can let us know what you think.

Gabe Aul

Experiencing Windows 8 touch on Windows 7 hardware

Steven Sinofsky | [2011-09-13T20:00:00+00:00](#)

As many no doubt know by now, we've released a Developer Preview of Windows 8. This is a build of the software designed for developers to begin developing apps using the new capabilities of Windows 8 and our new tools. The build is substantially complete across subsystems but is not a beta by any measures that we use to define a beta. The rich diversity and creativity you find within the ecosystem of hardware/software/peripherals for Windows can lead to a situation where different parts of an overall experience are available at different times. Windows 8 is in a developer preview state now, but there is not yet a broad set of PC hardware upon which to experience some of the new, hardware-specific, aspects of Windows 8. Of course Windows 8 is designed from the ground up to be an amazing upgrade (or clean install) for any PC that runs Windows 7. But we know many folks are anxious to try out some of the new scenarios and form factors that will also make Windows 8 shine.

There are no Windows 8 PCs yet, but there are PCs we have been using in our labs and that our team has been using to test the new capabilities of Windows 8. We work closely with PC makers to test early systems and will continue to do so. But we also wanted to let you know about the systems we have been working with that have touch capabilities and allow you to experience these new aspects of Windows 8. This is not an exhaustive list, and it is definitely not a recommended, certified or "logo" list. It is merely a list of machines we have experience with, and so we want to share that experience with you. We think if you're looking to experience some of the latest scenarios and aspects of Windows 8 before there are purpose-built Windows 8 machines, this is a good start.

So, this post focuses on the touch interface aspects of PCs designed for Windows 7, which also work with Windows 8. Grant George, corporate vice president for all of Windows test, and Jeff Piira, a test manager on our HIP team, authored this post together.

--Steven

Over many years in the PC ecosystem, we have seen the machine <--> human interface evolve many times. The first era was the command-line interface, where we only needed the keyboard to be able to type out commands on a black screen. The next era was the graphical user interface, where we enabled support for a mouse, in addition to a keyboard. The mouse made it easier for users to interact with "windows," icons, menus, and pointers. Recently we have seen more and more emphasis on a natural user interface, where users interact not just with a keyboard or mouse, but also with touch. The way people interact with technology has evolved, such that touch interfaces are now more common than ever, and they can be found today across a myriad of devices ranging from 3-inch mobile phones to book readers, notebooks, large desktop displays, point of sale devices, kiosks and more. Touch is everywhere and it's here to stay.

In Windows 8, we are taking the next step in adopting touch as a truly first-class input mechanism by evolving not only our UI, but many other platform elements as well. The goal of this blog post is not to introduce the overall story of touch for Windows 8 (that will come later), but to tell the story of hardware, how it is evolving, and what we think Windows 8 will bring to the ecosystem of touch.

Every touch interface has its own challenges to develop and perfect. However, to the end user, what matters most is the smooth, responsive, and natural experience of interacting with a device using touch. This sensation of performance is something we have prioritized heavily in Windows 8.

Performance of touch is not an easy thing to quantify, and there are many elements to consider. The speed at which the software input stack responds to the hardware is a primary factor. As much as we can make advances in optimizing the software, hardware plays a huge role in the "feel" of an immersive touch experience. For Windows 8, one of our approaches is to partner deeply with industry leaders on this aspect of touch, something which has paid off tremendously.

Another aspect of change in Windows 8 has been how we have approached the touch experience. Early on we decided to concentrate on ensuring the key user experiences are not only designed, but are fully optimized for touch. While this decision may seem trivial, it fundamentally changed how we evaluate Windows 8 on existing hardware and how we communicate with hardware partners. All of our requirements and tests are built off the user experiences rather than specific hardware centric capabilities. This helps to ensure that there is no gap between what the hardware can do and what the software expects.

So how do we define a good touch experience in Windows 8?

- Panning and touch response are precise and smooth (we call this "stick to your finger" panning).
- Touch visualization is direct and immediate.
- Targeting UI with your fingers is seamless and confident.
- Typing on the screen is quick, efficient and responsive.
- Touch application experiences are consistent. Touching these applications will work the same regardless of the device they are run on.

Touch hardware coverage

As the market for touch-enabled Windows PCs is broad, we focused our efforts on existing in-market devices to guide our initial development. Here are some of the newer Windows 7 systems that we use most commonly:

- HP Elitebook 2740p and 2760p convertible
- ASUS EP121 tablet
- Dell Inspiron Duo convertible
- Lenovo x201, x220t convertible
- 3M M2256PW 22" display

We also test Windows 8 on a broader set of in-market systems. Touch quality is not only about the touchscreen and its relationship to the user. When we're testing complete systems, things like bezel design, graphics, CPU and cover glass can impact the Windows 8 touch experience as well. We are committed to supporting the hardware that is running with Windows 7 today and working hard to bring a good experience to our customers who upgrade. As we continue through our development cycle on Windows 8, we will update this blog and call out how progress is coming with existing in-market systems.



Below is a list of the devices we currently have in our test labs.

3M M2256PW	Dell ST2220Tc	HP TouchSmart_IQ526t
Acer Aspire 1420p (PDC)	Dell Studio 1747	Lenovo C320
Acer Aspire 1825PT	Dell Studio One	Lenovo S10-3T
Acer Aspire 5738PG	Dell SX2210T	Lenovo ThinkPad T410S
Acer Aspire Z5610	Elo 1522L	Lenovo ThinkPad X201T
Acer Iconia	EXOPC	Lenovo ThinkPad X220T
Acer T230H	Fujitsu Lifebook T4310	Lenovo ThinkPad_X60
Acer W500	HP Compaq L2105TM	NEC MultiSync LCD175M
ASUS EP 121	HP EliteBook 2740P	Planar PX2230MW
ASUS TCA70	HP Mini 5102	Samsung Series 7 XE700T1A
Dell Inspiron 2305	HP Pavilion_tx2000	Sony V J series

Dell Inspiron Duo	HP Pavillion DV3T-2000	Sony V L series
Dell Inspiron One 2305	HP Tablet 500	Sony VPCL113FX/B
Dell Latitude E6420	HP TouchSmart 610	Sony VPCL-218FW
Dell Latitude XT	HP TouchSmart IQ500	Toshiba Portege_M700
Dell ST2220T	HP TouchSmart TX2Z	Toshiba Sattellite Har/Kar

Touch tests

Here are a couple of examples of tests that we run to see how hardware and software works together. The first test covers new Windows 8 features that you access by swiping a finger in from the edges of the screen, like Search, Share, and Settings. (We will talk more about these features in future posts.)



To get the best experience when swiping in from the edge, touch must be responsive across the entire active screen starting at pixel 1 on each side, so we've developed tools to ensure that swipes are always properly detected at the edges of the screen.

To ensure a smooth panning experience, we have requirements for the latency of hardware response and panning with touch. We use a high-speed camera to measure input lag or delay between when a user touches the screen and when that action is reflected on the display. The less lag or separation between the user's finger and the object being dragged the better!

Building new touch hardware for Windows 8

Keeping the user experience at the top of the requirements, Windows 8 will kick off a new generation of computing devices, and it is only natural that touchscreen technologies will evolve with it. Our goal on the Windows team is to work in lock step with external hardware partners in the development of new hardware that will more fully support Windows 8 requirements, and ultimately provide the smooth, responsive, and natural touch experience that Windows users expect. Our continuing work with our touch hardware partners, suppliers, IHV's (independent hardware vendors), and PC manufacturers will help us together deliver an immersive and intuitive touch experience in Windows 8.

--Grant and Jeff

Welcome to Windows 8 – The Developer Preview

Steven Sinofsky | [2011-09-13T09:05:00+00:00](#)

If you've been following this blog, then you know today is a big day for the Windows team. At the BUILD conference we are about to preview Windows 8. There's a ton to see in the product and so we'd really encourage everyone to check out the available streams on <http://buildwindows.com>, where we will webcast the keynote. The BUILD conference this week is focused on developers and hardware partners, and there are over 100 sessions (all of which will be available from the link above within about a day of the scheduled presentation time). In that sense it is good to keep in mind that today is the launch of the developer opportunity for Windows, not the launch of a product (and certainly not the launch of new devices).

Windows 8 represents a reimagining of Windows from the chipset to the experience. Since this is a week focused on developers, we also detailed the bold underpinnings of the re-imagining of the Windows platform, tools, and APIs. We will show off the opportunity to build applications for all of the customers of Windows 8, no matter what type of PC they have—from tablets to laptops to convertibles to desktops. We will show the brand new tools that allow you to code Metro style applications in HTML5/JavaScript, C/C++, and/or C#/XAML. The investments you have made as developers in all of these languages carry forward for Windows 8, which lets you choose how to best make use of the Windows 8 system services. We talked about Windows 8 being a no-compromise OS for end-users, and it is also a no-compromise platform for developers.

Many are interested in Windows 8 for ARM processors. Everything we showcased today at BUILD also runs on the ARM-based Windows PCs being created by ARM partners and PC manufacturers. Windows 8 running on ARM will ultimately be available with ARM-based hardware that you can purchase. ARM requires a deeper level of integrated engineering between hardware and software, as each ARM device is unique, and Windows allows this uniqueness to shine through. The new development tools enable you to start today to build Metro style applications that will seamlessly run on x86 (32 and 64 bit) or ARM architectures. Even if you use native C/C++ code, these tools will enable Metro style apps to target specific hardware if you choose. As new PCs become available for testing, PC manufacturers will develop seed programs for developers.

You probably want to try out the preview release—and you can. Starting **later tonight** you can download the Windows 8 Developer Preview. This includes a **64-bit (x64) build with development tools** to build apps, and a **32-bit (x86) or 64-bit (x64) build without development tools**. The releases also include a suite of sample applications (please note these are merely illustrations of potential apps, not apps that we intend to ship with Windows 8). The ISOs are linked to from <http://dev.windows.com>.

Upgrade from Windows 7 installation is not supported for pre-release code; only clean installs are supported. Reminder: this is a developer preview release and is not meant for production. It is not a beta release. We will be updating the release with various quality updates and drivers over the coming weeks/months just to exercise our overall update and telemetry mechanisms.

We've got a lot more blogging to do. So stay tuned. This blog continues to be a big part of the development process. Now we have a lot more shared context, and so we expect folks commenting on posts to be running the Preview so we share in the context of the release. Let's keep comments focused on the topic at hand and we'll pay attention for potential new topics. We know there will be a lot—that comes from reimagining a product used by a billion people!

--Steven

UPDATE (10:10 am PST) - Slight correction inserted **above**.

UPDATE 2 (11:30 am PST) - I bet a lot of you heard about the machine that attendees at the conference received (this was a limited production run and is not available for sale). It is pretty cool. Keep in mind, Windows 8 is a no-compromise OS, which means you do not need to have a tablet or touch-capable machine to experience Windows 8. Mouse and keyboard are first class in the whole experience. If you are curious, a little later today we will post some of the touch hardware we have experienced in our labs so you can see what existing Windows 7-based hardware experiences we have. Keep in mind there are no PCs designed for Windows 8 yet—that's a big part of the BUILD conference—gearing up for newly designed Windows 8 PCs of all types.

NOTE: The preview build does not include every feature shown this morning. Shown but not in the Developer Preview release include the Windows Store, Windows Live Metro style apps, and some of the user interface features. The focus of the preview is the API and development tools for building Metro style apps.

Best place to discuss Windows 8

Steven Sinofsky | [2011-09-14T06:00:00+00:00](#)

I wanted to offer a pointer to the best place to have discussions about Windows 8. Please head over to the forums we have set up to discuss the product and answer questions. As we have seen, the commenting mechanism is not a great place to have a wide variety of topics different than the topic of the post while keeping track of the discussion. We set up this short URL to be easy to remember: <http://win8.ms/forums> gets you to the general forum.

-Steven

Metro style browsing and plug-in free HTML5

Steven Sinofsky | [2011-09-14T17:47:00+00:00](#)

One of the first things a lot of folks will try after installing the developer preview of Windows 8 will be the IE10 browser—the most used tool in Windows. IE 10 in the preview is Platform Preview 3 of IE 10. You can read on the [IE blog](#) about the HTML 5 engine work we're doing. This post is about a big change in Metro style IE, which is the plug-in free experience. In Windows 8, IE 10 is available as a Metro style app and as a desktop app. The desktop app continues to fully support all plug-ins and extensions. The HTML5 and script engines are identical and you can easily switch between the different frame windows if you'd like. Metro style IE provides all the main navigation keyboard shortcuts and mouse support you've come to expect—creating tabs, moving between tabs, closing tabs, entering addresses, searching, and more. I'm using this browser full-time, and given the amount of time I spend in Windows Phone, the same experience and use of touch is definitely a plus. But you can decide on what works best for you, and not compromise. Dean Hachamovitch, who leads the IE team, wrote this post.

—Steven

For the web to move forward and for consumers to get the most out of touch-first browsing, the Metro style browser in Windows 8 is as HTML5-only as possible, and [plug-in free](#). The experience that plug-ins provide today is not a good match with Metro style browsing and the modern HTML5 web.

Running Metro style IE plug-in free improves battery life as well as security, reliability, and privacy for consumers. Plug-ins were important early on in the web's history. But the web has come a long way [since then](#) with HTML5. Providing compatibility with legacy plug-in technologies would detract from, rather than improve, the consumer experience of browsing in the Metro style UI.

The reality today is that sites are already rapidly engineering for a plug-in free experience. Google, for example, recently launched their [HTML5 YouTube](#) site for phones. A previous [IE blog post](#) discussed how plug-in free sites are becoming more mainstream, and what sites can do to run plug-in free. We examined the use of plug-ins across the top 97,000 sites world-wide, a corpus which includes local sites outside the US in significant depth. Many of the 62% of these sites that currently use Adobe Flash already fall back to HTML5 video in the absence of plug-in support. When serving ads in the absence of plug-ins, most sites already perform the equivalent of this fallback, showing that this approach is practical and scalable. There's a steep drop-off in plug-in usage after Flash, with one control used on 2% of sites and a small collection of controls used on between 0.5% and 0.75% of sites.

On Windows 8, consumer sites and “line of business” applications that require legacy ActiveX controls will continue to run in the desktop browser, and people can tap “Use Desktop View” in Metro style IE for these sites. For what these sites do, the power of HTML5 makes more sense, especially in Windows 8 apps.

Plug-in free browsers today already deliver great experiences with well-authored HTML5 content. These experiences get even better with touch in Metro style IE.

Thanks –

Dean

P.S. Below, you can see how IE adjusts its behavior site by site, as developers make the transition and stop relying on plug-ins for functionality available in HTML5 (for example, video or [XHR](#)). Most sites work fine in IE without plug-ins; others work fine in IE when IE identifies itself as another browser or runs the site in a different mode. As we work with the web developer community, IE continues to use the Compatibility View (CV) list to keep sites working for consumers.

Here's a fragment of the CV list that went live with the developer preview build of Windows 8 at the BUILD conference:

```
<?xml version="1.0" encoding="utf-8" ?>
<iecompatlistdescription>
  <version>1152921504606910005</version>
  <ttl>1</ttl>
  <domain docMode="EmulateIE7">monster.com</domain>
  <domain docMode="EmulateIE7">pbskids.org</domain>
  <domain docMode="EmulateIE8" uaStringImmersive="iPad">nate.com</domain>
  <domain docMode="EmulateIE8" versionVector="8"
uaString="8">bankofamerica.com</domain>
  <domain docMode="EmulateIE8" versionVector="8" uaString="8">wellsfargo.com</domain>
  <domain docMode="EmulateIE8">7-eleven.com</domain>
  <domain docMode="EmulateIE9" versionVector="9"
uaString="9">sportsillustrated.cnn.com</domain>
  <domain docMode="EmulateIE9">lowes.com</domain>
  <domain docMode="IE9">github.com</domain>
  <domain featureSwitch="createElementWithMarkup:false">dodge.com</domain>
```

```
<domain featureSwitch="createElementWithMarkup:false">krispykreme.com</domain>
<domain featureSwitch="createElementWithMarkup:false">youtube.com</domain>
<domain uaStringImmersive="Firefox 5">tv.slashgear.com</domain>
<domain uaStringImmersive="iPad">mashable.com</domain>
<domain uaStringImmersive="iPad">tested.com</domain>
<domain>about.zappos.com</domain>
<domain>airborne.gogoinflight.com</domain>
<domain>aol.com</domain>
</iecompatlistdescription>
```

Metro style browsing: one engine, two experiences, no compromises

Steven Sinofsky | [2011-09-14T10:50:00+00:00](#)

We are very happy to have received such a warm welcome from developers yesterday as we kicked off a pretty big opportunity with Windows 8. Our focus on B8 now moves to the Developer Preview and what is in it and how it evolves. We hope those choosing to participate in the blog are installing and using the build. While it is early and focused on developers, it is also quite fun to use. I am doing all my posts from the conference via the Samsung Preview tablet!

We wanted to talk some about Metro style browsing and the work we have done to deliver a truly chromeless browsing experience. We are very much focused on HTML5 and standards support with the very best performance and reliability along with IE's well-regarded safety features. We also continue to evolve and deliver the desktop experience that uses the same HTML5 technologies. That's how we deliver a no-compromise browsing experience with IE10. This post describes the IE10 platform preview 3 in Windows 8 developer preview. Dean authored this post. --Steven

There's so much more to moving the browser experience forward than just putting it on a touch device. To deliver the best browsing across any form factor running Windows 8, we re-imagined the experience of the web browser and the underlying architecture.

Our approach in Windows 8 starts with one great HTML5 browsing engine that powers two different experiences. The single engine provides strong support for web standards, hardware-accelerated performance, security, privacy, and more. Then, we built two experiences on top of that engine: a new Metro style experience as well as a more traditional, current-generation desktop browser with tabs and relatively minimal "chrome."

The result is that you can carry one device and it provides both experiences without compromise, acting as an immersive tablet and a flexible laptop. You can have both experiences on a powerful multi-monitor desktop as well – no compromise.

You'll get the best immersive, touch-first experience of HTML5 websites with the Metro style browser in Windows 8. If you prefer more traditional window and tab management, you have that in improved form with IE on the desktop. Both run on the same IE10 engine.

While building the entirely new Metro style experience, a funny thing happened: we realized it might just be a better way to browse even if you have a big screen, a desktop computer, and a mouse and keyboard. While it's a touch-first browser, it works well with keyboard and mouse or trackpad. If you've been doing more and more browsing on your phone, then you've likely become used to having almost no "chrome," far more visual tab management, and a more immersive, less "desktop"-y, more manual browsing experience. You might find that you prefer the new Metro style experience of Internet Explorer 10 to the desktop.

Just as a reminder: the Metro style browser in the current Windows Developer Preview is for developers, not consumers. There is work ahead with the developer community to make the experience consumer-ready (for example, having sites update older, out-of-date [libraries](#) that don't work well with IE10, or making sure that sites that already run plug-in free for other devices work that way with the Metro style browser).

One engine, two experiences

Because great HTML5 support on both desktop IE and the Metro style IE is so important, we adapted the IE10 engine's architecture to power both experiences. The two experiences share browsing history, typed addresses, settings, and more. The common engine delivers a consistently fast, safe, and powerful experience for today's sites as well as Metro style applications:

- **Performance.** Metro style IE has the same industry-leading performance as IE on the desktop. This includes full hardware acceleration of graphics, video, and audio, compiled JavaScript, and new layout and formatting engine optimizations for touch responsiveness.
- **Safety.** Metro style IE also has the same industry-leading security, privacy, and reliability features as IE on the desktop. This includes SmartScreen, XSS filtering, and InPrivate browsing.
- **HTML5.** Metro style IE delivers the same commitment to a richer HTML5 web programming model as IE on the desktop. At BUILD, we introduce new support for CSS Text Shadow, CSS 3D Transforms, IndexedDB, Web Sockets, HTML5 File APIs, HTML5 History, hyphenation, CSS Transitions and Animations, and HTML5 Application Cache, in addition to the other new features previously shown in IE10 Platform Previews.

You can read more about desktop IE and technical details of the underlying Trident browser engine and Chakra JavaScript engine on the [IE Blog](#).

Re-imagining the browser as a Metro style app

When we re-imagined the browser as a Metro style app, we saw a totally new way to move the web forward.

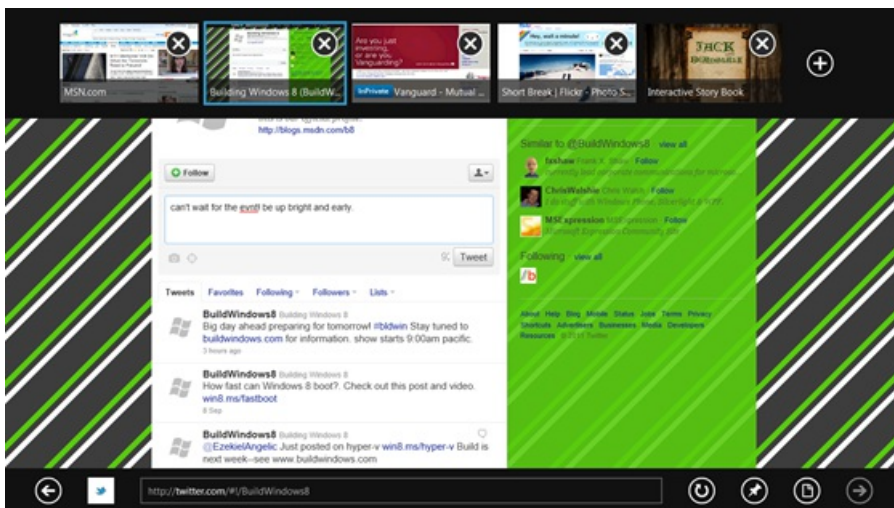
For example, the Metro style is immersive. For a web page in Metro style IE, there are no visual distractions at all. You can use standard gestures to get to functionality that otherwise distracts you from the web. You can search and share from Metro style IE using "charms," just as you do in other Metro style apps. You can use the Devices charm, for example, to play and project videos from web pages to external devices. You can "snap" IE side-by-side with another Metro style application. Using websites and Metro style apps together is easy because we built them to work together.

The new Metro style is much more than a visual design. For example, it enables you to get to your important sites with less typing. You see a touch-friendly, visual list of your frequent and pinned websites when you open a new tab or bring up the address bar:



A touch-friendly, visual list of your frequent and pinned web sites in the Metro style browser

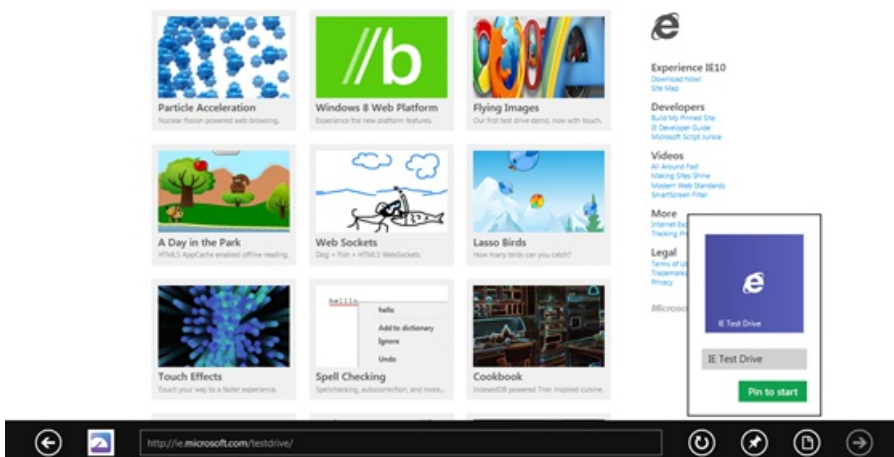
Typing in the address bar filters this list. When you bring up your open tabs, the address bar is immediately available so you can get to the site you want if it's not already open:



Open tabs and the address bar in the Metro style browser

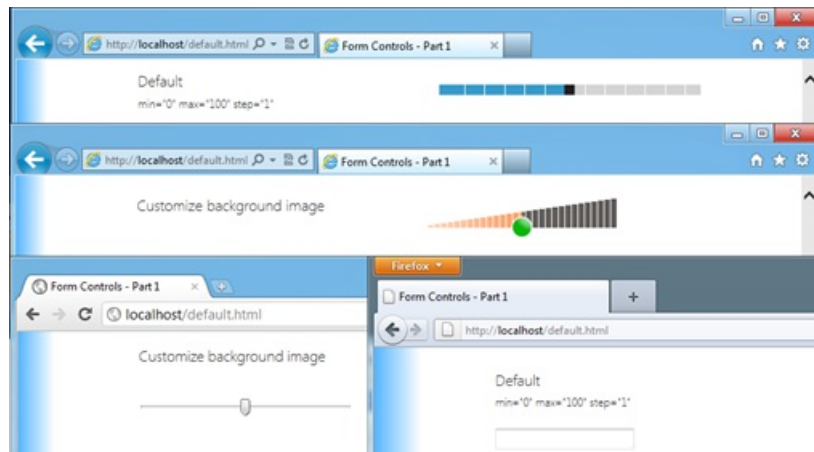
Spell-checking and auto-complete also reduce how much typing you need to do as you use sites. Notice that you can even browse with InPrivate and Tracking Protection.

To make the websites you care about easy to get to and work with, you can pin them to the new Windows 8 Start screen, the same way you pin Metro style apps there.



Pinning a site to the Start screen

IE10 draws controls (like check boxes and radio buttons) with more space around them to be more touch-friendly. Developers can make their web pages much more touch-friendly using standard web patterns like CSS to style controls. For example, below you see an HTML5 [range input type](#) control with a custom background image:



HTML5 control with custom background image in IE10 (top), in Chrome (bottom left), and in Mozilla Firefox (bottom right)

Metro style IE supports touch panning and pinching to zoom. Pages are fast and fluid like the rest of the system because the HTML5 engine takes advantage of full hardware acceleration. You can see this in the scrolling and panning performance.

Here's a demo that shows the performance as well as other parts of the browsing experience:

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

Metro style IE: looking ahead

We see new opportunities to move the web forward as a result of this new browser experience. There is work ahead with site developers to take full advantage of it. Touch is very important, and so is the immersive experience. The opportunities here are huge, given how much time people spend on the web today.

The current Windows Developer Preview is for developers, not consumers. For example, there's work ahead to make sure that sites that already run [plug-in free](#) for other devices can work that way in IE10. Similarly, some sites may need to update older, out-of-date [libraries](#) that don't work well with IE10. Including the current desktop style browser to deliver the compatibility that people expect from a Windows release is also important.

Pessimists may criticize what they will call "two browsers." There's only one browsing engine, which you can use with two different "skins." Over time, the Metro style experience will serve more and more mainstream browsing scenarios, even as the desktop browser continues to play a key role in many people's lives. You can set your default to either style, seamlessly switch between them, or use any other browser you choose on Windows 8.

Pessimists may also try to debate how much they need or don't need different aspects of each browser style. If you want to stay permanently immersed in that Metro world, you can. You won't even see the desktop, and Windows will not load the code unless you explicitly choose to launch it. But if you do see value in the desktop experience – in precise control, in powerful windowing and file management, in compatibility with plug-ins – those capabilities are just a finger tap away. In the architecture and design, we've been very deliberate to provide a no-compromise experience for your needs, even if you carry only one device. So, browse where you're comfortable.

Dean Hachamovitch

Protecting you from malware

Steven Sinofsky | [2011-09-15T13:30:00+00:00](#)

One of the things we talk quite a bit about with Windows 8 is making sure Windows is a safe, secure, and reliable computing environment. We have always provided a broad range of solutions for achieving these goals and work closely with a broad range of industry partners. We continue to enhance these capabilities with Windows 8 while making sure you always have choice and control over how to protect and manage your PC. With Windows 8 we are extending the protections provided by Defender to address a broader range of potential threats. Jason Garms, the group program manager of our reliability and security team authored this post that represents work across several teams. --Steven

I'm excited to share with you some investments we are making in Windows 8 to better protect you against the constantly changing landscape of malicious software ("malware"). In this blog I will talk about enhancements to mitigation features that help protect you against exploits used by malware, improvements to Windows Defender to provide you with real-time protection from all categories of malware, and the use of URL and application reputation to help protect you against social engineering attacks.

A view of the current landscape

Criminal attacks continue to evolve and malware has become their standard weapon against anyone who uses the Internet—on traditional form-factor devices, as well as on mobile devices like tablets and phones. Malware targets all operating systems and browsers, and in recent years, criminal attacks against applications have increased substantially.

Criminals also use social engineering to trick you into performing actions that put you at risk. An increasingly common social engineering strategy uses [online advertising campaigns](#) to lure you to a site that installs malware on your computer.

An [economy](#) has developed around building reliable vulnerability exploits, which criminals buy to help distribute their malware. Criminals make money from their malware, so they invest in ways to keep it alive such as producing a higher quantity of malware, updating it more frequently—e.g. multiples times each day—and increasing its size and complexity. Some malware is as complex as commercial applications.

Secure by design

We use the [Security Development Lifecycle \(SDL\)](#) to build Windows with the best security design, development and testing practices available. Some highlights include:

- **Threat modeling and security design reviews.** During the design process we consider how criminals might seek to attack features and scenarios, and incorporate this analysis into our designs.
- **Writing secure code.** Training and code quality tools help to prevent common coding issues from entering the Windows source code.
- **Penetration testing.** Security engineers take an attacker's perspective when reviewing a completed set of features that make up a scenario.
- **Security code reviews.** Security engineers provide additional security-oriented code reviews for highly sensitive components.
- **Security tools.** Tools continuously updated with the latest state of the art in finding and exploiting software provide a scalable solution to improve existing code.

Making it harder to create an exploit on Windows 8

With Windows XP SP2, we began creating defenses called [mitigations](#) that make it difficult to develop reliable exploits for security vulnerabilities. Each subsequent version of Windows has continued to expand and improve on these mitigations, because a single mitigation feature can break an entire class of exploits. Windows 8 includes mitigation enhancements that further reduce the likelihood of common attacks. Some of these improvements include:

- **Address Space Layout Randomization (ASLR).** [ASLR was first introduced in Windows Vista](#) and works by randomly shuffling the location of most code and data in memory to block assumptions that the code and data are at same address on all PCs. In Windows 8, we extended ASLR's protection to more parts of Windows and introduced enhancements such as increased randomization that will break many known techniques for circumventing ASLR.
- **Windows kernel.** In Windows 8, we bring many of the mitigations to the Windows kernel that previously only applied to user-mode applications. These will help improve protection against some of the most common type of threats. For example, we now prevent user-mode processes from allocating the low 64K of process memory, which prevents a whole class of kernel-mode NULL dereference vulnerabilities from being exploited. We also added integrity checks to the kernel pool memory allocator to mitigate kernel pool corruption attacks.
- **Windows heap.** Applications get dynamically allocated memory from the Windows user-mode heap. Major redesign of the Windows 8 heap adds significant protection in the form of new integrity checks to help defend against many exploit techniques. In addition, the Windows heap now randomizes the order of allocations so that exploits cannot depend on the predictable placement of objects—the same principle that makes ASLR successful. We also added guard pages to certain types of heap allocations, which helps prevent exploits that rely on overrunning the heap.

- **Internet Explorer.** “Use-after-free” vulnerabilities represented nearly 75% of the vulnerabilities reported in Internet Explorer over the last two years. For Windows 8, we implemented guards in Internet Explorer to prevent an attacker from crafting an invalid virtual function table, making these attacks more difficult. Internet Explorer will also take full advantage of the ASLR improvements provided by Windows 8.

Keeping malware off your PC

Having effective malware protection is important for any device connected to the Internet and almost all Windows PCs sold today include a traditional antimalware solution, though it is often a time-limited or trial version.

Shortly after Windows 7 general availability in October 2009, our telemetry data showed nearly all Windows 7 PCs had up-to-date antimalware software. However, a few months later the trend started to decline month-to-month, likely reflecting antimalware trial subscriptions expiring. A year later, at least 24% of Windows 7 PCs did not have current antimalware protection. Our data also shows that PCs that become unprotected tend to stay in this unprotected state for long periods of time. And when antimalware software is even one week out of date, its ability to protect against new malware [drops significantly](#).

We believe that all Windows 8 users should be protected by traditional antimalware software that provides an effective, industry-recognized level of protection. There are a lot of great antimalware solutions available that we expect will be updated to protect Windows 8 PCs and we believe most PC makers will continue to ship Windows PCs with these solutions installed.

Windows Defender

If you don't have another solution installed, Windows 8 will provide you protection with a significantly improved version of Windows Defender.

Improved protection for all types of malware. The improvements to Windows Defender will help protect you from all types of malware, including viruses, worms, bots and rootkits by using the complete set of malware signatures from the [Microsoft Malware Protection Center](#), which Windows Update will deliver regularly along with the latest [Microsoft antimalware engine](#). This expanded set of signatures is a significant improvement over previous versions, which only included signatures for spyware, adware, and potentially unwanted software.

In addition, Windows Defender will now provide you with real-time detection and protection from malware threats using a file system filter, and will interface with Windows secured boot, another new Windows 8 protection feature.

When you use a PC that supports UEFI-based Secure Boot (defined in the [UEFI 2.3.1 specification](#)), Windows secured boot will help ensure that all firmware and firmware updates are secure, and that the entire Windows boot path up to the antimalware driver has not been tampered with. It does this by loading only properly signed and validated code in the boot path. This helps ensure that malicious code can't load during boot or resume, and helps to protect you against boot sector and boot loader viruses, as well as bootkit and rootkit malware that try to load as drivers.

The same interfaces for secured boot used by Windows Defender, as well as all APIs used by Windows Defender, are available for use by our antimalware partners to deliver additional protection to Windows customers.

- **Improved user experience.** We have designed Windows Defender to be unobtrusive for most daily usage, and will notify you only when you need to perform an action, or critical information demands your attention. Windows Defender will also use the new Windows 8 maintenance scheduler to limit interruptions.
- **Improved performance.** Traditional antimalware technologies are well known for impacting system performance. It's not uncommon that running antimalware software doubles the amount of time required for core scenarios like file copy and boot. As you read in last week's [blog entry](#), we have a lot of people working on system performance and Windows Defender dramatically improves performance on all key scenarios compared to common antimalware solutions on Windows 7, while maintaining strong protection. For example, Windows Defender with its full protection functionality enabled adds only 4% to boot time, while dramatically reducing CPU time during boot by 75%, disk I/O by around 50MB, and peak working set by around 100MB.

These same improvements benefit energy efficiency, meaning Windows Defender consumes less power, and gives you longer battery life.

We're continuing to work with antimalware partners during the Windows 8 development process so you have the best possible Windows PC experience no matter what antimalware solution you choose. We provide them with resources, such as the technical details of how we architected the performance improvements for Windows Defender, so they have the opportunity to make similar improvements to their products.

Microsoft SmartScreen for Internet Explorer and now for Windows too

Traditional antimalware software plays a critical role in defending and remediating attacks. However, reputation-based technologies can help provide effective protection against social engineering attacks before traditional antimalware signatures are available, especially against malware that pretends to be legitimate software programs.

Windows 8 will help protect you with reputation-based technologies when launching applications as well as browsing with Internet Explorer.

Since its release, the SmartScreen filter has used URL reputation to help protect Internet Explorer customers from more than 1.5 billion attempted malware attacks and over 150 million attempted phishing attacks. [Application reputation](#), a new feature added to SmartScreen in Internet Explorer 9, provides an additional layer of defense to help you make a safer decision when URL reputation and traditional antimalware aren't enough to catch the attack. Telemetry data shows 95% of Internet Explorer 9 users are choosing to delete or not run malware when they receive a

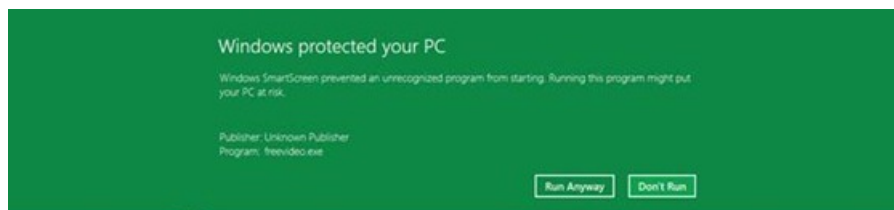
SmartScreen application reputation warning.

We understand that Internet Explorer isn't the only way you download applications from the Internet, so Windows now uses SmartScreen to perform an application reputation check the first time you launch applications that come from the Internet.

In Windows 7 when launching these downloaded applications, you get the following notification:



In Windows 8, SmartScreen will only notify you when you run an application that has not yet established a reputation and therefore is a higher risk:



The user experience for applications with an established reputation is simple and clean: you just click and run, removing the prompt you would have seen in Windows 7.

SmartScreen uses a marker placed on files at download time to trigger a reputation check. All major web browsers and many mail clients, and IM services already add this marker, known as the “[mark of the web](#),” to downloaded files.

We expect average users to see a SmartScreen prompt less than twice per year and when they do see it, it will signify a higher risk scenario. Telemetry data shows 92% of applications downloaded via Internet Explorer 9 already have an established reputation and show no warnings. The same data shows that when an application reputation warning is shown, the risk of getting a malware infection by running it is 25-70%. And SmartScreen gives you administrative controls to prevent your non-techie friends or children from ignoring these warnings.

We've seen dramatic results with this approach in Internet Explorer and we're happy to bring it to a broader set of Windows scenarios.

Here's a video that shows you Windows Defender and SmartScreen URL and application reputation in action:

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

In conclusion, we've taken a very broad approach to improving the level of protection you'll get from malware in Windows 8, including the use of SDL processes to be secure by design, the implementation and upgrading of mitigations to help protect you against exploits used by malware, improvements to Windows Defender to provide you with real-time protection against all categories of malware, and the use of URL and application reputation to help protect you against social engineering attacks.

Thanks,

--Jason Garms

Running Windows 8 Developer Preview in a virtual environment

Steven Sinofsky | [2011-09-16T13:30:00+00:00](#)

We wanted to do a quick post on compatibility with respect to virtualization technologies. We know there were some initial challenges and we've seen the community support process kick in and many people are unblocked as "how to" posts have gone up in various places. As this is a developer preview we expect to see other classes of issues related to app or device compatibility—the full breadth of our testing was not done (or intended to be done) for the Preview. Obviously this one was found early and we could have done a bit better. Sue Bohn, David Hicks, Cornel Lupu of our ACDC team (App Compat, Device Compat) authored this quick post. –Steven

We are seeing in the forums that there is a lot of interest in running the Windows 8 Developer Preview in a virtual environment. Our telemetry systems reported that approximately one-third of the early installations are on virtual machines. We apologize for not giving guidance up front on testing the Windows 8 Developer Preview in virtual machines. This blog post will provide some background and information on that topic.

For the ideal client computing experience, we recommend running Windows 8 Developer Preview natively on a dedicated computer. Windows 8 takes advantage of hardware acceleration to enable a fast and fluid user interface. If the option of a dedicated physical computer is not available, then using a dual boot setup is a great alternative that preserves your existing OS and setup. Here's a helpful [LifeHacker article on how to set up dual boot](#).

If you are not familiar with virtualization terminology, here is a quick primer. Virtual machine products allow you to run a guest OS on top of an existing host OS that runs natively on the physical machine. As many of you know, virtualization is a popular way to try out new operating system products since you don't have to dedicate a physical machine, add a spare disk, or repartition. To run the Windows 8 Developer Preview as a guest OS, you need a virtualization product that supports it.

Some virtualization products only provide a basic display driver that does not support the high performance graphics used in Windows 8. As a result, you get a noticeably slower, less responsive experience when compared with running the OS natively. The setup and configuration process can be complicated and error prone when running as a guest OS, especially if you are running it on older hardware that does not support built-in virtualization optimizations featured in the latest generations of Intel and AMD processors.

Windows 8 Developer Preview only came out a few days ago, so many of the virtualization products on the market have not yet been updated to work well with it. We are working closely with all of the major manufacturers of virtualization products to support Windows 8 as we move toward release.

Forum members are reporting success using a few products. Of the most popular options, our baseline assessment is as follows:

Functional:

- Hyper-V in Windows 8 Developer Preview
- Hyper-V in Windows Server 2008 R2
- VMware Workstation 8.0 for Windows
- VirtualBox 4.1.2 for Windows

Non-functional:

- Microsoft Virtual PC (all versions)
- Microsoft Virtual Server (all versions)
- Windows 7 XP Mode
- VMWare Workstation 7.x or older

You may be wondering why virtualization products that work today with Windows 7 do not work with the Windows 8 Developer Preview. We take compatibility very seriously. However, there are categories of software that run "very close to the metal," and deliberately take dependencies on internal data structures and intricacies of the Windows kernel. These dependencies are not typically publicly supported or exported APIs, and thus must change as Windows changes. We go to great lengths to avoid these changes, but sometimes they are necessary to enable innovation. As a result, some software will require updates when we make significant improvements to Windows. Other common categories include antimalware and security products. In this case, to improve boot performance and enable new CPU architectures, we took a new approach to high-resolution timers. Some virtualization products emulate older hardware timers that significantly compromise performance and will require updates before they can support Windows 8.

We work very closely with all software partners that build products that rely on these types of APIs and all have been informed of the changes in the Windows 8 Developer Preview. Each will choose to address future compatibility as is consistent with their business

goals. Collectively we are committed to delivering a great experience as the release audience broadens.

Sue Bohn, David Hicks, Cornel Lupu

Reengineering the Windows boot experience

Steven Sinofsky | [2011-09-20T13:00:00+00:00](#)

Phew! We're all back from BUILD and focused on our next milestone. It is fair to say we had an awesome time showing everyone Windows 8 in depth and all of our speakers and Microsoft attendees are unbelievably appreciative for the warm reception you gave the product. We know it is early still—a developer preview—and there are lots of questions. We're going to be answering them in new posts as we focus on using the Windows Developer Preview (WDP) as a baseline—so if you haven't been running it, consider it sort of like a prerequisite for many of the blog posts.

Boot is the sort of effort that gets no respect. It is either too long or all the work to make it nice and pleasant hopefully goes unnoticed since you never want to boot your machine. I remember a meeting many years ago where Bill Gates said (paraphrasing) "Boot is a one-line function call that computes a constant yet takes forever: fBoot = SystemBoot()" At the same time it seems like everything boots these days—phones, TVs, cable TV boxes, even my TV remote boots. In building Windows 8, we set out to take advantage of some new technology and revisited some old assumptions to totally rethink the boot experience. We also wanted to make it more accessible and better suited to devices without keyboards. Of course, we also did a lot of work to continue to minimize reboots altogether, but this post is about what happens when you do boot. Billie Sue Chafins authored this post. She is a long time program manager who spent many years on user interface design, and in this release she helped us to focus on the boot experience (in addition to the Metro style app sharing contract which you can learn about from BUILD [here](#)).

--Steven

With continued innovation in the hardware ecosystem, the biggest shift in firmware in 30 years, and software changes leading to boot times of ~7 seconds on machines with solid-state drives (SSDs), we decided it was time to bring the PC boot user experience into the 21st century. In a [previous post](#) by Gabe Aul, we discussed how fast startup mode will make faster boot times a reality in Windows 8. You may have noticed from his [fast boot video](#) that boot felt different – not only was it super fast, it was seamless, from the time the power was switched on all the way to the Start screen. In this post, we will explore how the team has reimagined the complete Windows boot user experience to make that possible.

If you think about the experience of powering up a PC today, first, you most likely see several circa-1980 text console screens flash by as the computer enters the Power-on Self-test (POST) phase of boot. A few seconds later, rendering is handed from the basic input/output system (BIOS) over to Windows, and you see a graphical animation before landing at the logon UI. In Windows 7 today, all of this happens over a span of about a minute on average. Now, imagine all of that flying by on your screen in approximately 7 seconds – that's a lot of transitions in a short period of time!

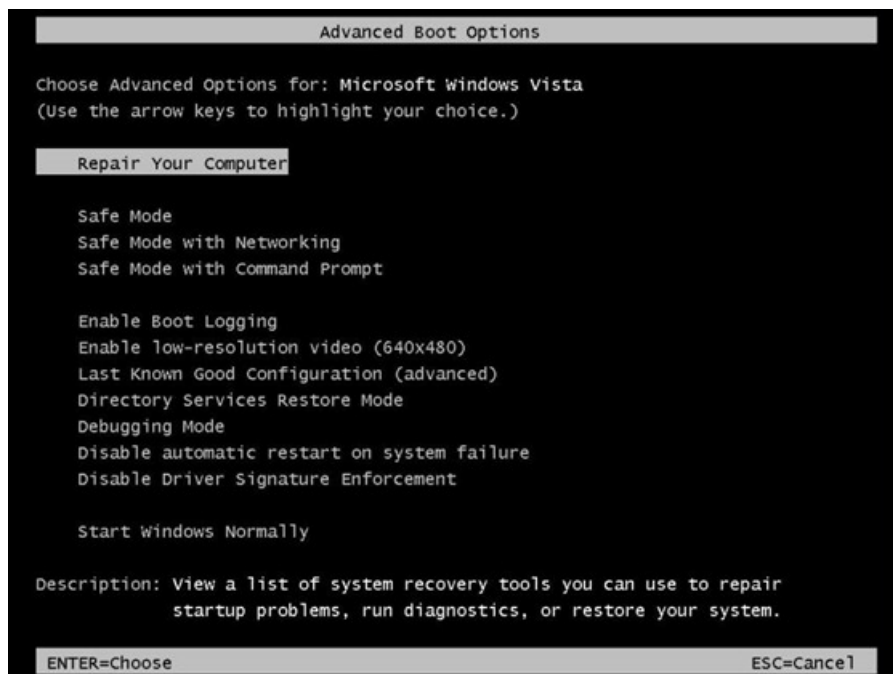


Once we realized just how fast boot was going to be in Windows 8, it was obvious that it was the perfect time to tackle the user experience to deliver something seamless, beautiful, and of consumer electronics quality.

Windows 8 will enter the market in a time when touch-first devices, such as the slate form factor, are becoming more and more prevalent. As such, we need to deliver a boot experience that is designed for touch, but works just as well for mouse and keyboard. From the beginning, we knew that it would be unacceptable to tell anyone that they'd need to go buy a physical keyboard to set up their machine or to perform recovery in the Windows Recovery Environment (Windows RE) if the PC cannot start.

Windows 8 will also enter the market in a time when the industry is shifting to the Unified Extensible Firmware Interface (UEFI) for BIOS on all new client systems. We will continue to support the legacy BIOS interface, but machines using the UEFI interface will have significantly richer capabilities. For instance, UEFI systems can render rich graphical experiences in native resolution via the Graphic Output Protocol (GOP) driver. With UEFI, the OS can finally communicate with boot firmware in a standard way; this work is strongly supported by standards work in UEFI and the TCG (Trusted Computing Group). This enables such features as secure boot, where the OS and firmware cooperate in creating a secure handoff mechanism. It also enables a seamless visual experience from the time you hit the power button – one experience owned by two distinct components.

The boot experience has not been thoroughly revamped, well, ever. The BIOS menus have been stuck in time for nearly 30 years while OSes and hardware have advanced at a logarithmic pace. We've introduced many features of the pre-OS environment over several releases of Windows, each designed with a different set of capabilities and limitations. For instance, due to the lack of full graphics capabilities, the Multi-OS and Advanced Boot Options menus displayed by the boot manager shown below appear as if they were from the MS-DOS era:



Due to the lack of theme support in the Windows RE, experiences built using standard controls offer a look and feel from the Windows 9x/2000 era:



Boot is a highly visible portion of the system—users see it on average 1-2 times per day. That's already too much, but this post is not about making reboots go away. We recognize that this number will change as slates and devices that are always on become more prevalent, but for those times when you may still need to boot, we want it to be fast and fluid. The experience of booting a PC should be approachable to mainstream consumers while maintaining the power of Windows for more advanced users who want to configure settings in the pre-OS environment. As you can imagine, satisfying all of these goals was challenging and in many situations, a balancing act.

Seamless setup and first boot on Windows 8

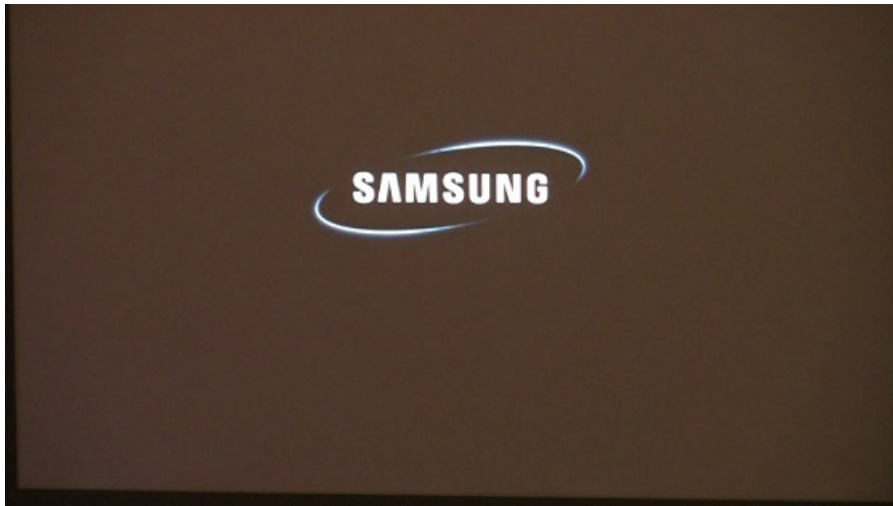
What better place to start thinking about the boot experience than the Windows 8 setup UI. This is one of the first places where we've made sure to give a great touch-first experience. The entire setup process, including entering your product key, joining a wireless network, and setting up a default account, will be accessible using the soft keyboard.

If you go out and buy a new PC that comes with Windows 8 pre-installed, you will likely encounter the "Specialize" phase of setup on your first boot. This is where machine-specific information and drivers are installed on your system. In the past, you would go through a series of screens with a visual appearance distinct from other phases of setup. Now, the visual experience will be seamless from POST, to boot, to setup.

Seamless boot, every time

The Windows 8 boot experience will reflect the personality of Windows; it will be fast and fluid, seamless, and beautiful every time. By leveraging the capabilities of UEFI and working together with the ecosystem, our goal is for the PC to power up to the manufacturer's logo and stay on that screen all the way from POST to Windows logon UI. The logo should be beautiful and reflect the brand you trust when you purchased your PC. Firmware renders the logo during POST, the logo persists on screen when Windows boot takes over, and remains through OS boot. In effect, we

are bridging two experiences (firmware + operating system) to deliver one experience, as you see here:



Advanced functionality

We know that some of you love to customize your machines by changing OS settings, booting from a physical device, or performing boot troubleshooting in Windows RE. You are not only getting a seamless experience every time you boot, you're also getting a beautiful, touch-first experience even if you are someone who wants to look under the hood.

We did a thorough inventory of all the advanced features available and designed an experience that gives a consolidated view of the functionality you may want before entering the OS. Unlike in previous versions of Windows, the advanced boot options in Windows 8 can be reached easily, are simple to navigate, and look and feel harmonious.

Your browser doesn't support HTML5 video.

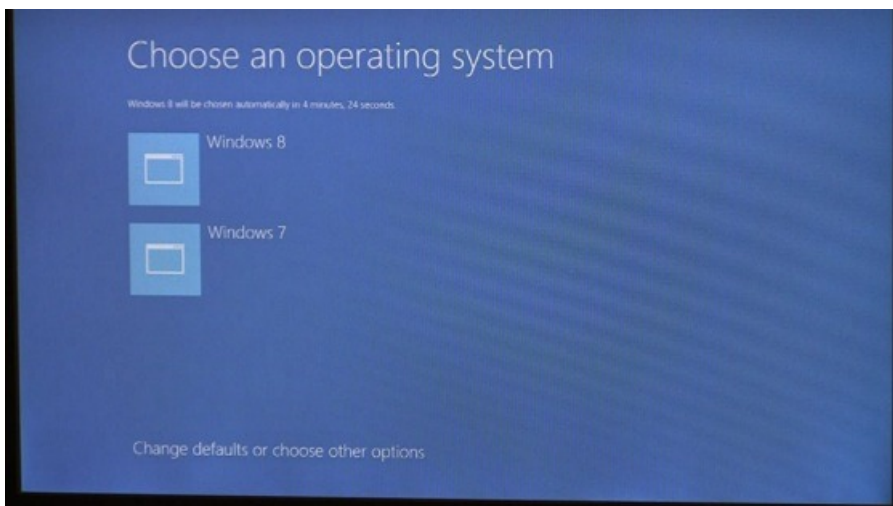
Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

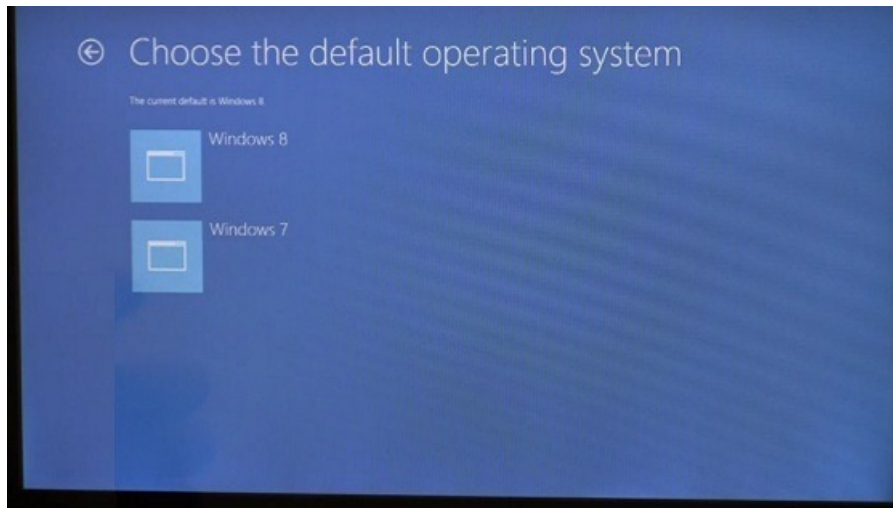
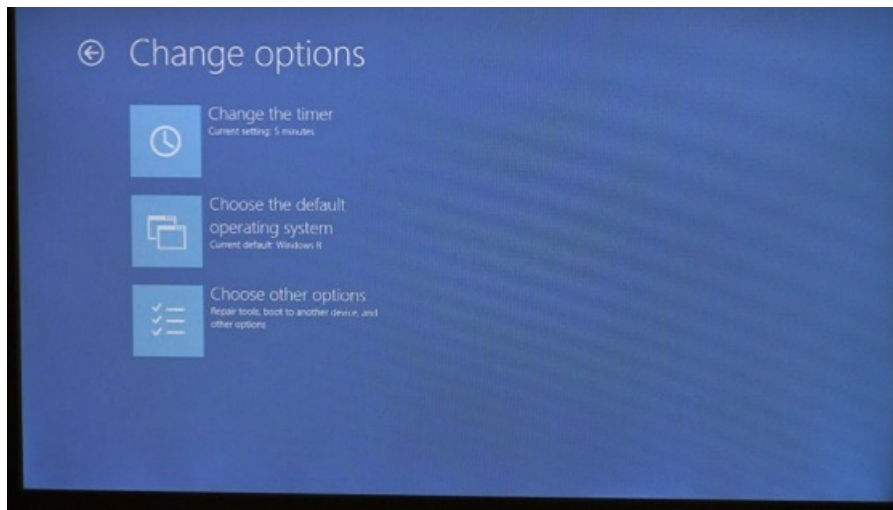
You can see a complete demonstration in the video above, but let's call attention to a few scenarios here.

Dual booting your PC

Let's say you're someone who has multiple copies or versions of Windows installed on a PC. In Windows 8, you will be presented with a high-fidelity, immersive, touchable UI where you can select which OS to boot with a single tap (or mouse click, or tab-key navigation).

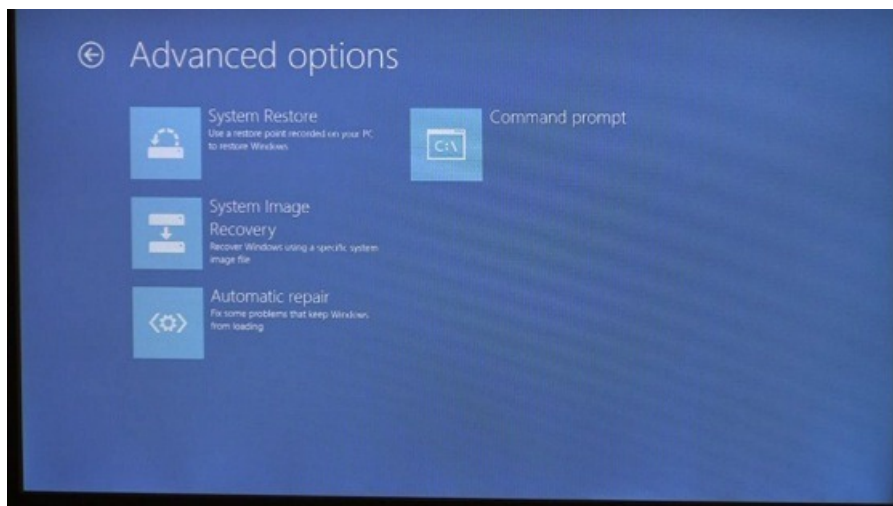


There are often reasons to change your default OS, or you may want to change the countdown window before the default OS starts. Changing these settings today is cumbersome, as you have to edit the Boot Configuration Data (BCD) store. We decided that this functionality was important enough to bubble up to the core user experience when booting with a dual OS setup. Instead of remembering bcdedit commands, or changing settings in msconfig (though these options are still available), you can easily configure the default OS and timer settings right within the boot UI.

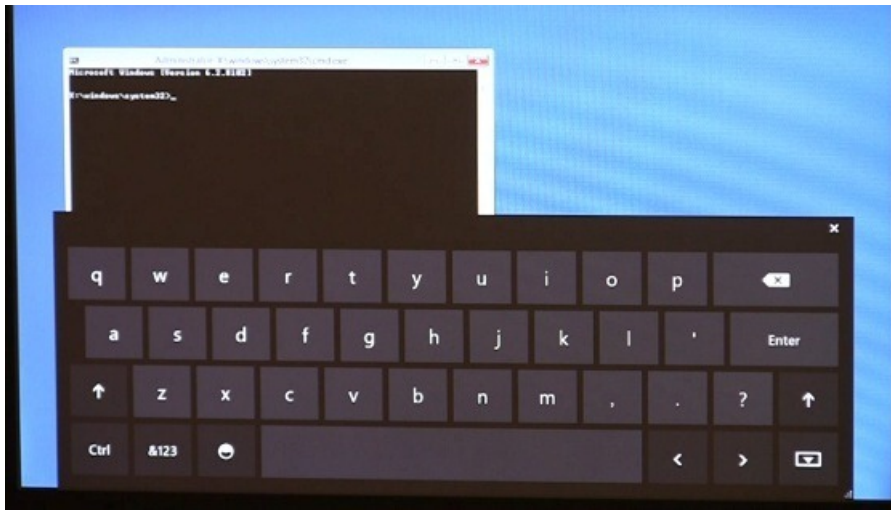


Performing troubleshooting in Windows RE

Let's now consider the scenario where you may need to boot into Windows RE to troubleshoot a startup problem or to restore Windows to a previous restore point. Even for such advanced functionality (that you may use rarely, if ever), we wanted to ensure that you would have a consistent and touchable experience.

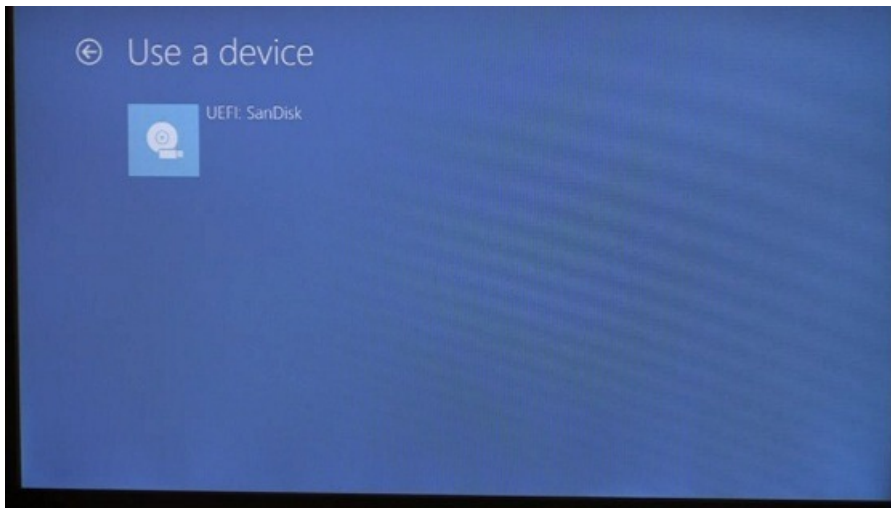


To illustrate just how deeply we thought about this experience, let's assume you need to launch a command prompt window from within Windows RE (to check the access control lists (ACLs) on some files, for instance). We've even made the soft keyboard available from the command prompt in Windows RE if you need that (imagine a field repair of a device with no keyboard!).



Booting to a device

We recognize that many of you boot to devices, for example, to a bootable USB stick. Today, this requires entering the BIOS boot options menu which could be under one of many Function keys, depending on the hardware/firmware vendor. But with UEFI firmware, the OS can call back into firmware to enumerate the BIOS boot options. This means that advanced boot options that were formerly only available from BIOS menus will be available alongside the Windows-provided functionality.



As mentioned above, we did a thorough inventory of *all* capabilities of the system and thought of how everything should feel like part of the overall Windows PC experience. Nothing escaped our inventory, not even the dreaded “[BSOD](#)”! Unfortunately, things may go wrong with hardware from time to time, so there was no way to completely rid the world of the “BSOD.” This was a very interesting balancing act as we worked through several design iterations to determine how much information to display. We wanted to meet the needs of power users (whether you’re troubleshooting your machine or a family member’s) and at the same time, make it less scary for the consumer. One thing you’ll notice is that in spite of all the changes, we did decide to keep it blue! ??



Your PC ran into a problem that it couldn't handle, and now it needs to restart.

You can search for the error online: SYSTEM_THREAD_EXCEPTION_NOT_HANDLED (pci.sys)

It'll restart in: 1 second

We hope you are as excited as we are to see the user experience of boot get the attention it deserves. We anticipate many questions about the work we've done and we look forward to a continued dialog with you.

Thanks,

Billie Sue

Protecting the pre-OS environment with UEFI

Steven Sinofsky | [2011-09-22T15:00:00+00:00](#)

There have been some comments about how Microsoft implemented secure boot and unfortunately these seemed to synthesize scenarios that are not the case so we are going to use this post as a chance to further describe how UEFI enables secure boot and the options available to PC manufacturers. The most important thing to understand is that we are introducing capabilities that provide a no-compromise approach to security to customers that seek this out while at the same time full and complete control over the PC continues to be available. Tony Mangefeste on our Ecosystem team authored this post. –Steven

Quick summary

- UEFI allows firmware to implement a security policy
- Secure boot is a UEFI protocol not a Windows 8 feature
- UEFI secure boot is part of Windows 8 secured boot architecture
- Windows 8 utilizes secure boot to ensure that the pre-OS environment is secure
- Secure boot doesn't "lock out" operating system loaders, but is a policy that allows firmware to validate authenticity of components
- OEMs have the ability to customize their firmware to meet the needs of their customers by customizing the level of certificate and policy management on their platform
- Microsoft does not mandate or control the settings on PC firmware that control or enable secured boot from any operating system other than Windows

The big picture – no compromises on security

The UEFI secure boot protocol is the foundation of an architecturally neutral approach to platform and firmware security. Based on the Public Key Infrastructure (PKI) process to validate firmware images before they are allowed to execute, secure boot helps reduce the risk of boot loader attacks. Microsoft relies on this protocol in Windows 8 to improve platform security for our customers.

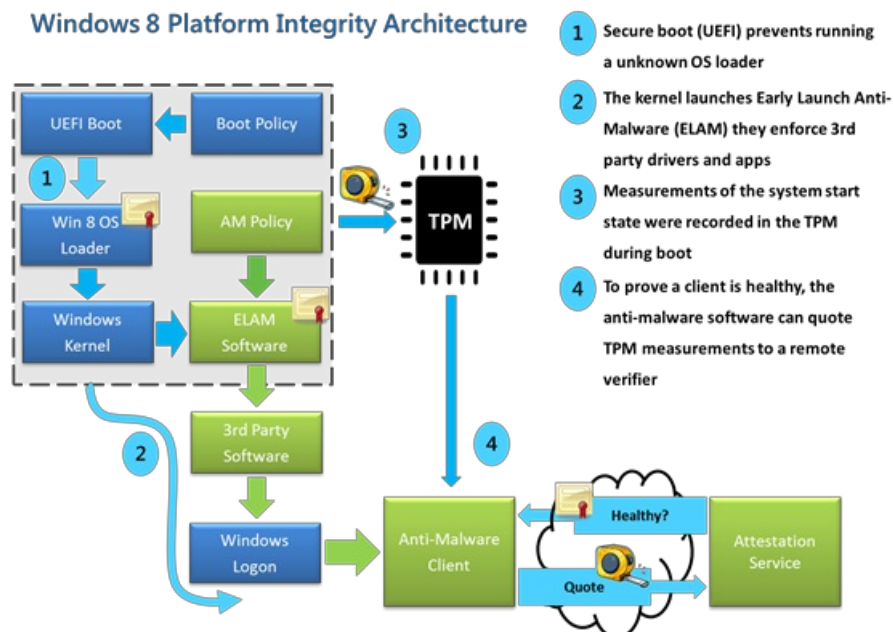


Figure 1 – Platform integrity architecture

Microsoft is working with our partners to ensure that secured boot delivers a great security experience for our customers. Microsoft supports OEMs having the flexibility to decide who manages security certificates and how to allow customers to import and manage those certificates, and manage secure boot. We believe it is important to support this flexibility to the OEMs and to allow our customers to decide how they want to manage their systems.

For Windows customers, Microsoft is using the Windows Certification program to ensure that systems shipping with Windows 8 have secure boot enabled by default, that firmware not allow **programmatic control** of secure boot (to prevent malware from disabling security policies in firmware), and that OEMs prevent unauthorized attempts at updating firmware that could compromise system integrity.

Most of these policies are not new to UEFI firmware, and most PCs today carry some form of firmware validation. Even the existing legacy support, such as BIOS password, is a form of secure boot that has been under OEM and end-user control for years. However, with secure boot & UEFI, the industry and Microsoft are raising the bar to create greater system integrity and health, and to provide customers with a strong level of protection against a growing class of threat.

What is UEFI?

UEFI (Unified Extensible Firmware Interface) is managed through the UEFI forum, a collection of chipset, hardware, system, firmware, and operating system vendors. The forum maintains specifications, test tools, and reference implementations that are used across many UEFI PCs. Microsoft is a board member of this forum, and the forum is open to any individual or company to join free of cost.

UEFI defines the next generation firmware interface for your personal computer. The Basic Input and Output System (BIOS) firmware, originally written in assembly and using software interrupts for I/O, has defined the PC ecosystem since its inception – but changes in the computing landscape have paved the way for a “modern firmware” definition to usher in the next generation of tablets and devices.

The intent of UEFI is to define a standard way for the operating system to communicate with the platform firmware during the boot process. Before UEFI, the primary mechanism to communicate with hardware during the boot process was software interrupts. Modern PCs are capable of performing faster, more efficient block I/O between hardware and software, and UEFI allows designs to utilize the full potential of their hardware.

UEFI allows for modular firmware design that enables hardware and system designers a greater flexibility in designing firmware for the more demanding modern computing environments. Whereas I/O was limited by software interrupts, UEFI promotes the concept of event-based, architecture-neutral coding standards.

What is secure boot?

UEFI has a firmware validation process, called secure boot, which is defined in Chapter 27 of the UEFI 2.3.1 specification. Secure boot defines how platform firmware manages security certificates, validation of firmware, and a definition of the interface (protocol) between firmware and the operating system.

Microsoft’s platform integrity architecture creates a root of trust with platform firmware using UEFI secure boot and certificates stored in firmware. A growing trend in the evolution of malware exploits is targeting the boot path as a preferred attack vector. This class of attack has been difficult to guard against, since antimalware products can be disabled by malicious software that prevents them from loading entirely. With Windows 8’s secured boot architecture and its establishment of a root of trust, the customer is protected from malicious code executing in the boot path by ensuring that only signed, certified “known good” code and boot loaders can execute before the operating system itself loads.

In most PCs today, the pre-operating system environment is vulnerable to attacks by redirecting the boot loader handoff to possible malicious loaders. These loaders would remain undetected to operating system security measures and antimalware software.



- The BIOS starts any OS loader, even malware

Figure 2 - Legacy BIOS boot path

Windows 8 addresses this vulnerability with UEFI secure boot, and using policy present in firmware along with certificates to ensure that only properly signed and authenticated components are allowed to execute.



- UEFI will only launch a verified OS loader – such as in Windows 8
- Malware cannot switch the boot loader

Figure 3 - Secure boot path with UEFI

Secure boot is only a part of the Windows 8 Platform Integrity story. Along with UEFI, Microsoft’s strategy is a holistic approach to other available hardware to further enhance the security of the platform.

Background: how does it work?

Powering on a PC starts the process of executing code that configures the processor, memory, and hardware peripherals in preparation for the operating system to execute. This process is consistent across all platforms, regardless of underlying silicon architectures (x86, ARM, etc.).

Shortly after the system is powered on, and before handoff to the OS loader occurs, the firmware will check the signature of firmware code that exists on hardware peripherals such as network cards, storage devices, or video cards. This device code, called Option ROMs, will continue the

process of configuration by ensuring that the peripheral is prepared for handoff to the operating system.

During this part of the boot process firmware will check for an embedded signature inside of the firmware module, much like an application, and if that signature matches against a database of signatures in firmware, then that module is allowed to execute. These signatures are stored in databases in firmware. These databases are the “Allowed” and “Disallowed” lists that determine if the booting process can continue.

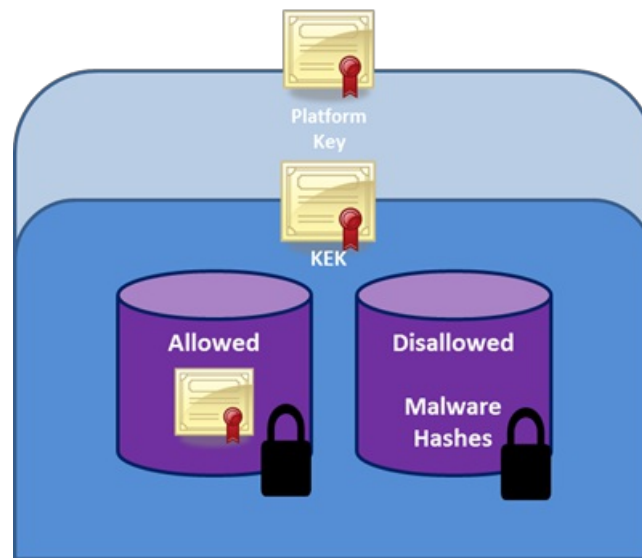


Figure 4 - Security databases for certificates

This figure represents the hierarchy of signatures and keys in a system with secure boot. The platform is secured through a platform key that the OEM installs in firmware during manufacturing. This is the process used today on most shipping systems, regardless of whether they are based on UEFI, or legacy BIOS. (Applications like firmware update utilities will use the platform key to protect the firmware image.) Other keys are used by secure boot to protect access to databases that store keys to allow or disallow execution of firmware.

The Allowed database contains keys that represent trusted firmware components and, more importantly, operating system loaders. Another database contains hashes of malware and firmware, and blocks execution of those malware components. The strength of these policies is based on signing firmware using Authenticode and Public Key Infrastructure (PKI). PKI is a well-established process for creating, managing, and revoking certificates that establish trust during information exchange. PKI is at the core of the security model for secure boot.

What is required for secure boot?

Secure boot requires firmware that meets or exceeds UEFI revision 2.3.1. The UEFI forum ratified the latest revision which updated the policies of Chapter 27 to improve upon the existing secure boot protocol to include time-authenticated variables, stronger keys for encryption, and clarification on how those certificates are stored.

The feature would be transparent to the consumer purchasing a PC. The benefit is that their system has an added measure of reliability from bootkit and rootkit attacks that target system vulnerabilities before the operating system itself even loads, as described above.

Who is in control?

At the end of the day, the customer is in control of their PC. Microsoft’s philosophy is to provide customers with the best experience first, and allow them to make decisions themselves. We work with our OEM ecosystem to provide customers with this flexibility. The security that UEFI has to offer with secure boot means that most customers will have their systems protected against boot loader attacks. For the enthusiast who wants to run older operating systems, the option is there to allow you to make that decision.

A demonstration of this control is found in the Samsung tablet with Windows 8 Developer Preview that was offered to //BUILD/ participants. In the screenshot below you will notice that we designed the firmware to allow the customer to disable secure boot. However, doing so comes at your own risk. OEMs are free to choose how to enable this support and can further customize the parameters as described above in an effort to deliver unique value propositions to their customers. Windows merely did work to provide great OS support for a scenario we believe many will find valuable across consumers and enterprise customers.

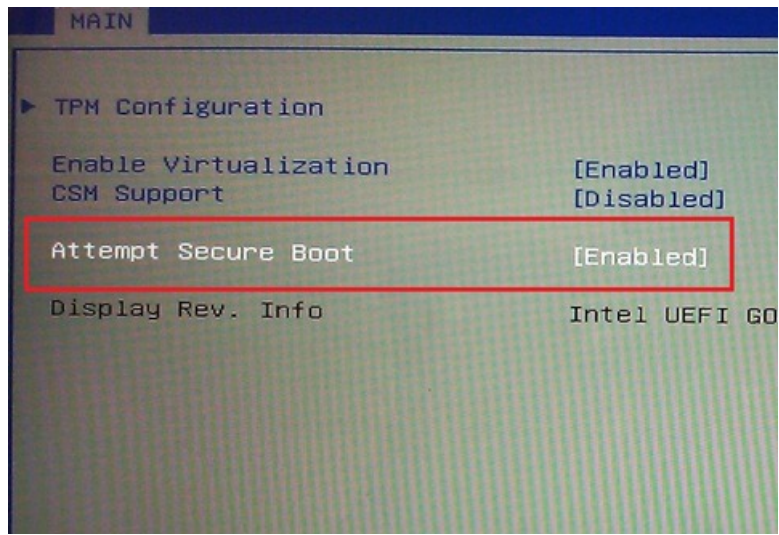


Figure 5 - Samsung PC secure boot setting

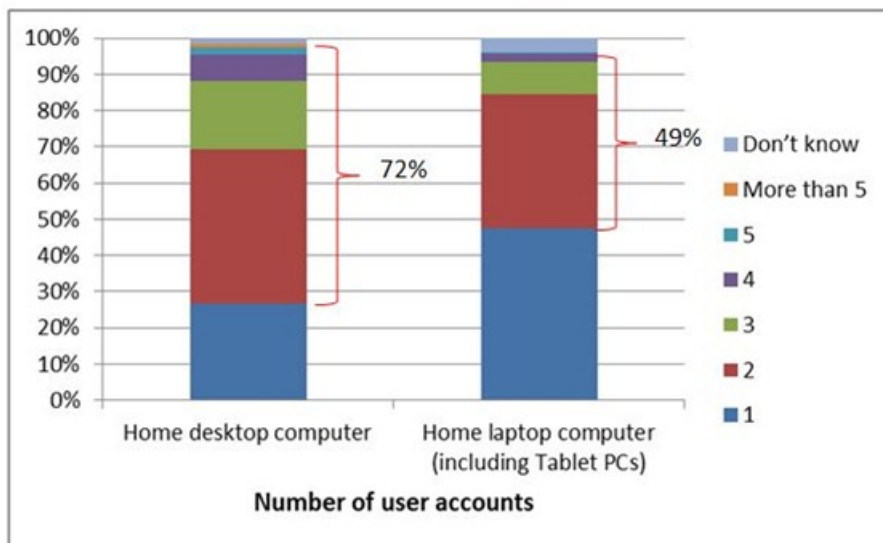
Tony Mangefeste
Ecosystem

Signing in to Windows 8 with a Windows Live ID

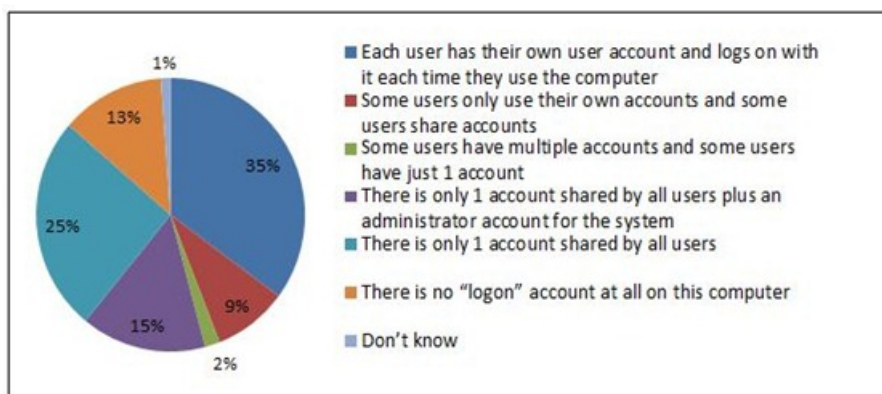
Steven Sinofsky | [2011-09-26T12:00:00+00:00](#)

With Windows 8, we introduce the optional capability to sign in to your PC with a Windows Live ID and, by doing so, gaining the ability to roam a broad range of settings across all of your PCs. In this article by Katie Frigon, the group program manager of the You-Centered Experience team, she describes the feature and its benefits. --Steven

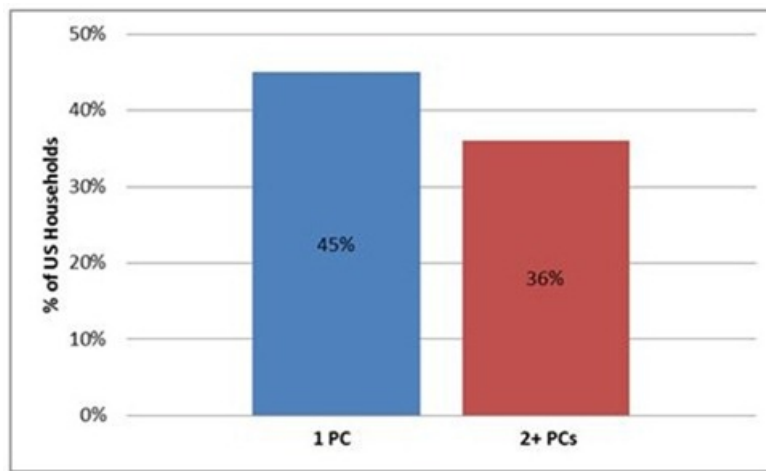
Each Windows user wants to have the ability to set up and use a PC in a way that is unique to them. Doing so, however, can be challenging in today's multiple user and multiple PC environment. We know that shared PC usage is common and we've heard from many of you that switching between multiple accounts can be cumbersome. The difficulties associated with managing multiple accounts often lead to the sharing of a single account on a PC, and a less personal (and potentially less private) experience for each user. We also know that users are utilizing multiple devices more often now, and setting up a new PC can be inconvenient and time consuming. In Windows 8, we have set out to ensure that each PC user has a truly personal experience that seamlessly bridges their online and offline tasks, is simpler to set up and use, and persists across their set of Windows 8 PCs. To do this, we've introduced the ability to log in to Windows (optionally) with a Windows Live ID that works across devices, apps, and services, allowing you a uniquely personal experience with Windows.



Shared PC usage occurs in 72% of desktops and 49% of laptops



How user accounts are used on shared computers



PCs per household in the United States

Benefits of signing in to Windows 8 with Windows Live ID

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

Signing in with an ID allows you to:

- Associate the most commonly used Windows settings with your user account. Saved settings are available when you sign in to your account on any Windows 8 PC. Your PC will be set up just the way you are used to!
- Easily reacquire your Metro style apps on multiple Windows 8 PCs. The app's settings and last-used state persist across all your Windows 8 PCs.
- Save sign-in credentials for the different apps and websites you use and easily get back into them without having to enter credentials every time.
- Automatically sign in to apps and services that use Windows Live ID for authentication.

When you buy a Windows 8 PC and set up your user account for the first time, you can optionally choose to create an account that is associated to a Windows Live ID. You can either use an existing ID or create a new one. If you choose to create a new one, you can use any email address you want as your new ID, and then create your unique password. For example, you can use `example@live.com` or you can use `someone@example.com`. You just need to identify an email address that you want to have associated with the Windows Live ID service, and provide a unique password. Of course, you can also continue to use local Windows accounts as you always have and obviously, domain-administered accounts work as they always have as well.

So, although many people assume they will need to sign up for a new email account to get a Windows Live ID, it's actually not necessary. In fact, many online services use a "string" like `someone@example.com` to represent a user name, even though that string looks like an email address. For example, when you order books at an online bookstore, your user name may look like an email address, even though your online book seller does not manage your email. The `someone@example.com` address is just a convenient way of identifying you, since most Internet users these days have email addresses. So, your email account and password will still be managed by whatever email provider you choose, and we use the user name and password you give us to help manage your settings and state across your Windows 8 PCs, even if you haven't signed up for Hotmail or other Microsoft services that use this ID.

Like all of us, you probably spend a significant amount of time personalizing your Windows experience to reflect your style, your life, and how you use your PC. We all know how frustrating it is when all that work is lost when you buy a new PC or use a different one (or just reformat your hard drive). With Windows 8, we are working to change that—you will be able to have your personal Windows experience on any Windows 8 PC you sign in to with your Windows Live ID. Settings such as your lock screen picture, desktop background, user tile, browser favorites and history, spell check dictionaries, Explorer settings, mouse settings, and accessibility settings, among many others are now associated with your Windows 8 account and stored in the cloud. They are kept in sync and come down to each machine you use as they are changed or updated.

Having a truly personal experience in Windows 8 also includes your Metro style apps—how you use them, the settings you use, and where you left off. It will be easy to see which Metro style apps you've purchased and choose which ones you want to have on each of your Windows 8 PCs. By using your ID to sign in to Windows, the settings and state for your Metro style apps stay in sync between each PC you use. For example, let's say you are reading the news in a reader app on your tablet. If you add specific feeds you want to continue to follow, those feeds could automatically be available in the same reader app on any of your other Windows 8 PCs. We will also enable developers to build Metro style apps that tell Windows their state, so you can pick up where you left off as you move between PCs. You can pick up on the same page of a book, the same level of a game, or the same place in the movie you were watching as you switch between your Windows 8 PCs. In the developer preview of Windows 8, you can see this functionality in Internet Explorer 10.

You might be wondering how you can roam non-Metro style apps and settings without a domain. This isn't something that can be done with Windows Live ID, and we would discourage using tools that manually attempt to do this by mechanisms such as going through the registry or

copying around executables. However, using the new Restore/Refresh tools, it is possible to easily create an image that has your preferred desktop apps installed, and then use that as a refresh point. If you do want to roam your settings for desktop apps then you can continue to use the mechanisms available for roaming profiles and client side caching of files available with Active Directory and Windows Server.

Another benefit of signing in with a Windows Live ID is how we've simplified the need to sign in to multiple services and applications. We accomplish this in two ways. First, once you've signed in to Windows with your ID, you do not need to enter it again to sign in to any app or website that also uses Windows Live ID. For example, once you sign in to Windows with your ID, you can launch the Windows Messaging app and start talking with your friends without the need to sign in again. Similarly, you can browse to your Hotmail inbox page without needing to enter your email address and password again. You can always sign out of a webpage and sign in as a different user, but by default you will be automatically signed in. To be clear, however, those applications and websites do not have special access to your Windows PC or your personal data.

Second, if you choose to, Windows can store separate Metro style app and web site credentials. Those credentials can then sync to each Windows 8 PC that you've trusted and verified yourself with. You won't have to type in your user name or password; just confirm your sign-in as needed. Similar to the Messaging application example, when launching a Metro style application that uses this feature, you will be signed in automatically and the application will resume right where you left off.

User controls

There is a lot of benefit to using a Windows Live ID to sign in to Windows. However, it is important to note that every Windows user is unique in their needs. Your Windows 8 experience is in your control. When you create a Windows account, you choose the type of account you want to use. You can choose to create one that associates with Windows Live ID, or stick with a local account that works just like in Windows 7. You can also change a local account to link it with a Windows Live ID at a later date.

If you choose to associate your local account with an ID, we've provided control over what you want to sync to each Windows 8 PC you use. In Control Panel, there is a section called "Sync PC Settings" where you can manually turn settings sync on or off.

You can choose to turn off all syncing or you can turn off syncing per the type of setting. The settings groups include:

- Personalize
- Themes
- Ease of access
- Language preferences
- Apps
- Web browser
- Other stuff
- Some passwords

We've recommended a default behavior that assumes you want to roam settings that are used most often to personalize and customize the way you use your PC. In particular, we've heard from you that visual personalization for your PC is important. For Windows 8 we've included key settings like changing your lock screen image. In addition, you can also roam the desktop themes you use and create, including colors, sounds, and desktop background (note: currently for the background image we roam the original image that was selected if it's under 2MB. If the image is over 2MB we compress and crop the image to 1920x1200).

It is also important that you maintain control of your data when work and personal start to mix. In Windows 8, when you link your Windows domain account to a Windows Live ID, we ask you up front (before data is synced) what data you want to sync between your domain-joined PC and other PCs you use with that ID. That way, you can decide if things like your web history, favorites, or credentials should sync to your work machine, or if you'd prefer to keep those or anything else that is synced only on your personal machines.

We also empower IT administrators to control what a user can sync to a work PC through group policy. We have provided control to IT administrators to decide if a worker can link their domain account to an ID, and if the admin allows that link, what types of data the worker is allowed to sync.

Finally it's important to note that credentials that are entered and stored on a domain-joined machine do not get uploaded to the cloud, and never get synced to your other PCs – this ensures that corporate credentials stay on the PCs that are managed by the IT admin.

Privacy and security

We understand that when using services connected to the cloud, privacy and security are on the top of your mind. When you associate your Windows user account with a Windows Live ID, there are three categories of data that are especially interesting from the privacy and security perspective:

1. Your Windows Live ID user name and password
2. Your Windows Live ID user profile
3. The settings and data you choose to sync

We've taken measures to safeguard the ID and password you use to sign in to Windows. We do this in a couple ways. First, we will require a

strong password (and you can't leave password blank). Next, we'll collect a secondary proof of your identity. This will allow us to establish "trust" with specific PCs that you use frequently or own. This in turn will also enable more secure syncing of private data like passwords. Collecting the secondary proof of your identity also helps make account recovery easier and more secure. Examples of secondary proofs are alternative email addresses, mobile phone numbers, and questions with secret answers—something that generally only you will know.

Signing in with a Windows Live ID also gives you much more control over your password, including your ability to recover a lost one. If you use a local account and you forget your password, you're in a tough spot, and your options are limited. You may be able to recover your password with a hint or a recovery key, but if neither of those works, you're generally left with having to rebuild your PC from scratch. (Technically there are some password cracking tools available on the Internet that you could download and try, but they're unlikely to work on a suitably strong password, and many of the cracking tools available online are actually malware downloads!) However, if you sign in to your PC with your Windows Live ID and you later forget your password, you can reset your password from another PC by navigating to <https://login.live.com> and clicking on "forgot my password." This will allow you to reset your password in a secure fashion without losing any information on your PC. Resetting your password this way is also more secure because it takes advantage of the secondary proof we mentioned earlier to make sure it's really you resetting your password.

You might also be wondering, "what happens if somehow my Windows Live ID gets stolen?" Well, we have some help for you there too. Windows Live ID includes a number of different safety features to detect if your account is stolen, and it will change your account to a "compromised" state (limiting what it can do) until you can regain control of your account using the two-factor authentication features (secondary proofs) that you set up earlier. Importantly, you will still have full access to your PC, since your PC will allow you to log in with the password you had before your account was stolen – you just won't be able to use the services and applications that rely on this ID until you go through our "recover my account" workflow online.

With Windows 8, we want to put you in control of how your data is used and what you want to sync between Windows 8 PCs. When you choose to sign in to your Windows 8 PC with a Windows Live ID, only a small amount – your first name, last name, and display name -- are shared with Windows. Windows does not use any of your other profile data. Your profile data stored in the cloud is released to apps or websites that you allow to have that data. While any Metro style app can leverage Windows Live ID for their own sign-in authentication, they must always ask you first if you want to allow access to particular details from your profile.

As mentioned earlier, there are three categories of data that can be synced to your Windows 8 PCs when you sign in with your ID: 1) Windows settings, 2) App settings and data, and 3) credentials. This data is stored in the cloud so that it is available to you when you sign in to your various Windows 8 PCs. The size of the data we roam is minimal and we only enforce some limits on a per setting basis, for example, the file size for the lock screen image. None of this counts against your Windows Live storage quota. This data is also stored separately from your other Windows Live data, for example, what you store on [SkyDrive](#).

You might be concerned with how profile data is protected. In order to secure user data, we've taken several measures. First, we do not roam data over WWAN by default. Second, all user data is encrypted on the client before it is sent to the cloud. All data and settings that leave your PC are transmitted using SSL/TLS. The most sensitive information, like your credential information, is encrypted once based on your password and then encrypted again as it is sent across the Internet. The data stored is not available to other Microsoft services or third parties. Lastly, before the sensitive information can be accessed on a second Windows 8 PC for the first time, you must establish "trust" for that PC by providing further proof of your identity. This further proof can be done by providing Windows with a code sent to your mobile phone number or by following the instructions sent to an alternate email address.

Any of the data that is saved to the cloud via the roaming mechanism is only accessed by Windows for roaming. This is very important. So for example, Internet Explorer's history is saved as a roaming state but is not used or accessed in any other context—it is no different than if you had manually created that same record of website history on another PC.

We are very excited about the opportunity to make the Windows 8 experience more personal and easier to set up in a way that protects your privacy and safety. We look forward to hearing about how you are enjoying the feature and to receiving your feedback!

Katie Frigon

Extending "Windows 8" apps to the cloud with SkyDrive

Steven Sinofsky | [2011-09-28T12:00:00+00:00](#)

Building on the recent post about signing in to Windows 8 with a Windows Live ID, we wanted to talk a bit about using SkyDrive from within new Windows 8 Metro style apps. While apps get some amount of "automatic usage" of SkyDrive to roam settings, and the apps themselves roam, we know developers are anxious to make it easy for customers of their applications to have data they create roam easily across devices. SkyDrive is a great way to do this as every Windows Live ID has a free cloud-based drive. This post talks about how developers can build Metro style apps that use SkyDrive storage from within the apps. This is our first post with code—expect more of those! Mike Torres, the group program manager of the SkyDrive Devices and Roaming team, authored this post. —Steven

Despite the trend towards multiple devices per person, many people still store all their important files on a single PC or storage device, and don't have access to them from their other devices. These files are associated with a "place"—a desktop PC, a laptop, or a USB key. If you don't have access to that place, you don't have access to your documents or photos. Not only is accessing your files difficult (or sometimes impossible), sharing them with someone on a different network can be just as tricky.

Now, the cloud is making it possible (and easy!) for people to access their content from almost anywhere. Files are kept in a single place and are accessible from any device that connects to the internet from anywhere in the world. Sharing photos and collaborating in real-time on documents has also been made easier by having a single copy of the file in the cloud. However, the cloud has not yet reached mainstream usage for accessing content.

At the //build/ conference opening keynote, Chris Jones talked about how every Windows 8 customer will get a SkyDrive: a single cloud for everyone, where a person's important files are centrally available, instantly accessible, and ready to share. There were also sessions during the conference covering how Windows 8 developers can cloud-power their apps through Live Connect and the Live SDK. When used together, Metro style apps can use the Live cloud to enable [single sign-on with Windows Live ID](#) and access personal data like documents, photos, and videos on SkyDrive with the user's permission.

If you're thinking about developing a Metro style app for Windows 8, this post shows how to enable single sign-on and access a user's data on SkyDrive to make your Metro style app more personal—with the user's consent, of course.

Connecting your apps to the Live cloud

As a developer, when you set out to build the world's next great app, there are two classes of problems you'll repeatedly face on any platform:

- **Enabling sign-in and sign-up for users.** Users dislike having to sign in to websites and apps, yet developers know that engagement and loyalty to their app increases after a user has signed in. Sign-in allows you to personalize the user's experience and to have your app remember the user's data and customizations.
- **Easily bringing the user's content into your app to power your experience.** Seemingly simple tasks like asking the user to set a profile picture can quickly get complicated when you consider that users have their photos spread out across disparate devices and the cloud.

In Windows 8, we've addressed both of these problems for our own apps like Photos and Mail, and your Metro style apps are able to use the same platform and technology. Specifically, this is how we've addressed the above problems:

1. The user's cloud-based identity is now an OS primitive that is universally accessible to apps & websites with user permission for single sign-on. This means your app can inherit the signed in state of a user and their identity, and you don't have to worry about a separate authentication process. This becomes especially important to enable #2.
2. We've made access to the user's content in the Live cloud available to apps using industry standard protocols such as OAuth for authentication & authorization, JSON as payloads for the data returned when accessing SkyDrive and Hotmail, and XMPP for interoperability with Windows Live Messenger. In addition, we offer the Live SDK for Windows 8 Developer Preview to make development on the WinRT easy, with integration into Visual Studio 11 Express.

After the first time a user connects your app to their Windows Live ID, the user will always have a seamless single sign-on experience from any Windows 8 PC where they are signed in with a Windows Live ID. The same extends to your website, where they get a single sign-on experience if they are signed in to their PC with a Windows Live ID or signed in to any site that supports Windows Live ID, such as Hotmail or SkyDrive.

The easiest way to use single sign-on with Windows Live ID and integrate SkyDrive content into your Metro style app is to leverage the Live SDK.

This does not mean that your application needs to use any of these services—their use is entirely up to the developer. There's no requirement that apps for Windows 8 sign on with a Live ID or use any specific cloud-based storage. These are simply services available to app developers to use as they choose.

Using the Live SDK in your app

First, if you haven't done so already, you'll need to install the [Windows 8 Developer Preview](#) and the [Live SDK Developer Preview](#) on your PC. This also installs Microsoft Visual Studio 11 Express for Windows Developer Preview. Second, you'll need to visit the [Windows Push Notifications](#)

[and Live Connect app management site](#) to configure your Metro style app to access the Live cloud. Follow the steps on the site to register your app to use Live Connect.

After your app is configured to use Live Connect, you'll need to add a reference to the Live SDK. The Live SDK is available for development in C#, JavaScript, and VB. You can add it to your project by right-clicking the Project and selecting **Add Reference**, selecting **Extension SDKs** and from the list, and then selecting the entry for the Live SDK as shown below:

In a JavaScript application, you just need to add the following script reference to Default.html:

```
<script src="ms-wwa:///LiveSDKHTML.5.0/js/wl.js" ></script>
```

In C#, you should add the following reference to be able to leverage the Live SDK in your code:

```
using Microsoft.Live;
```

And in VB, the following import statement is all it takes:

```
Imports Microsoft.Live
```

This is just a good example of how we support programming in the language of your choice when building Metro style apps for Windows 8.

Using Windows Live ID in your Metro style app

To take advantage of Single Sign On (SSO) in your app, you'll need to place a sign-in button somewhere in the app. When the user clicks the sign-in button, they will be automatically signed in if they are signed into their PC with a Windows Live ID, otherwise, they'll be asked to sign in. After that the user is asked to provide consent to your app to access their data such as their SkyDrive photos. This workflow is handled automatically for you by simply adding the sign-in button. Notice that your customers still confirm sign-on to your application and are not automatically signed on—this is an important design consideration.

This is sample HTML for the sign-in button (note that it's just a DIV and will need to be configured).

```
<div id="signinbutton" style="width: 251px; margin-left: auto; margin-top: 40%; height: 64px; top: 0px;"></div>
```

Once you've added the sign-in button to the page, you'll now have to hook it up since all the sample HTML did was allocate space for it on the page. And we need to configure that sign-in button with the scopes that your app needs. A *scope* defines what the app will have access to, and what the user has to agree to provide. To access SkyDrive data, your app will need these two scopes:

- **wl.signin** - This scope enables automatic sign-in to your app.
- **wl.skydrive** - This scope grants read access to the user's SkyDrive albums and photos (note: your app should use **wl.skydrive_update** if you plan to upload content to SkyDrive).

This is sample JavaScript code for initialization, assuming that you've already created the sign-in button on the page with the id "signinbutton":

```
function init() {
    WL.init();

    WL.ui({
        name: "signin",
        element: "signinbutton",
        scope: ["wl.signin", "wl.skydrive"],
    });
}
```

And this is sample JavaScript code for handling login:

```
WL.Event.subscribe("auth.login", onLoginComplete);

function onLoginComplete() {
    var session = WL.getSession();
    if (!session.error) {
        signInUser();
    }
}
```

When the user clicks the sign-in button, they are asked to grant consent to allow your app to access their data. As noted earlier, if the user is signed into their PC with a Windows Live ID, then single sign-on is in effect and the user doesn't have to sign in again. Instead they go directly to this consent screen:

Accessing SkyDrive content from your app

Once the user has granted access to their data to your app, OAuth 2.0 access tokens are returned to your app, and can then be used to make RESTful API calls against the Live cloud. Here's what the code to access a photo looks like in JavaScript:

```
function downloadPicture(folderId) {  
    var path = folderId + "/files"  
    // Submit request  
    WL.api({ path: path, method: "GET" }, onEnumerateFolderComplete)  
};
```

With a number of the above calls, an app can embed SkyDrive content into their experience such as PowerPoint slides, videos, Excel spreadsheets, or just plain photos, as shown below in a sample app.

As you can see, integrating single sign-on and user data from SkyDrive into your app requires just a few lines of code with the Live SDK. Your app will be more personal and can take full advantage of a user's photos or documents in the cloud. The full source code for the sample app shown in this blog post can be [downloaded from here](#).

For more information about using the Live SDK to enable single sign-on in your apps and take advantage of the SkyDrive APIs, watch Dare Obasanjo's session, [Power your app with Live services](#) and Steve Gordon's session, [The complete developer's guide to the SkyDrive API](#) from the BUILD conference. You can also learn more by checking out <http://dev.live.com> and downloading the [Live SDK Developer Preview](#). Happy coding!

Mike Torres

Evolving the Start menu

Steven Sinofsky | [2011-10-03T12:15:00+00:00](#)

This post kicks off a series of posts on the design of the Start screen and the evolution of the core activity of launching and switching programs. Some folks are calling the Start screen the "Metro shell" for Windows 8, but for us it is the evolution of the Start menu and associated functions. We've been watching the comments closely and have seen the full spectrum of reactions as one would expect when the core interface changes. We want to use these blog posts to have a dialog that reflects back on your comments, and so we'll start by walking you through the history and decisions that led to the current design. Because the Developer Preview is focused on building apps, and the core user experience is still under development, we want to make sure our discussions start from first principles and work through the design to provide a fuller context for where we will be at the next project milestone.

This post was authored by Chaitanya Sareen, program manager lead on our Core Experience Evolved team. You might remember Chaitanya, as he also worked on the Windows 7 experience and authored posts on the Engineering 7 blog.

–Steven

We'd like to share a series of blog posts on the how and why of reimagining Start. This first post talks about the history and evolution of the Start menu, and several of the problems and trends we've learned from you. We think it's always important to understand where we've come from before we talk about where we're headed. We'll then have another post that dives into how we crafted the new Start screen, and then we'll see where the discussion leads us from there.

We recognize that to some people, any change to Windows is going to be disruptive, and so we want to make sure we continue an open dialog about those changes. Since Windows is such an integral part of so many people's lives, most any change can generate visceral reactions like "how can I turn it off," or debates over whether things are more or less efficient.

The debate around touch today is looking eerily like the debate in the 1980s over whether a mouse was a gimmick, a productivity time waster, or an innovation in the user experience. We say this knowing that many comments have been emphatic about the superiority of the mouse over touch. Unlike when the mouse was introduced—before desktop publishing programs came along there were few use cases for the mouse other than early paint programs—today we are surrounded by touch screens—at the airport, the gas station, the movie theater, every cash register, and of course, on our phones. The one place touch has not yet become mainstream is on the most capable of all the devices you use. Just like the introduction of the mouse, innovations like this do not happen without new OS support, new apps, and new hardware. We believe that, as with the mouse, we will see touch augmenting, but not replacing, most every aspect of the PC experience over time. Achieving this starts with the Windows 8 Developer Preview. So with that, let's start the dialog about how things will evolve, not just in the Windows core user experience, but in hardware and apps as well.

With regard to the main user experience, particularly Start, we're noticing some themes in your comments. Will there be a way to close Metro style apps without going to Task Manager? (Yes there will be, but we also want to talk about why you probably won't need to use it.) Are you going to do anything to make the mouse more efficient in scrolling through your programs in Start? (Yes, we'll improve that experience, and show you much more in the beta.) Some of you have talked about it feeling less efficient to cycle through your recent programs compared to using the taskbar (and we'll have more to say about that in future posts). There are other comments as well, and the point here is just to make sure you know we are aware of the questions. Some things will be easier to discuss if we first agree on some shared terminology. For example, Metro style is a design language we can apply to any element of Windows, and the Start screen is the evolution of the Start menu (as well as taskbar, notifications, and gadgets), and not really a "Metro shell." Another example is that we don't see "Metro" as a mode of Windows, rather it is a way to describe the attributes of applications written to WinRT (as in these //build/ talks on [traits of Metro style apps](#) or [principles of Metro style](#)). There's lots of ground to cover. We know with our initial focus on the platform and tools, we probably did not provide enough early context around the user experience changes through this blog.

The Start menu is one of the most visible parts of Windows, and so we don't take any changes we make to it lightly. The environment around Windows has changed immensely since we first introduced the Start menu, and we want to make sure we're still delivering an experience that is both relevant and tuned to the dynamic computing world we live in today. The evolution of the Start menu is inextricably linked with the development of several other related, but disparate concepts, such as application launching, application switching, system notifications, and gadgets. The history behind these and the divergent paths they have each taken create an opportunity for us to do a much better job in providing a cleaner, more powerful, and more uniform way of working across the wide variety of apps and PCs we use today. The new Start screen embodies this effort.

So, before we tackle Windows 8, let's first take a trip down memory lane and see what we can learn about the Start menu.

A brief history of Start

The design of the Start menu began in 1992 for its debut in Windows 95. The menu was conceived in a world where PC towers and 15" CRTs dominated cubicles. The Web was still an experiment and people had to drive to a store to buy software. It was a very different time. The fundamental goal of the menu was to provide an obvious place for people to start their computing tasks. It replaced the venerable Program Manager, that Windows 3.x concept that placed shortcuts in a floating window which happened to interfere with the desktop and other

applications. Anchored to the taskbar, the Start menu was a consistent and consolidated portal to your apps and system functions. It was essentially the fastest way to start programs without hunting down an executable somewhere in the system.

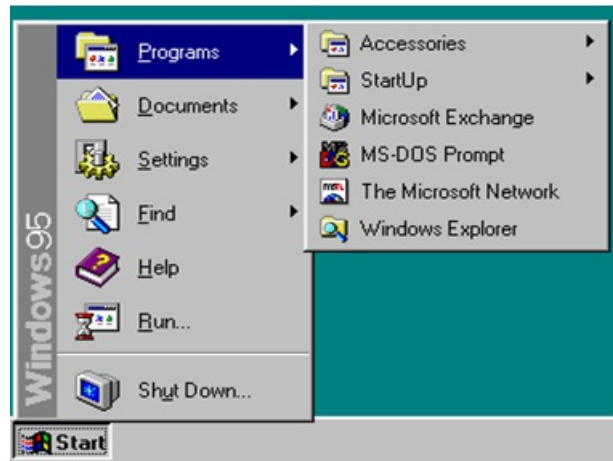


Fig 1: The Windows 95 Start menu

Looking at the Windows 95 Start menu, you may immediately see the areas for improvement that subsequent versions of Windows addressed with incremental changes. For example, we heard feedback that putting an alphabetical list of programs under a flyout made it slow to navigate. Windows XP addressed this with the introduction of the “most frequently used” (MFU) section that surfaced the programs you used regularly. This change in turn introduced some new problems because there was no way to really customize it, and some people struggled to understand how the MFU was populated (a complex heuristic determined which apps you use the most). To address the customization aspect, Windows XP (and later, Windows Vista and Windows 7) featured a section where you could pin the apps that are most important to you, to put them at your fingertips. However, this functionality was still limited. You could put apps in the pinned section and reorder them, but you still couldn’t group or organize them if you had more than just a few items.

There were also problems with traversing All Programs on Window XP. It wasn’t uncommon to have your mouse “fall off” the menu and then you’d have to restart the task all over again (this was especially challenging for laptop customers who used track pads or for those with limited dexterity). It was also difficult to fit all this UI into lower resolution screens. Vista addressed this with the introduction of a single menu and a tree control that required fewer acrobatics with the mouse. However, All Programs still felt cramped, since the menu required a scroll bar (fig 2). The Start menu was already starting to feel full.

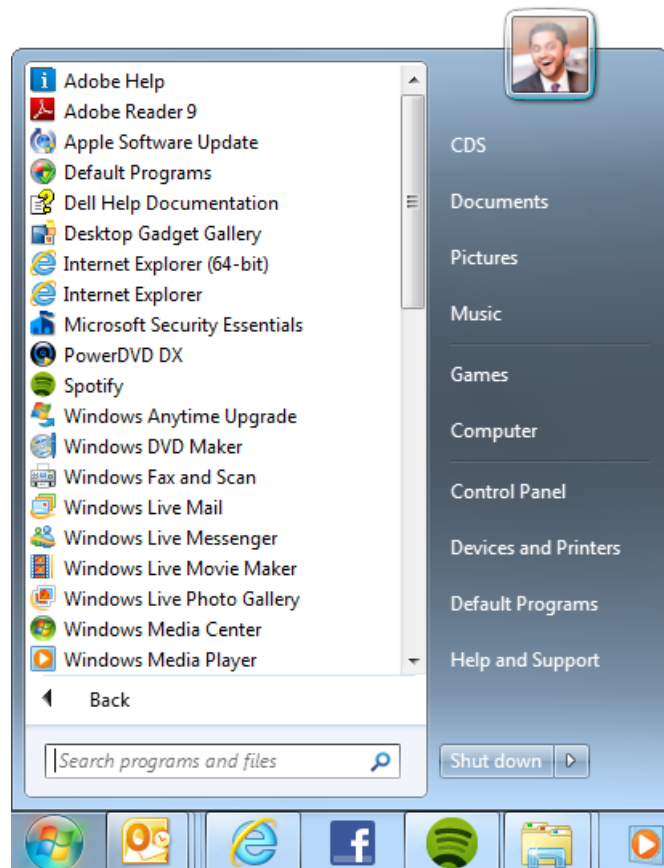


Fig 2: All Programs in the Start menu in Windows 7

Another critical evolution of Start was the introduction of instant search. Vista and Windows 7 both made it really easy for you to open the menu and just type what you wanted. We know many of you enjoy this powerful method, as it reduces “time to launch.” The introduction of advanced

commands also reduced the need to use the Run dialog. Nonetheless, we knew there were still areas for improvement. Search results sometimes feel overcrowded and showing lots of different types of data (email, files, control panel settings, etc.) didn't always work best in a one-size-fits-all column. Some of you have asked whether the Windows 8 Start screen will also support search. Yes, it does—on the Start screen, just start typing to instantly see search results that you can filter by apps, files, or settings. And just like in Windows 7, the full power of search is available in Explorer.

Based on these challenges and your feedback, we've continued to refine the Start experience over the years. However, we find that even the Windows 7 Start menu still faces core usability challenges:

- The menu feels cramped relative to available screen real estate when you try to see and navigate the full catalog of your programs.
- Search doesn't have the space it deserves to quickly show you rich results across all sources of information, especially on larger screens.
- It's hard to customize the menu to make it feel like it's really yours.
- Icons and shortcuts are static and don't leverage more of the pixels we see in modern graphical interfaces to surface connected scenarios.

Of course, the above list isn't a complete set of everything we are improving. We also aim to unify Start with the rest of the system and enable new scenarios. An important part of design is sometimes taking a step back to fundamentally reimagine something from the ground up in order to bring more than incremental improvements to a product. This is especially true for something like Start that was born in a very different time when we didn't use our PCs the way we do today.

It is important not to lose sight of the breadth of the problem space. The Start menu is almost exclusively used to launch items (except for the subset of search functionality). The full program experience in Windows 7 also includes switching and pinning via the taskbar, alerts in the notification area, and gadgets on the desktop. As we continue our dialog, we'll talk about how Windows 8 brings these all together in a harmonious manner.

How is the Start menu used?

Now that we've briefly discussed the history of the Start menu, let's discover how people are actually using it. We thought it would be interesting to see how the usage of the menu has changed over time. Figure 3 reveals the change in Start menu usage across the two versions of Windows.

Start Menu Feature	Change in usage
Pictures	-61%
Documents	-56%
Control Panel	-54%
Pinned items	-51%
All Programs	-42%
Computer	-40%
MFU	-28%
Menu open	-11%

Fig 3: Change in Start menu feature usage between Windows Vista and Windows 7

It is striking to see how dramatically different the use of the Start menu is in Windows Vista vs. Windows 7. Some of the Special Folders (what we call those items on the right side of the menu) dropped in use by over 50%. Likewise, people accessed pinned items on the Start menu half as often in Windows 7 than they did in Vista. People also access All Programs and the MFU far less often. Finally, we see an 11% drop in how often people are opening the Start menu at all. While 11% may seem like a small number at first, across our hundreds of millions of customers it is eye opening to see such a drop for a universally recognizable element of the Windows interface. We're not talking about some hidden setting that is tweaked by a minority of people—we're talking about a fundamental piece of Windows that people are using less and less.

So why the change in how people are using the Start menu? Here's a hint—it has something to do with that bar at the bottom of your screen that was introduced in Windows 7.

The "Start bar"

The evolution of the Windows taskbar directly impacted the Start menu. What once was locked behind a menu suddenly came closer to you. The most obvious advancements were the introduction of Quick Launch by Internet Explorer 4.0's Windows Desktop Update in 1997, as well as the more recent taskbar pinning in Windows 7.

Interesting side story: did you know that Quick Launch was initially disabled by default in Windows XP because some people believed the MFU list and pinning in the Start menu would suffice? We saw a volume of evidence to the contrary, and so we reversed the decision (though back then, the data upon which we based these decisions was limited, so we don't really know what a broad variety of customers were doing). What we took away from this was that it was important for you to be able to designate what apps you care about, see them all in one place, and have them be one click away, rather than trying to guess what is important through software heuristics or having important items mixed with less important items.

To really bring this all home, let's take a look at where people are pinning their apps. Figure 4 reveals that 85% of people have three or more items pinned to the taskbar compared to a mere 23% who have the same number pinned to the Start menu. Although the taskbar and Start menu have different pinned defaults, many people do customize both of them when they want to. The message is clear that the majority of people want most

of their apps on the taskbar rather than having to dig into Start.

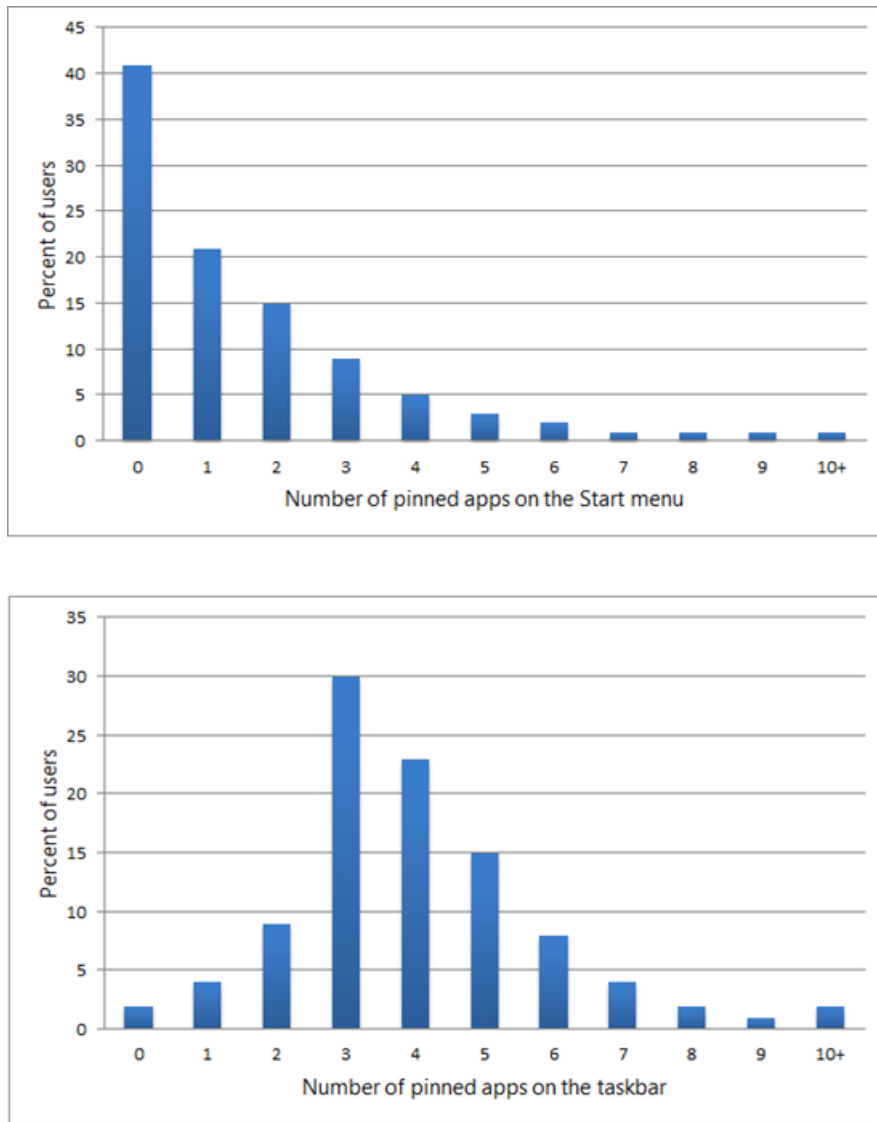


Fig 4: Number of pinned apps on the Start menu (top) vs. on the taskbar (bottom)

We also know that enthusiasts in particular use their Windows 7 taskbar even more than the Start menu. Keyboard shortcuts like Win + <n> (where *n* corresponds to the sequence of an application icon on the taskbar) make it even faster for the keyboard experts to instantly launch and switch with the taskbar (and those shortcuts continue to work in Windows 8). When we visit IT pros, it's not uncommon for us to see a taskbar filled with icons for standard corporate desktops. We even see items like Control Panel pinned to the taskbar to save people a trip to Start. Pinning is also increasing in popularity because you can now also pin websites to your taskbar with IE 9. Fortunately, there's plenty of room on the taskbar—even at 1024x768 the taskbar can hold 22 small icons. Add the power of Jump lists, and theoretically, you can also have access to 220 files, folders, and sites at that same resolution! This means that for those who wish to just use desktop apps, the taskbar provides the room to quickly access the things you need every day without going to the Start menu.

Speaking of Jump lists, we've seen also how pinning Explorer by default to the taskbar and populating its Jump list with common folders makes it even easier to access system folders like Documents (not surprisingly, use of Documents in Start has also dropped, as shown above).

In summary, the taskbar has evolved to replace many aspects of the Start menu. You can even say the taskbar reveals many of the weaknesses of the Start menu and that the menu is no longer as valuable as it once was long ago. Search and access to All Programs are still unique strengths of the Start menu that we know you depend upon, but when it comes to the apps you use every day, one-click access from the taskbar is hard to beat. You, and many like you, are the ones who gave us this strong feedback over the years, which pushed us to make the taskbar a powerful primary launcher and switcher for the desktop. In fact, we sometimes even referred to the taskbar in Windows 7 as the "Start bar," since it became clear that most people now start with the bar, rather than with the menu.

A new opportunity for Start

With the Windows taskbar becoming the key launcher and switcher for the desktop, and the Start menu being revealed as a poor everyday launcher, an opportunity appeared to reimagine Start and make it into something more valuable. Since we now know most of you can (and do) just use the taskbar to access the things you commonly use on the desktop, this freed us up to make Start even better at its unique strengths and to unlock new scenarios. Improved search, more room for all your programs, tiles that are alive with activity, and richer customization all suddenly become possible when the venerable, but aging, Start menu is transformed into a modern Start screen. Stay tuned for our next post, where we'll talk about the Start screen and how it represents the way we use our PCs today.

Designing the Start screen

Steven Sinofsky | [2011-10-04T11:03:00+00:00](#)

Thank you for the comments and feedback on the previous post. We definitely get the message that there's a lot of feedback and passion around the design. We're going to continue talking about the design and answering your questions and comments through these blog posts. We designed Start to be a modern, fast and fluid replacement for the combination of launching, switching, notifying, and at-a-glance viewing of information. That's a tall order. And of course, we set out to do this for the vast majority of customers, who are more familiar with the Start menu, mouse and keyboard, as well as for new customers using touch-capable devices. This post is authored by Alice Steinglass, the group program manager for the Core Experience Evolved team. –Steven

As we wrote about in our post on [evolving the Start menu](#), after studying real world usage of the Start menu through a variety of techniques, we realized that it was serving mainly as the launcher for programs you rarely use. As more and more launching takes place from the task bar, the Start menu looks like a lot of user interface for programs you don't use very frequently. And the Start menu is not well-optimized for this purpose. It affords limited customization, provides virtually no useful information, and offers only a small space for search results. We found that people “in the know” who valued efficiency were moving away from the Start menu, and pinning their frequently used programs to the taskbar so that they could access them instantly in one click. We see this quite a bit on professional workstations where there are sets of tools that all fit on the taskbar and are all used regularly—machines used by engineers, designers, developers, information workers, etc.

So, as evidence mounts that the menu hasn't kept up with the modern way in which we use our PCs today, we've seen a growing interest in replacements for the Start menu (whether for touch, or mouse and keyboard). At the same time, we've seen an ever-increasing use of cumbersome notification tray icons (with ever-increasing menus and actions), and a continued interest in desktop gadgets that have yet to realize their potential.

In light of these realizations, we stepped back and reimagined the role of Start in Windows 8. We knew that we already had a powerful launcher for desktop programs in the taskbar. The Start screen is not just a replacement for the Start menu—it is designed to be a great launcher and switcher of apps, a place that is alive with notifications, customizable, powerful, and efficient. It brings together a set of solutions that today are disparate and poorly integrated. As we have said, some of these features, as well as the full scope of mouse and keyboard support, are not included in the Windows 8 Developer Preview, which was focused on building Metro style apps and the WinRT APIs.

Alive with activity

As we analyzed Windows systems, we found that the average PC is cluttered with a large array of system tray notifications, a long list of folders and shortcuts for installed software, and applets in numerous places in the system, all begging for your attention. In addition to the programs, people access a large number of websites with updated data from the Internet or a company intranet. These programs and websites consume and present a constant stream of fresh data: new email, business data, communications, articles, pictures, feeds, etc.

We designed the Windows 8 Start screen so you can create a connected dashboard that keeps you in touch with all the apps, activities, places, and people you care about. The news app shows the latest headlines, the weather app shows the forecast, an RSS app tells you what's new, a social networking app displays your status, or a game can tell you when it is your turn—and when it isn't. While these are just examples, it is not hard to imagine the apps you use today (whether in the browser, on the desktop, gadgets, or notifications) being reimagined as Metro style apps that connect to the same exact data sources, but instead provide a rich, customizable, interactive "heads-up display." We expect corporate applications to be developed that display Live files for important internal systems and processes too. You can envision even the most mundane uses being improved by this ability to track live data. For example, our development team has been using a Metro style bug tracker (an example of a corporate application) that connects up to our existing bug database. It was a small amount of work to create, and replaced several varied notification tray icons and gadgets in use around the team.

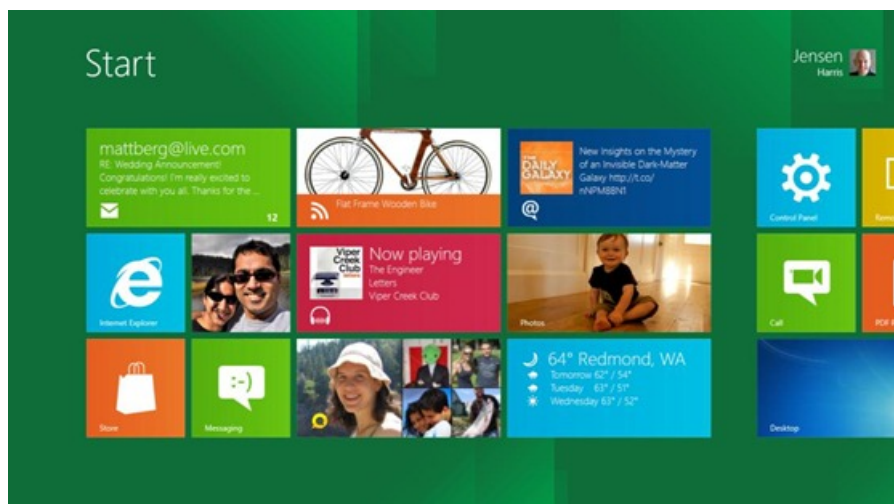


Fig 1: Live tiles on the Start screen

Apps can still represent themselves with just an icon and a name. And, for certain tools, that will continue to make sense: Command Prompt, Task Manager, etc. But, for most of today's more relevant and connected apps, a simple icon and name is limiting, when they have so much more information they can share. And that information can be shared at a glance without any window management or any context change on your part.

We knew we needed to be able to load live tiles instantly and efficiently. Long battery life and instant performance is fundamental to the experience of any mobile device. If every app launched and loaded a process when you entered Start (the traditional "gadget" model), it would slow down the performance of navigating to it, scrolling, etc. It would also increase the background memory and CPU footprint, which would decrease battery life. Today, when people want to check to see if they have new email, they run their mail app, see an app-specific notification competing with other notifications, or open up another browser tab. If they want to see what's going on with their social networks, they leave open a bunch of different web sites or apps. If they are waiting for their turn in a game, that app is open. This means they are forced into a situation where they are either impacting their performance and battery life by running each of these programs simultaneously, or they are disconnected and have to constantly open and close programs. Why should your mobile device be so much better at these routine tasks than your PC (of any form factor) which has a vastly larger screen, more storage, and more connectivity and processing power?

To address this, the Start screen uses a single process to pull down notifications from the [Windows Notification Service](#) and keep the tiles up to date. The tiles are cached, so they can load instantly when you go to Start. The result is that the tiles aren't apps—they are a system-provided surface that can quickly tell you what's new with your app. They are an extension of the apps you use (or the apps you develop), providing instant access to relevant content without costing battery life or slowing down performance.

Customization

A critical part of creating a meaningful dashboard and launcher is enabling you to customize it to be yours. The old Start menu offered some limited customizability—you could pin a few apps to a short list, and use the customization dialog to choose which of the limited built-in quick links you wanted to show. But, the choices were severely limited. You could not add your own links. You could not link to anything other than an app. You couldn't change the order of the apps, group apps, or pin more than a few apps. In fact, we've heard a lot of complaints about the challenges of ordering the Start menu items manually and maintaining that ordering. The taskbar helps with some of these issues, but it has a limited surface area to work with relative to the screen.

As we started to design the new Start screen, we considered options where we would automatically sort apps or pin a set of quick links (similar to the right side of the Start menu today). But, in our user research, we found that people didn't want us to guess what they would use. People who are proficient using their PCs want the flexibility to design their own Start experience. We know in the current Developer Preview, the automatic layout is a particular concern and of course as we said, we're not done yet and this is something we will make sure is under your control. The organizational tools we'll describe below (naming, grouping, zooming) were shown in [Jensen Harris's talk at //build/](#), but were not in the Developer Preview.

Good customization options start with organization. The Windows 7 Start menu is just a simple flat list. But, as people collect more and more apps, the ability to organize and group apps together becomes more important. We brought a variety of people of different skill-levels into our test labs and asked them to organize the apps and websites they use frequently. The variability was surprisingly large. People did not fit all their apps into the same predefined groups, or even the same group sizes. Some people had 5 games they thought belonged together. Others had 40. Sometimes a group of tiles had a clear name, like Games or News. Other times, people couldn't come up with a good name for the groupings they created, and chose to refer to them only as "things I go to frequently," for instance. Because of this, we designed the Start screen to give you flexibility over the number of groups, the size of any group, the layout of tiles within the group, and whether or not you want to name a group.



Fig 2: Tiles may be grouped with or without group names

But creating a custom dashboard is about more than just organizing apps. Organized or proficient users might want custom shortcuts to information or a location within an app. For example, instead of just having the headlines for an entire news site on your Start screen, you might also want a tile

that shows the headlines for the Sports or Tech section. With Windows 8, apps can provide these deep links too, so that people can create their own powerful and customized Start experiences. This means that tiles for apps can live alongside tiles that represent links to web pages, albums, playlists, specific people, a level within a game, a particular stock, etc. Any of these secondary tiles can be small or large, and can be put anywhere on the Start screen. They are “live,” just like app tiles, meaning that they are constantly updated with fresh and relevant content. This is a great way for app developers to provide differentiated functionality.

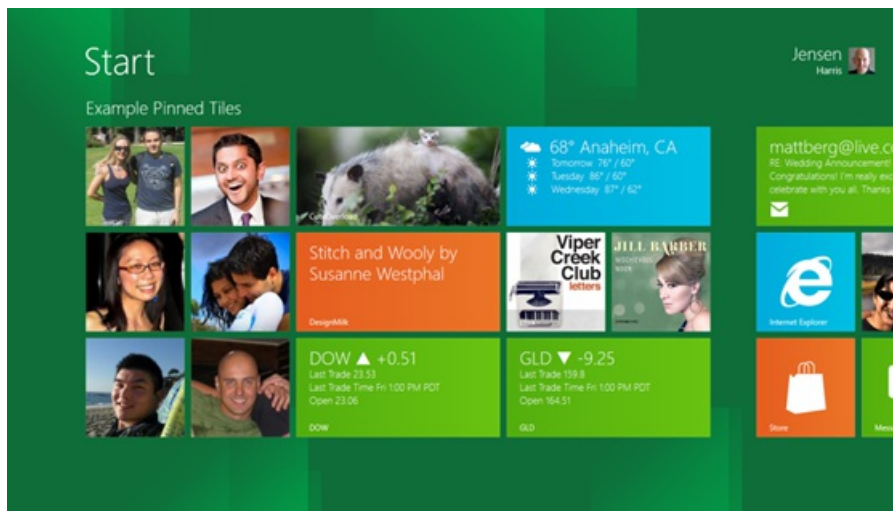


Fig 3: It's easy to customize Start by pinning deep links to apps wherever you want them

Start launches with instant access to everything on the first screen, bounded only by the size of the display. And, while Start supports standard scrolling, people told us they also wanted a way to quickly jump to a particular group. We enabled zoom as a way to step back, survey the landscape of the Start menu, and go directly to any group. We considered starting zoomed out and letting you dive into a group, but early usage data indicated that the vast majority of the time, people activate a tile that is on the first page. The standard zoomed-in view allows you to instantly glance at your dashboard just by hitting the Windows key on the keyboard, and then pressing it again to return to what you were doing. This means that checking anything on the Start screen is always just a single click or key press away. When you compare zoom to meticulously navigating a populated Start menu hierarchy, or uncovering pinned items on the taskbar, the experience is much faster and more fluid, and scales to many more programs and pinned items.

We of course considered folders, but our experience with folders broadly and in the Start menu tells us that folders are a way of burying things, not organizing them. Folders also make it impossible to see the up-to-date information an app might present. Once the apps are organized into groups, zooming out provides an at-a-glance view of the groups (similar to looking at a folder list). From the zoomed out view, you can jump directly into any group just as you would open a folder. For those wishing to stash certain programs out of sight, you can always remove the pinned icon from Start and use search to access it, or just put the program at the far end of the Start page. This is by far the most efficient way to manage a large library of apps.

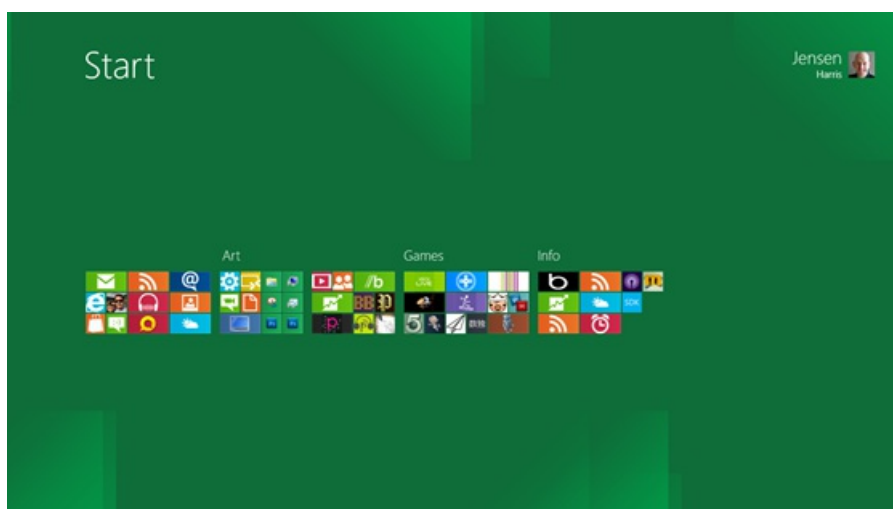


Fig 4: Zooming out in Start makes it easy to see groups of apps and target a specific section of the screen

Powerful and efficient

As we covered in our last post, the number of apps and websites that people use has been increasing dramatically over the last 10 years. When people had to go to a physical store and buy every program that existed on their PCs, it made sense to have a Start menu that was optimized for showing around 10 apps. Today, people use so many more apps and websites (which is another way to deliver app-like functionality), that the experience needed to be rethought for how people were using PCs today.

As we looked at layout options for the Start screen, we considered whether it should be full screen or appear on top of your apps as a small temporary window in the corner. Small popup windows are great for scenarios where you need to see the context of what is on the screen while you're using it (although modern user interfaces are using popups less and less). For example, it's a good design for doing advanced font settings in your Word doc. Its small size allows you to look at the text you're changing on the screen while setting the new font style.

But, when you're launching a new app, you're leaving the thing you're currently doing. So we wanted to take advantage of the whole screen to make launching and switching apps as efficient as possible. The full-screen Start gives you the power and flexibility to launch more apps with a single click. You can still put your most frequently used desktop apps on the taskbar in desktop. But the new Start screen has space to duplicate the 10-12 app links that you had pinned to the old Start menu, and still fit an additional 12 to 14 items on the first screen of a 1366 x 768 display. With a higher density display, obviously there's room to add even more apps that you can get to with a single click. As a reminder, Windows 8 requires a 1024 x 768 minimum resolution for Metro style apps, and as long as your screen is at least 1366px wide, you can use snap to show two apps at once. We're aware of the feedback about lack of diagnostic information other than the published system requirements on the download page—rest assured we are working on making this clearer.

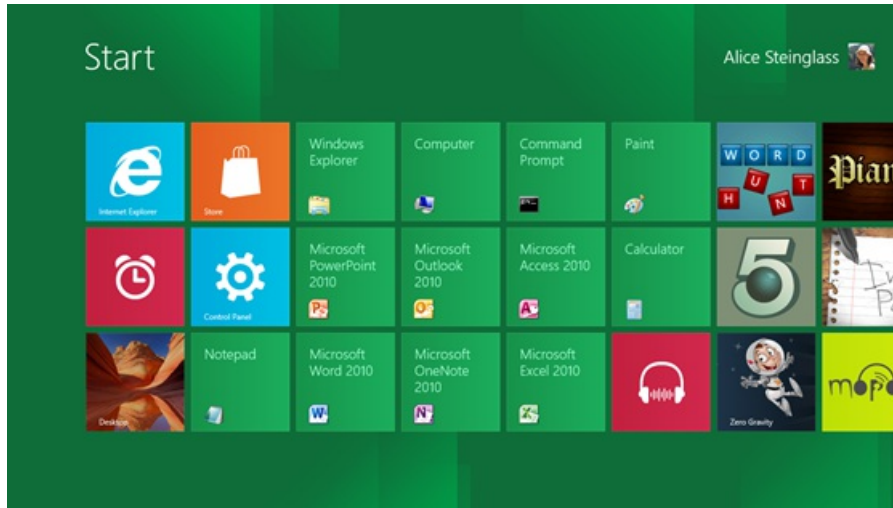


Fig 5: At 1366x768, Start shows 24 custom shortcuts on the first screen

People also want the efficiency of being able to instantly launch apps, documents, and settings without moving their fingers off the keyboard. We needed to retain this ability from Windows 7 while working to make it even better. If you have only one app with the word Excel in the name, launching it works exactly the same as it always has. Hit Start. Start typing "Ex..." and watch it autocomplete. Hit Enter, and Excel launches. Four keystrokes. Given the significantly increasing number of apps that people are using, search is clearly a more efficient way to access them, whether using a physical or on-screen keyboard. By combining the new Start screen with Search you get an ever-narrowing scope and easy hit targets whether for keyboard, mouse, or touch.

By comparison, as soon as the user searches for something with many hit results, the Start menu in Windows 7 can't scale to the results. For example, if you are looking for a Control Panel option with the word "input", the Start menu only returns the first 3 results in each category. To see the full list of results, you need to arrow down to the category (such as Control Panel), wait for Explorer to open, and then find the result you want there. If you are opening a document, you then need to manually close the Explorer window when you are done. While the new search capabilities in the Vista and Windows 7 Start menu clearly improved your ability to get to your top programs quickly, searching for anything other than a frequently used item remained inefficient and frustrating.

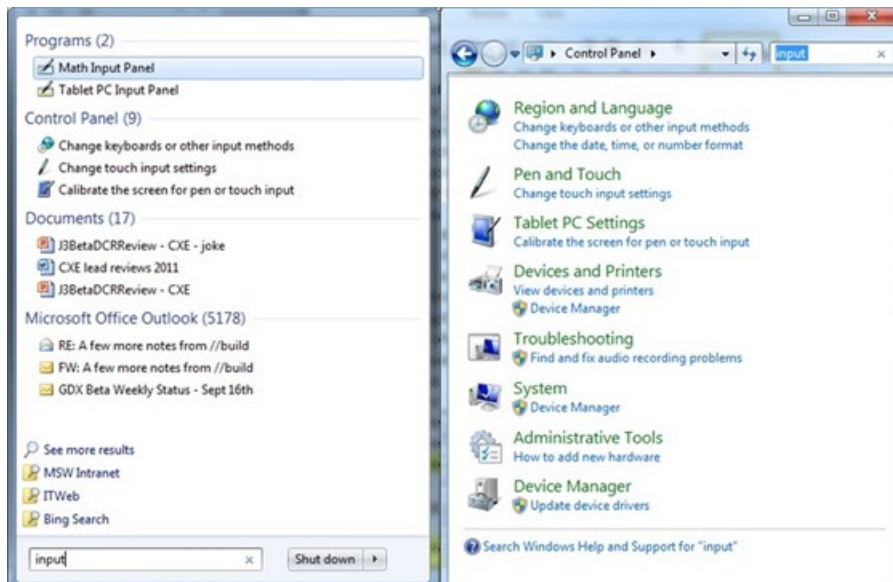


Fig 6: Limited room in the Start menu prevents a full lists of results from being shown and requires opening Explorer

With Windows 8, we wanted to minimize the keystrokes to instantly search and launch your app/file/setting/email/etc. Because the search results are full-screen, we can show at least 48 items on most screens, instead of just three. And, if you want to see the rest, you simply scroll (instead of launching Explorer and redoing the search). This simple change allows you to efficiently launch any app or file on the PC with a minimum of keystrokes.

We looked at two ways of improving the efficiency of search:

- Putting more on the screen
- Making it easier and faster to recognize the correct search result

The Windows 7 model forced all the search results into a standard template of an icon and text. As we looked at different data types, we saw that people could more quickly recognize their search result if the view was tailored to the data. Pictures should be displayed as thumbnails, email messages should say who they are from, videos should include their length, etc. So we designed a search model for the Start screen where each app displays the data in an optimized format. Thus, instead of seeing just 3 results per category type (all as text), you now can hit Start, type a search term, see an entire page of app results, or continue down the list to look at the results for files, settings, email, web, social networks, or any other app on your system. The [Search contract](#) is one of the exciting platform APIs in WinRT that developers can take advantage of. It allows this unified search experience while also letting the unique elements of an app's data shine through. It is a win-win for both the person using the app and the app developer.

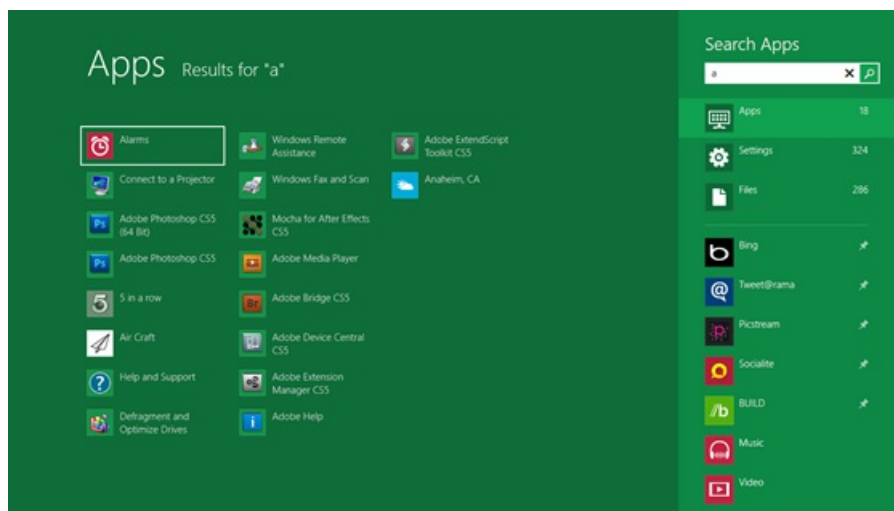


Fig 7: The Start screen has plenty of room to show app search results

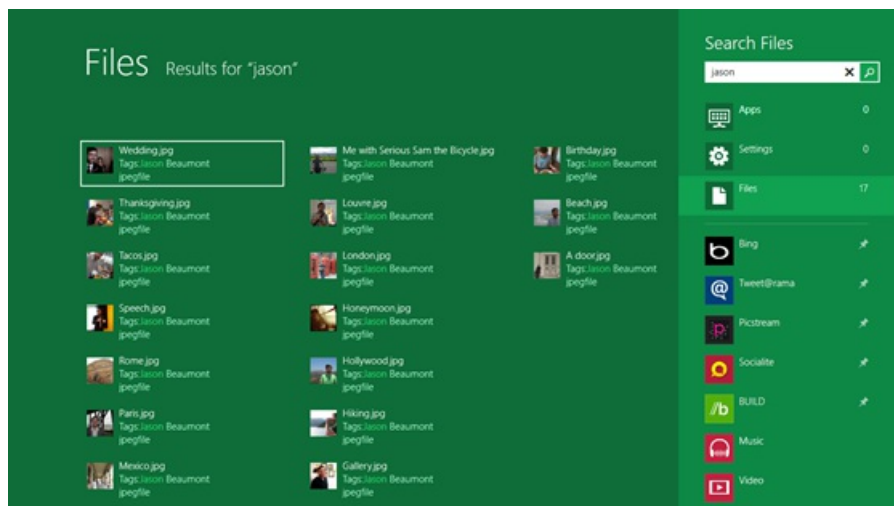


Fig 8: Start screen has more room to show you more detailed search results for files

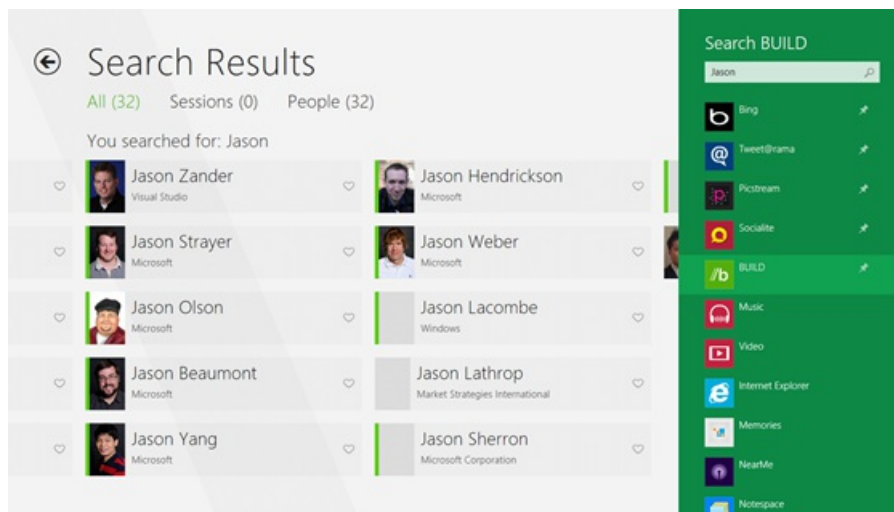


Fig 9: Search isn't just restricted to the system—apps can display search results optimized for that app

A full-screen Start experience empowers you to pick the things you care about (even if you care about way more than 12) and organize them the way you want so you can launch them instantly without scrolling. And, when you want to search, you can instantly see all search results (not just a couple). The design trades off peripheral awareness of what you're leaving behind in favor of optimizing for where you're going next, so that you can get there faster and more efficiently.

Keyboard and mouse

Because we have often demonstrated touch interactions with Start and its lineage in the Windows Phone Metro style, many believe that our design is all about touch rather than keyboard and mouse, or even that we're putting the phone interface on a PC—it is neither.

For mouse people, the position of the Start button in the lower-left corner of Windows 8 makes it an easy click-target (even in a full-screen app). Once in Start, more items are directly accessible to the mouse without scrolling or opening menu flyouts. For keyboard people, pinning frequently used desktop apps on the desktop taskbar enables instant shortcuts: Win+1, Win+2, etc. And, getting to less frequently used apps through search follows the existing paradigm of hitting the Windows key and typing the search term. The larger search results improve speed (both for searching and browsing).

Of course, there are things we're still working on, that aren't yet finished in the Developer Preview. For example, we know there are bugs in interacting at high speed with the scroll wheel on the mouse, and we're working on fixing these. We're also adding the ability to instantly zoom out with the mouse and keyboard, and we're looking at ways to make scrolling faster and easier. And, we are working on fixing a bug in the Developer Preview that causes inconsistent and slow page-down/page-up behavior. We're also looking at making rearranging more predictable for mouse, keyboard, and touch.

One picture we often use to talk about change is the following. The y-axis is some measure of efficiency—such as time to complete a task, seconds it takes to do something, etc. The x-axis is calendar time. If someone is proficient with something and then a change takes place, there is by definition a dip in functionality. But after an adjustment period, the metrics of success improve. The net result is that over time, work becomes more efficient, even for the same task. And combined with new tasks and capabilities there is an overall net win.

Summary

The Windows 8, the Start screen is not just a replacement for the Windows 7 Start menu but a bringing together of several different ways of navigating your machine. Even in Windows 7, people who are proficient with Windows are already replacing the Start menu with the taskbar for their frequently used desktop apps.

For people using mostly desktop apps, the Start screen complements the functionality of the taskbar. Using both together, you have instant access to your most frequently used apps combined with a more powerful way to launch your less frequently used apps (through search or by grouping items on the Start screen). And, for Metro apps, Live tiles transform the Start screen into a dashboard that helps you stay up to date and connected in a high quality experience substantially improved over the notification tray. The new experience offers a way to more efficiently launch apps, stay connected to the most relevant information from apps, and find the things you care about. It also lets you launch and switch quickly between your apps and specific locations within those apps all without sacrificing performance or draining the battery of a laptop or tablet PC.

Alice Steinglass

Reducing runtime memory in Windows 8

Steven Sinofsky | [2011-10-07T08:45:00+00:00](#)

Fundamentals such as memory usage represent a key engineering tenet of Windows 8. In building Windows 8 we set out to significantly reduce the overall runtime memory requirements of the core system. This is always good for everyone and especially in a world where people want to run more and more apps at the same time or run on systems with only 1 or 2GB of memory. The laptop we talk about in this post is the exact same one we talked about at the Windows 7 PDC in 2008 – an off-the-shelf, first-generation, ATOM-based netbook with 1GB of memory. This post details our efforts around memory footprint and was authored by Bill Karagounis, the group program manager of our Performance team. --Steven

The runtime memory usage of Windows 8 is an important factor in determining the Windows 8 system requirements, as well as the broadened spectrum of devices that will host Windows 8. As you know, we're delivering the complete Windows 8 experience on SoC-based devices characterized by low power consumption. This makes it even more important to leave lots of memory available for multiple concurrent apps and to sustain the overall responsiveness of devices.

Something that might not be obvious is that minimizing memory usage on low-power platforms can prolong battery life. Huh? In any PC, RAM is constantly consuming power. If an OS uses a lot of memory, it can force device manufacturers to include more physical RAM. The more RAM you have on board, the more power it uses, the less battery life you get. Having additional RAM on a tablet device can, in some instances, shave days off the amount of time the tablet can sit on your coffee table looking off but staying fresh and up to date.

Memory usage goals

Our goal with Windows 8 from the beginning was to ship with the same [system requirements as Windows 7](#). We know if we do even better that there are more resources for apps, even if we keep the published requirements the same. It is fun to think about what the "low end" hardware looked like in 2009 and how you can't even find things like 256MB memory modules anymore. We wanted to ensure that people running on Windows 7-era hardware would have the option to easily upgrade their existing machines to Windows 8 and take advantage of the functionality it has to offer. We also expect that many machines that predate the Windows 7 release will run Windows 8 based on the experiences we've had with older machines we intentionally keep in our performance test infrastructure.

An important task for Windows 8 was to make room for new functionality while looking for opportunities to reduce the memory consumed by existing functionality and consumed across the board. Windows 8 is tracking well towards meeting the goal we set ourselves.

Task Manager memory use comparison

The easiest way to make a ballpark comparison of Windows 8 vs. Windows 7 memory use is to install both operating systems on a 1GB RAM machine (minimum OS RAM requirement) and compare them when they've been rebooted multiple times, and then idled for a while.

The Windows Task Manager contains the main view of system memory through its "In Use" statistic (described in detail in [this doc](#)). The below graphics compare memory consumption on Steven's 3+ year old netbook that he was using at the //build/ keynote recently, running Windows 7 at idle, and then with the same machine running Windows 8.

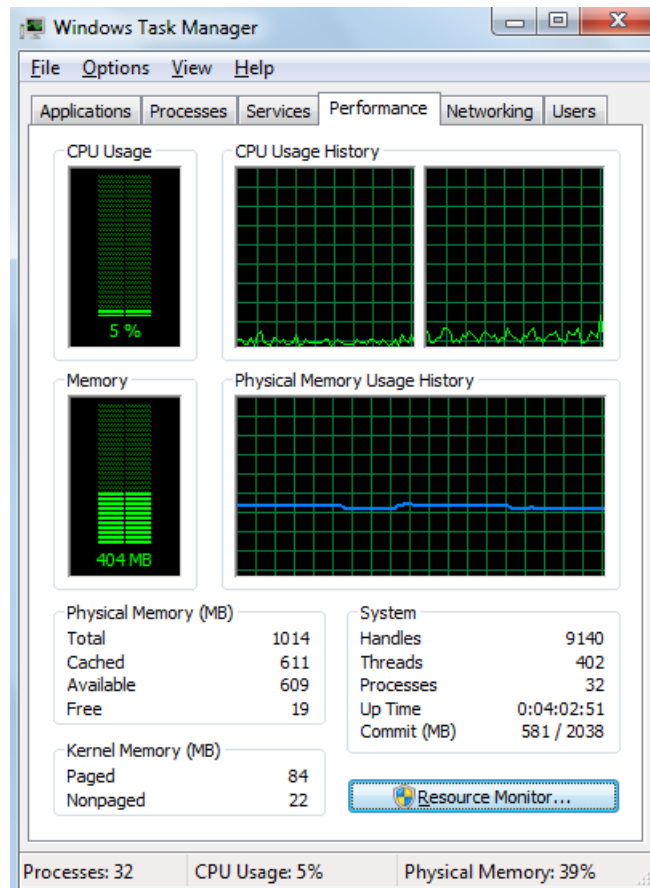


Figure 1 – Memory usage in Windows 7 SP1

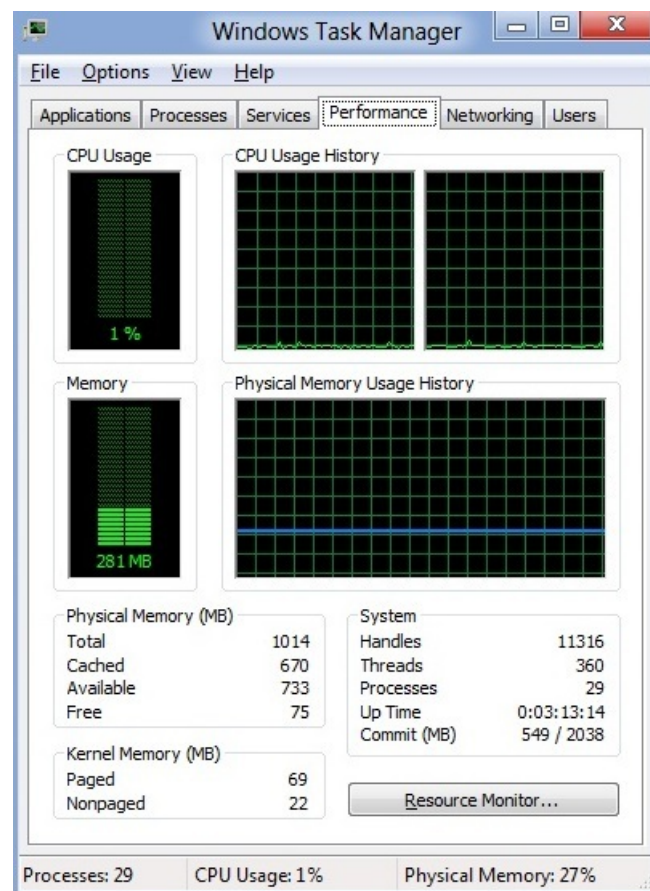


Figure 2 – Memory usage in Windows 8

The specific hardware making up a machine, memory use of drivers, and even uptime can cause variability, so memory results will be different on different machines (or even the same machine at different times). As you can see though, Windows 8 is doing well relative to Windows 7.

For a bit of extra fun on a test machine, go to device manager and disable your display adapter (unload the graphics driver). You'd never run your machine this way but this does give you an even closer approximation of the memory use of Windows itself. With a disabled graphics driver, the machine above gets under 200MB after idling for a while.

NOTE: For Windows 8, a clean install also contains the extended Windows Defender technology, which, for the first time incorporates complete antimalware functionality – also optimized for memory and resource use per [Jason’s blog about protecting you from malware](#). (This functionality does not exist on a clean install of Windows 7 where we would recommend that you add security software).

Making room in Windows 8

We made hundreds of specific changes to minimize OS memory use in Windows 8. I’m going to call out a few specific areas that resulted in substantial memory savings.

Memory combining

When assessing the contents of RAM in a typical running PC, many parts of memory have the same content. The redundant copies of data across system RAM present an opportunity to reduce the memory footprint even for services and OS components.

How can this happen? Applications will sometimes allocate memory for future use and will initialize it all to the same value. The application may never actually use the memory as it may be there in anticipation of functionality that is the user never invokes. If multiple running applications are doing this at the same time, redundant copies of memory are in the system.

Memory combining is a technique in which Windows efficiently assesses the content of system RAM during normal activity and locates duplicate content across all system memory. Windows will then free up duplicates and keep a single copy. If the application tries to write to the memory in future, Windows will give it a private copy. All of this happens under the covers in the memory manager, with no impact on applications. This approach can liberate 10s to 100s of MBs of memory (depending on how many applications are running concurrently).

Service changes and reductions

OS services configured to run all the time are a significant source of ambient memory use. When assessing the set of OS services during Windows 8 planning, we decided to remove a number of them (13), move a different set of services to “manual” start, and also made some of the “always running” services move to a “[start on demand](#)” model. This is where a “trigger” in the OS (like device arrival or the availability of a network address) causes the following to occur:

1. The service starts.
2. The service does its thing (whatever that happens to be).
3. It hangs around for a while to make sure there isn’t anything else to do, and
4. The service goes away.

You’ll notice that Plug and Play, Windows Update, and the the user mode driver framework service are all trigger-started in Windows 8, in contrast to Windows 7, where these services were always running.

Of course we have added a ton of new functionality (and new code) to Windows 8. Some of this new functionality is packaged in the form of new services. Of these new services, two are auto-started; all others are manual or trigger-started.

Doing the same job with less memory

As Windows executes applications and performs its own system housekeeping, program files and data are loaded off the disk into main memory. During Windows 7 and Windows 8 development to date, we’ve analyzed the pieces (pages) of memory during normal execution and how often they were referenced. The idea here is that if you’re going to pay the price for allocating a piece of memory, you’d better be using it (referencing it) often. If you’re not referencing that memory often but need it, consolidate it with something else.

Shortly after we shipped Windows 7, we applied a similar technique to several of the low level components of Windows dating back to the early days of NT (early 1990s). The work included re-architecture of code and changing data structures to completely separate “hot” parts of memory (frequently referenced) from “cold” parts. By densely consolidating the hot items, we brought down the overall runtime memory cost.

Given the nature of the changes (low-level OS), we wanted to get the work done as early as possible in the schedule to get ample runtime on the changes. To date, these changes have been in place on Windows 8 for almost 2 years with thousands of Microsoft employees using the product to get their daily jobs done. And we’ve seen consistent results showing memory usage reduced by tens of MB on an average machine.

Lazy initialization of the “desktop”

Back in June, you saw Steven and Julie introduce the Metro style UI for the first time. We expect many people using tablets to spend a lot of time in that environment, typically using Metro style apps. As part of that demonstration, we also showed that for Windows 8, you can also bring forward your existing applications and use them in the very familiar desktop environment.

From a memory perspective, we’ve taken advantage of the fact that there will be some set of devices on which users will stay in the immersive, Metro style UI almost all the time. In this instance, Windows 8 will only initialize OS components unique to the desktop environment when necessary. This is another source of memory savings, approximately 23MB right now. (Note that Task Manager runs in the desktop, so the memory numbers shown above include its cost).

More granular prioritization of memory

Windows 8 has a better scheme for the prioritization of memory allocations made by applications and system components. This means that Windows can make better decisions about what memory to keep around and what memory to remove sooner.

For example, antivirus programs (AV) do various checks on files when they are being opened by other programs. The memory that the AV program allocates to check virus signatures is usually a one-time allocation (it is unlikely that specific memory will be needed again). On Windows 7, the memory is treated as if it had the same priority in the system as other memory (say, memory allocated by a running instance of Microsoft Excel). If memory became scarce, Windows 7 could end up removing the memory that helps another running application (like Excel) stay responsive for the user, which wouldn't be the best choice for system responsiveness in this case.

In Windows 8, any program has the ability to allocate memory as "low priority." This is an important signal to Windows that if there is memory pressure, Windows can remove this low priority memory to make space, and it doesn't affect other memory required to sustain the responsiveness of the system.

To wrap up, I've called out our philosophy and approach to reducing memory usage in Windows 8. You've seen some sample results and I've just scratched the surface on some of the engineering work done to date in this area. One thing I haven't discussed at all is the Windows 8 application model, and process lifecycle changes made to make new Windows 8 apps more "memory friendly." Look out for this in the [//build/](#) content and in future blog posts, as it's also a really important part of the story of reimagining Windows.

We've already come a long way but we're not done.

--Bill Karagounis

Reflecting on your comments on the Start screen

Steven Sinofsky | [2011-10-11T14:00:00+00:00](#)

We've been having a lot of discussion regarding the two recent posts on the Windows 8 Start experience. Those of you who have used the Developer Preview are contributing to our understanding of your individual usage patterns and what is easier or more difficult than in Windows 7. As a reminder, we released Windows Developer Preview build with the full product "enabled" even though we still had much feature work to do in the user interface. We did this in order to foster the dialog and we want folks to understand that the product is not done. We've seen some small amount of visceral feedback focused on "choice" or "disable"—a natural reaction to change, but perhaps not the best way to have a dialog leading to a new product. We're going to focus this post on making sure we heard your constructive feedback around the design as we continue to evolve it. Marina Dukhon, a senior program manager lead on the Core Experience team, authored this post focused on specific comments and the actions we are taking based on what you have said. --Steven

On behalf of the team, I want to thank everyone for their active engagement on the Start screen blogs over this past week. We have been following all of the comments and responding as much as we can. We know major changes like this can be controversial and we are looking forward to continuing this dialog with you. I wanted to address some of the specific topics that have been brought up so far as they pertain to the design. I know this doesn't address all of your questions, but rest assured that we are listening and will be continuing this ongoing conversation.

Does the data support all customers?

@Andrew wrote:

"I'd like to point out that this data you collect is most likely from non-corporate users, you're basing all your statistics around home users and not business users. Most enterprises will turn off the CEIP by default in Group Policy as a security precaution and to prevent chatter from the network."

Andrew, while it's true that some enterprises choose not to enable the CEIP (Customer Experience Improvement Program, which gives us anonymous, opt-in feedback about how people are using Windows,) we still receive a huge amount of data from this program, including from enterprise customers. In addition, knowing the region, language, edition, and deployment attributes of the product allows us to further refine the data as needed. We often refer to this data as a full "census" (again noting that the data is opt-in and anonymous) as the number of unique data points is magnitudes beyond a "sampling."

In addition to the CEIP program, we have a wide variety of channels to our corporate customers to understand their needs. For example, we collect feedback continuously during direct engagement with customers (such as during on-site visits and in our briefing centers around the world), from advisory council and early-adopter program members, and at public events such as [TechEd](#) and [//build/](#). We also work closely with industry analysts (via consultations and their research) and execute a wide range of our own research studies directly. From these interactions, we know the kind of functionality and control that enterprises want over the Start menu and we are definitely taking these into account as we are designing and developing the changes for Windows 8.

When you look at the data, we can see that enterprise customers do, in fact, have some different experiences with their Start menus:

- While 81% of home users have the default links like Control Panel, Games, and Documents on right hand-side of the Start menu, fewer than 2% of our enterprise customers have this experience.
- Most people have removed some items in this part of the Start menu (with Games and Media Center entry points most often removed).
- Enterprise users are launching pinned Start menu apps 68% more often than home users, but the usage of pinned items is still less than 10% of the sessions.

What are we doing with this information?

In general, individual enterprise customers are using Start menus that their administrators have customized. Using this research and our engagement with the enterprise community, we are working on special features that can help address the need for customization in the Start screen. For example, enterprises can remove items like Games and Help & Support from the Start screen. For Windows 8, we support deployment scenarios that include Start screens with a layout of tiles that matches their business group's needs, allowing for an even greater number of pinned apps to be pre-defined for their users. We also support the managed lockdown of customization of the Start screen so that it is consistent across the corporation. These features have been built especially for our enterprise customers, taking into account the existing functionality that we have provided in the past and the needs that we perceive they will have in the future. And as many know, tech-savvy individuals can use these customizations as well.

Is Start less effective for "at a glance" viewing of my PC?

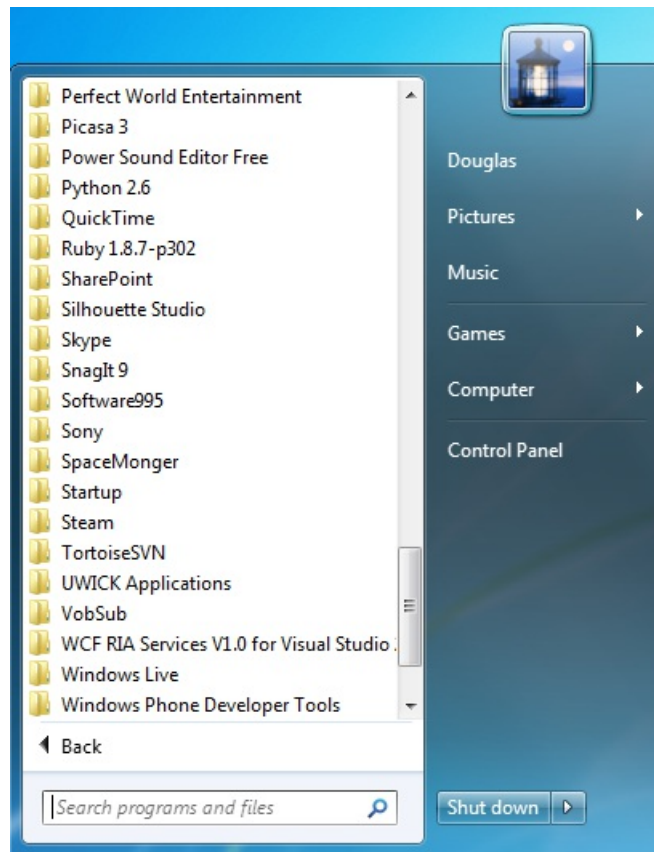
@mt327000 wrote:

"The Start Screen feels like a mess of icons, having all of the problems with the Start Menu you described and adding some of its own. I keep a very neat and orderly desktop, and can see everything on my computer in a glance in the "All

Programs" view introduced in Windows Vista. To me, the Start Screen just doesn't work, nor does it have any advantages over the superior Start Menu."

The comments have been very clear that knowing what's on your PC and seeing it at a glance is an important aspect of feeling in control. Let's talk a little bit about how this works in the Start menu and how it compares to the Start screen.

In the Start menu today, when you expand the All Programs flyout, by default you can see a total of 20 apps without scrolling, regardless of how big your monitor is. In one of our studies, we found users launched an **average of 57 different apps** over the course of several months. And this doesn't even include the large number of websites that people use day to day (for the purposes of launching and pinning we believe counting websites is important), some of which may evolve into Metro style apps. So you can see how a little window that shows 20 items does not prove scalable in this scenario. The comments have been clear that this scale is routine for those that are reading the blog, and we believe your usage would skew in this direction.



Once apps are installed on the machine, you'll likely need to scroll All Programs view in order to see all app folders

In addition to the limited real estate, apps in All Programs are buried under folders and subfolders of hierarchy, without any iconography to help you navigate to the right place. To make matters worse, things are often jumping around as you expand and collapse folders looking for the right app, making the experience even less efficient. Some have noted that this limitation is a design regression from the Windows XP Start menu. While technically that is true, we are fundamentally working with a menu, and as such, it is a single column with hierarchy that requires significant dexterity to navigate. The feedback around the scale of the old Windows XP design was resoundingly negative over time and led to the redesign for Vista and Windows 7.

In Windows 8 we assume that there are even more apps (and sites) than the XP/Vista/7 eras and so we needed even more scale. We also wanted to provide an at-a-glance view and a navigation model that requires much less dexterity. By using the full screen, we can now show more apps without the need to scroll or navigate hierarchy. By flattening the hierarchy, we provide a way for you to leverage the iconography of the apps and remove the burden of clicking through folders trying to find an app under its manufacturer's name. Over time this will also address another common complaint, which is that when renaming, combining, or reorganizing folders (which you might do in order to keep the menu from wrapping) you would lose the ability to uninstall cleanly, and thus subject yourself to a periodic garbage collection of your Start menu to avoid dead links.

As we will talk about later in this post, the dexterity required to navigate a very large menu interface is inconsistent with good user interface design. Even if the items you wish to target are rarely targeted, the whole experience is degraded when constrained to a menu. Some have suggested that using XP-style menus that wrap around the screen, or increasing the size of today's Start menu would "solve" the issues we are working to solve. Below we will talk about Fitts' Law and how no increase in size or wrapping will address this. As DPI and monitor sizes increase, it becomes increasingly difficult to zig-zag around the menu to hit narrow buttons. Here is a screen shot submitted via a comment by @Bleipriester, where you can see the mouse "path" required as well as the additional navigation aid of the down/up chevrons. Keep this in mind as we discuss Fitts' Law below.



@Bleipriester's evolved Start menu, showing several columns of navigation

Thus as your monitor gets bigger, the Apps screen (an all-apps view of the Start screen) becomes more powerful. Here is how the number of apps that show up in the Apps screen grows across different monitor sizes in our latest builds:

Likely form factor	Size (inches)	Resolution(s)	# Tiles on 1 page of Apps screen	# of Items on 1 page of All Programs
Laptop	12.1	1280x800	36	20
	13	1366x768	40	20
	13.3	1440x900	42	20
Desktop	21.5	1920x1080	80	20
	23	1920x1080	80	20
	27	2560x1440	150	20

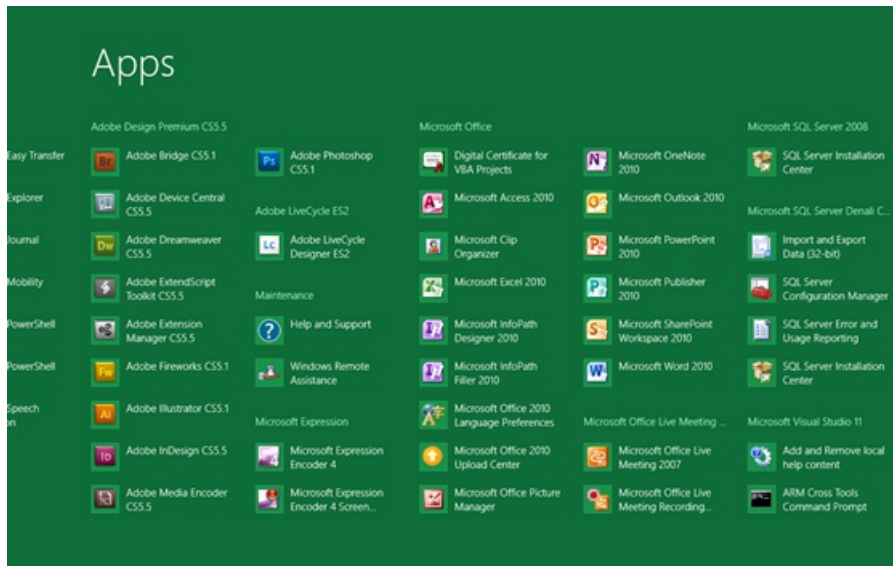
Estimated number of apps visible on the Apps screen across different monitors

Your comments have been clear and we agree with many of the design issues you've raised. Some of you have mentioned how it's difficult to find an app when its folder name is no longer available and how completely removing the folder structure has made it difficult to find an app that came in a suite.

To cite @aroush:

"The current metro-list of all apps is not suitable, since that lists everything alphabetically and I don't know the names of all those additional programs."

We are working on addressing this feedback as we speak. **Here is our latest design of the Apps screen, which would add back the structure that you're used to with folders in All Programs today.**



You can see here that, as in the Start menu, suites of apps are now organized in groups, instead of in one alphabetical list. This way, if you are looking for something that you know came in your Visual Studio suite, but can't recall the exact name of the app, it should be much easier for you to find. And your alphabetical list should no longer be cluttered with app tiles that have obscure names because the developer was relying on the folder name to convey the actual name of the executable.

In addition to adding folder structure to this screen and organizing apps within their respective suites, we are also making this view denser. Fitting even more content helps you see what your computer has installed at a glance and decreases the need to scroll. It also decreases the need to navigate a wrapping menu structure or maintain folders or nested folders of programs.

With this design, we improved the scannability of your system, giving you confidence about what is on it at any given time.

Does the new Start support the kind of customization I require to be productive for my work?

@Ed1P wrote:

"While I can see that the Metro style Start replacement works well for touch screens on small form-factor computers it will dramatically reduce my productivity on a desktop with a large widescreen monitor. I have 50 apps+folders that I visit regularly during the course of a working session. I do not now use my customized Windows 7 Start Menu (yes you CAN customize it, and do all the things Alice says are impossible, just by right-clicking on 'All Programs' at the base of the start menu and rejigging the Program Folders), Instead I now use the free Stardock Fences app which allows me to group these as immediately accessible tiles on the screen.

I recognize the similarity between my groups of Stardock Fences and Metro Start Screen 'pages', however the one big difference which makes Fences more productive than Metro is that I can have them grouped and pinned vertically on the left hand side of my screen, leaving the right hand side free for live update gadgets and the center as a very workable 1200x1024 area --- I can happily code or 3D model in this area and instantly switch what I'm doing while still keeping an eye on Live updates. I rarely use the taskbar, it just becomes an autohide alert area.

Metro would be much more usable for my desktop layout if it were possible to use it vertically and pin it to the side. Even better, if it can be split in two vertically scrollable areas - Live updates/gadgets in one area and app, folder/file launching in another, leaving me a large area of working screen real estate in the center."

Thanks for writing this up. You've obviously taken a lot of time to customize your machine to get it how you like it. This is a good example of how Windows is able to provide flexibility to our wide breadth of users. We will continue supporting such flexibility in Windows 8 and we expect there to continue to be a wide array of 3rd party launchers available to users to meet their specific needs. A good deal of obvious extensibility was intentionally omitted from the Developer Preview and will be there in the final product—colors and backgrounds, for example. But let's focus on this advanced level of customization.

The level of customization that you have applied to your machine is certainly something that we consider an "advanced" user might do. Your level of advancement is also apparent in your app and folder usage within a working session. The table shows what we see people doing on their

machines during the course of a working session:

Peak number of open windows	% of sessions
0-5	20.40%
6-9	49.30%
10-14	21.30%
15-19	4.60%
20-24	2.69%
25-29	1.30%
30-39	0.23%
40-49	0.08%
50-59	0.03%
60-79	0.03%
80-99	0.01%
100+	0.03%

The maximum number of windows people have open at a given time during a session

So you can see that your numbers are certainly beyond our “average” user, but we do have users of all levels using our system. At the low end, some folks might say this represents a “quick” session where you log on to do one thing and then log off (and even professionals do that). At the high end, this data might also include people who inadvertently launched malware and had tons of open windows. That’s why when looking at the data in aggregate we are confident that the averages tend to work out to be realistic. We know that there’s a tendency to try to use data to make one point or another—that’s why we want to provide the full context of the data here and make sure any limitations are understood. We provide this data to illuminate choices in the design, not to prescribe them.

While some might say we design the system for the low-end, that is not the case at all. At the other end of the spectrum, we hope everyone can see that designing the system for the high end would put a conceptual burden on broad set of customers. Our design point is to focus on a sweet spot and to provide the flexibility for the high end. There's nothing new to our approach here and it is how we approach Windows design overall.

One of the popular aspects of Fences is that you can group your items together in a logical manner and even name your groups. But you also pointed out the difficulty in this design – the groups are on the desktop, which inherently sits underneath all of your open windows, making it difficult to get to while you’re in the midst of working on something. Since I don’t know what your setup looks like, it’s hard to know if my assumptions below are correct, but perhaps I can assume what at least some with this approach might have to manage around routinely (though from the sound of things you work hard to find a careful balance). One would spend time reorganizing the workspace to allow open windows to sit next to your launcher, allowing yourself to quickly access the launcher and keep an eye on the live updates, but at the cost of less screen real estate and more manual and fragile window management.

The value of arranging content on a 2-D plane

Another important aspect of Fences is the spatial arrangement that you can use to organize shortcuts. We know that remembering where something is located is much easier in a 2-dimensional space than in a 1-dimensional list. Our brains are naturally inclined to remember location, in addition to other properties like color and size. So finding an item that you already remember is in the top right of your screen is often faster than scanning through an alphabetical list. Another common critique of Start menu folders is that they all start with the same letter and differentiating

requires reading several words in (for example, graphics professionals have a lot of folders starting with "A" for one manufacturer of those tools).

There is a large body of research to support that having multiple characteristics or attributes makes it easier to locate a specific item quickly and efficiently. Windows already takes advantage of this, by showing details about files or search results, or showing both a thumbnail and a title for windows you have open. We designed the Start screen to take advantage of characteristics of human cognitive processing. These characteristics are basic neurological patterns baked into the evolution that got us to using computers in the first place:

- **Human spatial memory** - Your ability to remember where you put something or where something will appear. This also includes taking advantage of spatial relationships, how different items are located in space relative to each other.
- **Muscle memory** - A motor task that becomes automatic and can be performed without conscious effort.
- **Chunking** - Grouping of items to make them easier to recall later.
- **Signal detection theory** - Your ability to identify an item of interest even when there is lots of "noise" or items which are not of interest.

We wanted to create a design that capitalizes on these attributes. With the All Programs view and the Most Frequently Used (MFU) or Pinned lists in the Start menu, we were very limited in terms of space and layout. It is impossible to develop a rich spatial framework with a one-dimensional list. With the Start screen we can take advantage of a two-dimensional space. Microsoft Research has demonstrated in a series of different research studies, including their work on spatial memory for [document management](#), for [information retrieval](#), and on the [Task Gallery](#), that it is possible to improve retrieval of items, even after 6 months of disuse, by adding richer organization over one-dimensional visual text lists. We wanted to take advantage of this effect to make it faster to locate specific apps on the Start screen.

Many have mentioned using large monitors or multiple monitors. While the immediate reaction has been that the Start screen is less optimal for this approach, our design goal has been precisely to bring enhanced functionality for this environment. As with many cases, it should be no surprise to learn that the development team comprises a large number of very high tech power users with multiple HD+ screens running many Win32 applications all the time. The Start screen on a central monitor allows for the most rapid "in and out" of launching and switching when you are using a large number of apps and sites. And at the same time, the ability to have a heads-up display of status across a variety of (yet to be written) business applications will provide a new level of functionality.

Taking advantage of spatial arrangement on the Start screen

The grouping of tiles in the Start screen was designed with these principles in mind. We know that sizes of groups will naturally vary based on the kinds of items that you're throwing together. Not only does this flexibility help with organization, but it also helps by creating a heterogeneous layout where shapes and sizes vary from group to group. This makes it easier to find a tile when you know it's in a small group with an uneven edge on its right side or in a large group that looks like a full rectangle.



Start screen layout takes advantage of position, shape, co-location, and color to help you find apps

In addition to group sizes and shapes, I can leverage several other factors to find my tile. Whether it's because it is at the top right of a group (the red tile), next to the wide green tile in the big group (the black tile), the first square tile at the top of my big group (light blue tile), or the last tile in my Start screen (yellow tile), I have several attributes I can now rely on to find something. The same thing happens when you look at groups of tiles - I can use general color and group shape to identify the group that contains my games or the group that contains my news apps as I scroll through the screen.

Explaining spatial recognition through evolution

From an evolutionary perspective, this type of recognition is rooted in our most basic survival skills in our subconscious. Humans use more than one sense to map a stimulus. You need to locate each stimulus (where is it?) and triage it (will it eat me?). You also need to remember it for future processing and comparison. The key to making this **fast and fluid** is to present enough information that you can select correctly and remember your selection, without taking so much processing that your brain needs to pause to interpret what it has just perceived.

If all of this sounds familiar, it is basically why iconic presentations tend to be more efficient. It is also why irregular patterns can provide visual cues that reduce the need to process information, and rely just on sensory-motor skills. And of course, it is why large blocks of similarly formatted text in a menu (or graphical buttons) can take the most time and brain processing power. Here's a good layperson's article on [elements of visual perception](#) and of course there are many deep technical articles as well.

Incidentally, some folks have suggested we use less spacing, more transparency, or rounded corners to add more visual "candy" to the design. The clarity of spacing, solid edges and backgrounds, and rectangles is a significant improvement in the ability to identify your programs and to prevent overloading your brain causing headaches and the like (see this University of Massachusetts examination of the [edge enhancement illusion](#) and this one on the [value that colors provide](#)). Essentially these aesthetic additions trick your brain into thinking it needs to spend more time "understanding"

the stimuli rather than just reacting to what it perceives.

How we are making customization better

In terms of customization, you are definitely correct in saying that today you can customize the existing Start menu. The method that @Ed1p mentioned allows you to rename folders (breaking uninstall), move around files (breaking per user and per machine setup) and basically reorganize the tree of apps that exist on the system. For those brave souls out there who want to use drag and drop within the Start menu, this is also possible (albeit highly error prone).

However, these are very advanced ways of customizing your system, and unfortunately do not scale to a broad set of customers even if we initially intended them to. Not only do they take a lot of time, but the method is indirect since you're not actually working within the Start menu. So it requires a lot of burdensome back and forth between Explorer windows and menu flyouts to get to the final result.

The personalization of the Start screen is one of the features that we want to make great, and we're still iterating on it and to make it better. In the Windows Developer Preview, you can already try flexible group sizes, unpinning tiles, and resizing wide tiles to square tiles. **And in the Beta, you'll also be able to use other improvements based on this dialog, in addition to creating, naming, and rearranging groups.**

@drewfus pointed out:

"When i said 'The list of apps (and hence tiles) on a PC is neither known nor fixed', i was alluding to the fact that this list is not constant - it grows over time, but more importantly that the chronological order of additions in no way matches the importance of new additions (except by coincidence), resulting in a constant impact on the users existing Start layout."

This is a good point – your set of apps is likely going to continue to grow and change over time and you may find your new favorite apps months after you first organized your Start screen. Our goal is to balance your ability to keep control over your Start screen (i.e. not impacting what you've already organized when you acquire new apps by putting them at the end), while also making it simple to change it when you want. Group rearranging helps enable the particular scenario that @drewfus mentions – as you get more apps over time, it's quite possible that your new favorite apps are now at the end of your Start screen. With group rearranging, we make it easy for you to move an entire group of apps to the front, without having to move them one tile at a time and you can just as easily demote a group of apps and put them at the end.

The Developer Preview was obviously incomplete in this regard, and given the importance we attach to this, we fully expect to land on a solution that combines flexibility with overall improvement that justifies the change from previous products.

The ability to put apps where you want them in a spatial layout, to use groupings to better enable recognition, and to move the tiles around on the screen should be a vast improvement over the Start menu. We believe this opens up a whole new world of organization and customization that will dramatically improve working with extremely large sets of apps and shortcuts.

Did you just make us invest in jump lists and then take them away?

@tN0 wrote:

"Implement Jump Lists to the Live Tiles at the Start screen. Swiping up on a tile or right click could bring up a Jump List."

Having a way to quickly access content within an app is a great feature and we're happy to see the enthusiasm and increasing usage for jump lists in Windows 7. We have developed something new for Metro style apps that builds on the jump list concept. We think it will be even more powerful for end-users and an even richer opportunity for app developers. But first, some background on jump list usage in Windows today.

Current usage of jump lists

Though jump lists are often referenced with positive energy by our enthusiast users, the fact of the matter is that the usage of jump lists in the Start menu (most recently used documents for an app, for example) has not really gained as much traction as on the taskbar. To compare, 20% of sessions record a click to open a taskbar jump list, while only 1.2% of sessions record a click to invoke a Start menu jump list. People also use hover to invoke the Start menu jump list (and drag to invoke the taskbar jump list), but it's difficult to use these numbers because we can't tell whether the menu was opened intentionally or simply because the mouse was hovering over the item long enough to trigger it. Either way, even with accidental activations via mouse hover, at best, the Start menu jump lists are used half as often as those of the taskbar.

Applying this to Metro style apps

Given this data, we knew it was important to keep jump lists on the taskbar for your most commonly used desktop apps. But, we wanted to build something more customized for Metro style apps. The downside of existing jump lists is that they're limited to what Windows understands best—files. This is great for file-centric apps, but apps today are moving away from the notion of files and turning to hosted content, which makes the concept of document jump lists less relevant.

Instead of building on and promoting file structure, our view for Metro style apps is more app-centric. The apps know better what kind of content they host: whether it's an RSS feed, an album, a score tracker, or a person's profile, and they can do a much better job exposing quick access to this content to the user. This content doesn't involve files on the system that Windows knows about – it's knowledge within the app. We've expanded the jump list concept to provide semantically richer links.

But we don't want to have to manage several lists of our favorite stuff. One of the promises of the Start screen is that it is your personal place to host the apps that you love. We based the secondary tiles feature, on the notion that people want fast access to app content that they require for work, and they want a single, predictable place to access it. With this feature, any Metro style app can allow a user to pin a new tile to their Start screen that can navigate them to any part of the app. The tile can even be live, providing updates for that specific content. There's no reason a file-centric app would not provide this same functionality for files. We know from usage data that people are fairly meticulous and deliberate in reusing common documents—MRUs composed of pinned files are extremely popular in Office apps and on the taskbar. The support we provide for developers makes this straightforward.

For example, I can have a social tile of my best friend pinned to my Start screen and keep up to date with her updates. Or I can track the XKCD feed from my RSS reader. Or quickly jump to a playlist that I like to listen to in the morning the same way I would have from a jump list. We expect line of business applications to allow this “deep linking” to specific machines for monitoring, account information, or other exception handling (as we described with our bug tracking application). All from the Start screen. All of these organized among other apps that I like to use, so they are fast to access and get me quickly to the content that I want to consume.

Building on secondary tiles

We're continuing to invest in enabling Metro style app developers to provide personal and rich content to their users through live tiles. Secondary tiles will be a big part of making your machine feel more useful and personal, and something that you love to use. To help, **we're building even more live tile templates into our catalog** so that developers can enable more scenarios for their users.

Overall, isn't this a real usability problem?

@mt327000 wrote:

"All the requests for a return of the classic Start Menu are not just complaints about change. To me, the new Start Screen actually feels less efficient than the Start Menu. I will admit, some commenters on this blog have gone too far and resorted to mudslinging to make their point, but from a scientific perspective, if you measure usability of Windows 7 and of Windows 8 in terms of click counts, Windows 7 wins hands down. This is not simple complaining, but a real usability problem that Microsoft will hopefully fix."

We do have to assert that efficiency, that is, time to accurately complete a task, is of paramount importance in design. We never say "most important" because we consider a broad range of attributes in designing how a feature works (resource utilization, reliability, accessibility, localizability, security, training, discoverability, and so on). As we work to improve our products, both in terms of efficiency and usability, we consider several factors for user interface approaches, such as mouse *mileage*, target size, loading time, parsing time, and mouse click counts (among others). It's likely that in any change, there are efficiency gains and sometimes efficiency losses, but we take great pains to achieve a *net gain* in efficiency when all of these are considered.

One common theme in the comments has been an immediate rejection of change with the assumption that any change will reduce productivity so much that it will never be regained. One analogy we use looks at improvements in roads or traffic flow—for example, a new lane or exit. These types of projects might take years and during construction, we all might get frustrated at how much time we lose. But once the project is done, our use of the road is improved every single day, and so is the usage by everyone else—the net gain is to the whole universe of travelers, present and future. This comes at some near term cost to current users, but the net is an improvement for everyone. Yet we know that during construction we're all the type of folks who sit and calculate whether we will ever make up for the time lost—this is the concern we hear. Unlike road construction, we design our changes to Windows so the payback comes for everyone in the span of hours, days, or perhaps weeks. If improving traffic flow started from the premise that no one would be interrupted even for a little bit, then there would never be any improvements and everyone's usage would gradually decay. With Windows we see the same challenges—we need to improve the product for new uses and new hardware capabilities, and as such, there is always some transition. Much like engineering roads, you don't keep both paths open and operational in parallel. But fortunately, unlike construction, you can control your own PC and can choose to switch when you want. This is especially the case for businesses as we commit to a 10-year minimum lifecycle.

One small example of this net gain is the ability to press the Windows key and immediately start typing to search for an app. Even though the search box doesn't appear on the screen, we did extra work to make sure you can type right away, thus protecting the efficiency of searching for apps. Our design choice means that there is a short period before people discover this feature, but once they do, they see a huge efficiency gain. As a practical matter, the discoverability of this feature usually happens within hours of usage of Windows 8, as we have seen in the tweets regarding usage of the Developer Preview. Even if it doesn't, the search command is in fact still there—the edit control is two clicks away. And we make things better for everyone by not having the UI clutter.

Mouse distance and mouse clicks

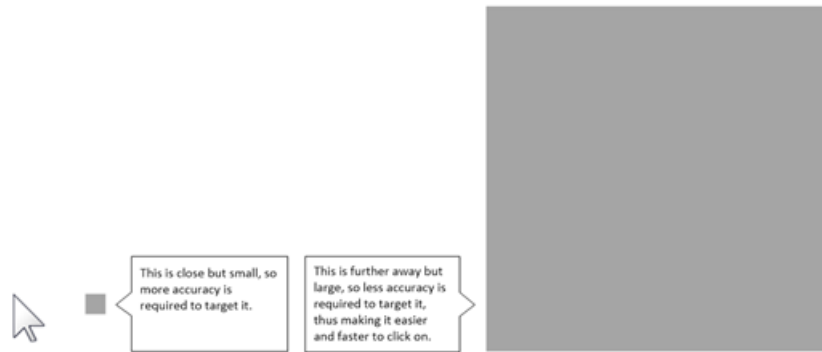
There has been a common thread in the comments when discussing efficiency that focuses on number of mouse clicks and mouse travel distance. Though these are important measures of efficiency, another factor that strongly plays into this equation is the target size. Many of you already know about [Fitts' Law](#), but let's do a quick summary of what this is and how it applies to software.

Fitts' Law is named after Paul Fitts, a psychologist at Ohio State University, with expertise in aviation. He developed his research to model cockpit ergonomics and created a model that was formulated to project how quickly a human can point at a physical button. Soon after, people started applying this model to software, tracking how quickly someone can target something on the screen with a mouse.

The mathematical formula is somewhat complex, but the basic premise is as follows:

- The farther away a target is, the longer it takes to acquire it with a mouse
- The smaller a target is, the longer it takes to acquire it with a mouse

So the speed with which a target can be clicked on with a mouse is a factor of both size and distance:



The closer the target, the faster you can hit it. The larger the target, the faster you can hit it.

One common formula that can be used to compare two hit targets more mathematically is the Shannon formulation:

$$T = a + b \log_2 \left(1 + \frac{D}{W} \right)$$

Where:

- T is the average time taken to acquire the target.
- a and b are empirical constants determined through linear regression.
- D is the distance from the starting point to the center of the target.
- W is the width of the target measured along the axis of motion (how close to the target you need to get to acquire it.)

How does Fitts' apply to Windows 8?

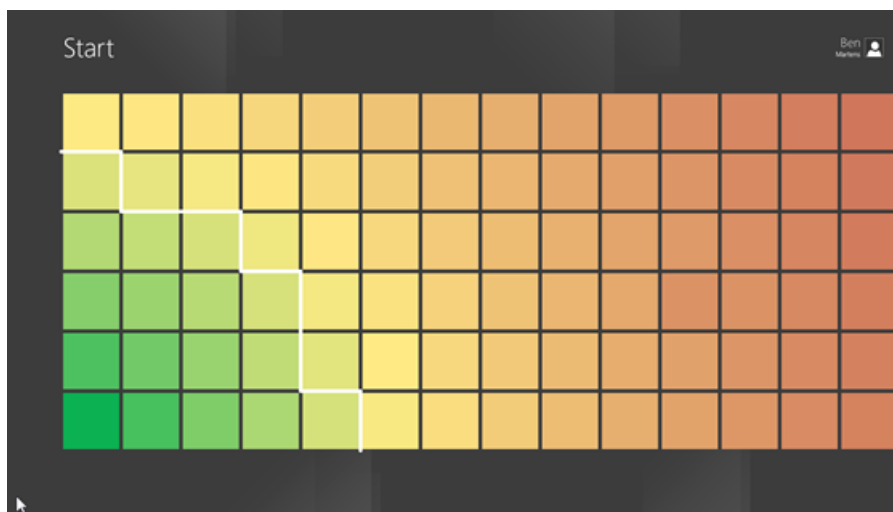
One of the most obvious ways to apply this in Windows 8 is with the Start button. Although we optimized Charms for touch (with the Start button accessible with a swipe from the right edge of the screen,) we preserved the notion of a control in the far left corner for mouse users. The corners are considered infinitely wide when it comes to Fitts' Law, which makes UI in this location the easiest to target. It was important to keep the efficiency of the Start button high for our users, so we were adamant about making sure that this is not something we lost as we created a new UI paradigm.

The other obvious example of Fitts' Law in action is the Start screen. In general, tiles are further away from your mouse cursor than entry points in the Start menu, but they are also larger in size, which helps negate the efficiency loss that was introduced with distance, and even brings efficiency gain.

We took a look at desktop monitors, and by controlling for constants a and b because we're on the same device, and varying D and W based on the targets in the Start menu and Start screen, we calculated the speed of acquiring an app link. We then applied a heat map to show the results and see the following comparisons:



*Heat map of time to reach items in the Start menu from the Start button
(green items are the fastest to get to, red items are the slowest)*

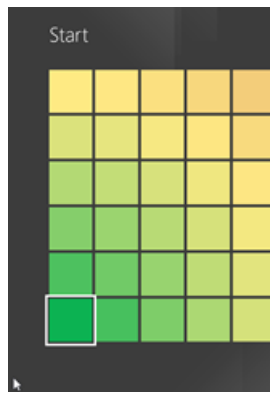


*Heat map of time to reach tiles in the Start screen from the Start button
(green tiles are the fastest to get to, red tiles are the slowest)*

If you count the number of items that show up as green (delineated with the white line,) it is considerably larger on the Start screen (about 17 square tiles) than on the Start menu (2 apps). So there are many more items that you can reach more quickly on the Start screen.

In the Start menu, the top item (which is usually the most frequently used app or your favorite pinned app) is closer to the darker red, which is unfortunate. Lists are generally ordered top-down, which is why the Start menu used this logic, but to really emphasize efficiency, it would have been better to flip the order here and put it at the bottom of the list. Whereas in the Start screen, the bottom left tile is the easiest thing to get to with the mouse and even easier than any item on the Start menu.





The app that you're using most frequently is further away on the Start menu than on the Start screen

It took us much iteration over the course of many months to get to the final size and shape of the tiles. As you can imagine, we iterated through many possibilities and tried many of them out in the lab. We asked test subjects to target a variety of buttons, much as you could imagine Fitts optimizing an air force cockpit design. Mouse distance (and touch target size) is just part of the story. In addition to these, we also considered the following factors when coming up with the tile size:

- **Screen size** – How many apps should be visible on one page of the screen across monitors?
- **Form factors** – How does your usage of different form factors affect your need for something to be smaller or larger (e.g. when you're sitting on the couch with a slate vs. sitting further away from a large monitor on your desk)?
- **Efficiency of scanning** – How do we provide enough breathing room to make it easy to scan the content, while also providing enough density and useful information?
- **Layout** – What layout works best for scanning a grid of content, and how should different tile sizes relate to each other for easier parsing?
- **Space for live content and app branding** – Tiles need to be big enough to provide useful information, but not so large that the amount of information displayed is overwhelming. And this also needs to be balanced with being able to actually launch your apps without requiring a lot of scrolling.
- **Visually pleasing shapes** – The tiles need to be visually pleasing, and the shapes that they create when laid out on a page also need to appeal to the eye.

This is just a sample of some of the questions that we were asking ourselves when designing the size of tiles and the density of the Start screen. The end result is our attempt to balance efficiency of mouse movement, mouse targeting, parsing, and ability to see live data at a glance across various form factors and screen sizes to make the system feel powerful and efficient to use.

So, how many clicks does it take?

As Alice mentioned in a previous blog post, the current Start menu is primarily used for launching infrequently used apps, while users continue to launch more frequently used apps from the taskbar and Explorer. In fact, 88% of app launches are from outside the Start menu today. Instead, most launches are from the taskbar (41%) and the remaining are split between Explorer and the desktop (47%). So it was clear to us that the Start menu was trending away from being useful and we had an opportunity to redesign it to make it more useful and valuable. We want to be careful in this dialog of spending a lot of energy debating what amounts to a “long tail” usage case.

However, once we left that old paradigm, the next question was – how can we complete the same tasks without requiring more clicks? We kept this in mind throughout the design process, and once we had a design, we picked a couple of different tasks to compare click-to-click.

Launching an MFU or pinned app

How many clicks does it take to launch an app on the left side of the Start menu?

In Windows 7, if we assume your favorite program is in the left pane of the Start menu, it takes 2 clicks: one for the Start button and one for the app itself. It was important to us to keep this parity for the Start screen, so if an app is in the first page of the Start screen, it also takes 2 clicks to launch it.

However, the number of apps that gain this “2-click” benefit varies across the two UIs. By default, the Start menu provides 2-click access to 10 of your favorite apps, plus 10 special folders that Windows adds for you – few of which are used frequently. The highest usage item here is the Computer folder, with about 8% of sessions using it, and the numbers for the rest of the items drastically drop off. Also, while this area of the Start menu allows some limited customization, 81% of home users keep the default behavior.

In comparison, the Start screen provides 2-click access to many more apps and allows you to control the full layout of the screen. If you don't want a link to Help and Support, don't put it there – instead, use the space for your favorite app. And the number of apps that get this ability only increases the larger your monitor is. Incidentally, we made the customization much easier and you will no longer break out add/remove programs when you organize things. You can see below how many more tiles you get on one page as your monitor size increases.



Form factor	Size (inches)	Resolution(s)	# of tiles in 1 page of Start screen	# of items in Start menu
Slate	10.1	1366x768 1920x1080	12 wide or 24 square	10
	10.6	1366x768 1920x1080	12 wide or 24 square	10
	11.6	1366x768 1920x1080	12 wide or 24 square	10
Laptop	12.1	1280x800	16 wide or 32 square	10
	12.1	1366x768	20 wide or 40 square	10
	13	1366x768	20 wide or 40 square	10
	13.3	1440x900	25 wide or 50 square	10
Desktop	21.5	1920x1080	36 wide or 72 square	10
	23	1920x1080	36 wide or 72 square	10
	27	2560x1440	42 wide or 84 square	10

How the Start screen scales with larger monitor sizes, compared to Start menu

In addition to the difference in the number of apps shown, the logic for what you see after you click the Start button has changed. The Start menu uses heuristics to calculate the MFU (most frequently used) apps that appear there. Unfortunately, these complex heuristics are sometimes wrong, and so the set of apps that you see here changes over time, adding a level of unpredictability into the launcher. On the other hand, the Start screen puts more value on user control and predictability, encouraging customization and increasing confidence about where things will be—a design goal that we followed as we designed the taskbar as well.

Launching an app from the All Programs list

The number of clicks to launch from the All Programs list varies depending on what you're launching (is it closer to A or to Z?) If we were to generalize to a user who has some apps installed on their system, the most likely workflow is something along these lines:

Start button → All Programs button → Scrollbar button → Expand the folder of the app I'm looking for (cross your fingers it's the right one!) → App = **5 clicks**

In the Start screen this flow is different, but looks like this for the same scenario:

Start button → Hover in corner → Search button to launch Apps screen → Scrollbar → App = **5 clicks**

This comparison leads to the same number of clicks when using the All Programs feature as when using the Apps screen, assuming you expanded the right Start menu folder the first time. Also, since you're using more of your monitor with the Start screen, it is more likely that you won't need to use the scrollbar to find the app, decreasing this to 4 clicks in Windows 8. You can see how other tasks, like launching one of the items in the right side of the Start menu (e.g. Control Panel or Computer) would also show the same number of clicks between the 2 UIs.

We would find the same results relative to keystroke counting as well. We have been careful to at least maintain parity and often improve relative to these measures.

Launching from other parts of the system

As I mentioned previously, 88% of app launches don't actually originate from the Start menu. The rest of the launches are from the taskbar, Explorer, and the desktop, and the math here does not change in Windows 8. In order to be complete, however, it's worth mentioning that there is a one-time additional click to get to your taskbar or desktop when you start up your machine, since we boot the machine to the Start screen. In the grand scheme of things, with all of the clicks that you do throughout your working session, one additional click to get to the desktop does not impact your overall efficiency, but since some people are asking about this, I thought it would be worth talking quickly about why we do this.

Since the Start screen is a launcher (and can also be the switcher) for both Metro style apps and desktop apps, we take you directly to the Start screen when you first turn on your machine. It is your new home base. This allows you to make a choice in terms of what app you want to launch first – it may be a desktop app or a Metro style app. It also provides an opportunity to see the dashboard of latest updates from your favorite apps without requiring you to launch them before you get into your day-to-day tasks. I know many folks have commented on not wanting to ever see such notifications or a dashboard. We would note two things.

First, even from above comments you've told us about the importance of apps that do report notifications or gadgets.

Second, given that this is a Developer Preview release, we all have to recognize that we simply don't have many Metro style apps available yet, so our natural inclination is to always go to the desktop – making it seem silly for us to start here. But once your machine is packed with apps that you love, this should make a lot more sense. And if your main goal is still to use desktop apps, you can easily do this by clicking the Desktop tile and using the taskbar, or you can customize the Start screen to put your favorite desktop apps at the beginning of the Start screen and launch them directly. It is important to keep this in mind—today you might be going to the desktop so you can immediately get to the task bar. You can always put the taskbar apps on the Start screen and launch (or switch) from there, or just put the first one you always use right there in a Fitts-friendly location. And of course we should not forget that there are substantial savings yet to be had in logging on from a lock screen (in terms of number of clicks), and so there's an immediate savings to overall workflow which fully accounts for the extra key.

How are we continuing to improve the efficiency of Start?

As we continue to build upon what we've shown in the Windows Developer Preview, we are keeping efficiency close to heart. **Based on your feedback, one of the things that we're doing to make it faster to get to All Programs is to take you directly to the Apps screen when you click Search in the desktop.** This potentially removes another step from this task, making it even more efficient in Windows 8 to launch an app from the desktop relative to Windows 7. **Another thing that we're doing is increasing the number of rows of tiles that you can see on large monitors so that you can fit even more of your favorite apps closer to your mouse and make it faster to launch apps than before.**

In conclusion, we are striving to help you gain efficiency with the new Start screen. This sort of analysis is generally difficult since we're not comparing apples to apples. In some cases, there is a loss because of mouse distance, while in other cases there is a gain because of target size. In some cases, spatial arrangement or color can make it easier to find an app, in other cases having an app right under your mouse makes it really easy to click. The efficiency gain of the Start screen may not be in all of the same ways that you're used to, and there may even be some efficiency gains that you don't expect (for example, having a live tile tell you the latest stock quote so you don't need to take time to launch the app is a great efficiency gain that is hard to measure quantitatively.) We are continually testing the efficiency of the new UI and we will continue to improve it.

If you've made it this far, you might be wondering why we put all of these issues in one really long post, and yet we still have more feedback and questions to answer. Our intention is to build on the unprecedented transparency we provide in building Windows and to bring you inside the development of the product. By now you can see that building Windows 8 is a complex endeavor with tons of variables and choices to be made, lots of data, and in considering all that, we go through a great deal of work when making even the smallest change. We simply love the dialog we're having with you, and the opportunity to describe the depth of the work we do to bring you Windows. All of us on the Windows team are devoting our professional careers to building a great product, and so the opportunity to talk with passionate and informed people about the details of what we do is itself an added bonus.

--Marina Dukhon

The Windows 8 Task Manager

Steven Sinofsky | [2011-10-13T12:00:00+00:00](#)

As we mentioned during the Windows 8 keynote at //build/, every 15 years or so we choose to update Task Manager. Of course that was said in jest as we have incrementally improved the utility in just about every release of Windows. For Windows 8, we took a new look at the tool and thought through some new scenarios and a new way of tuning the tool for "both ends of the spectrum" in terms of end-users and those that need very fine-grained control over what is going on with their PC. Ryan Haveson, the group program manager of our In Control of Your PC team, authored this post. Note: This post is about Task Manager, not about closing Metro style applications ??

--Steven

We are really excited to share some of the improvements we are making to the Task Manager in Windows 8. Task Manager is one of the most widely used apps, and it has a long history. It showed up in early versions of Windows as a simple utility to close and switch between programs, and has had functionality added to it through several releases to make it what it is today.

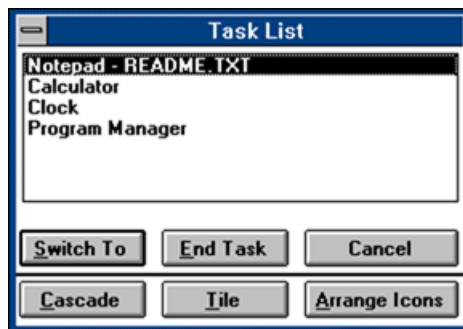


Figure 1: Windows 3.0 Task List

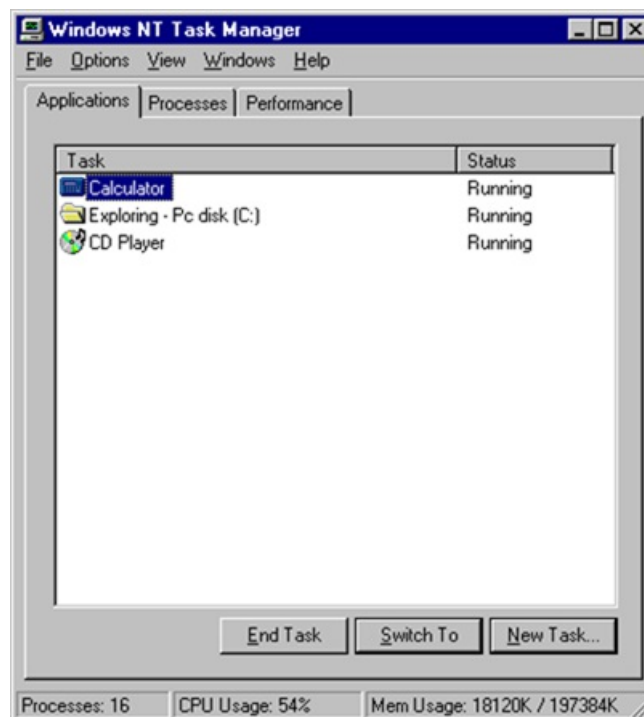


Figure 2: Windows NT 4.0 Task Manager (now with ["new task"](#))

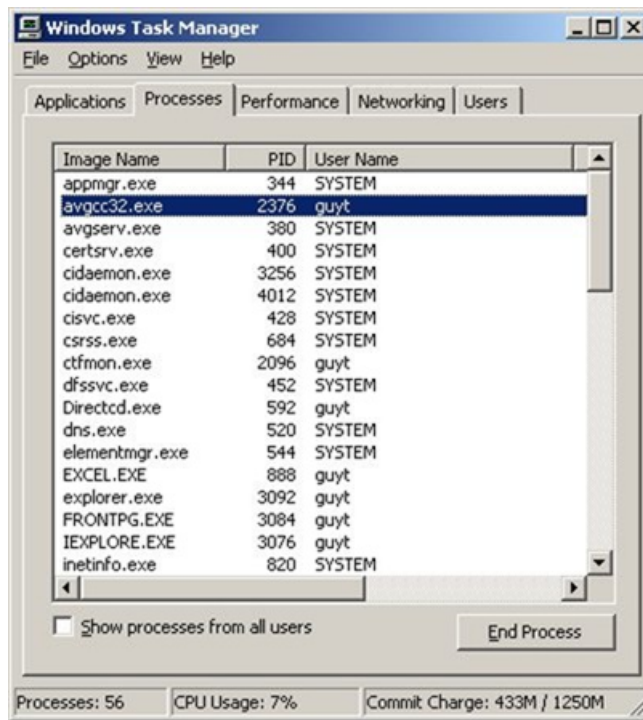


Figure 3: [Windows XP Task Manager](#) (with new Networking and Users tabs)

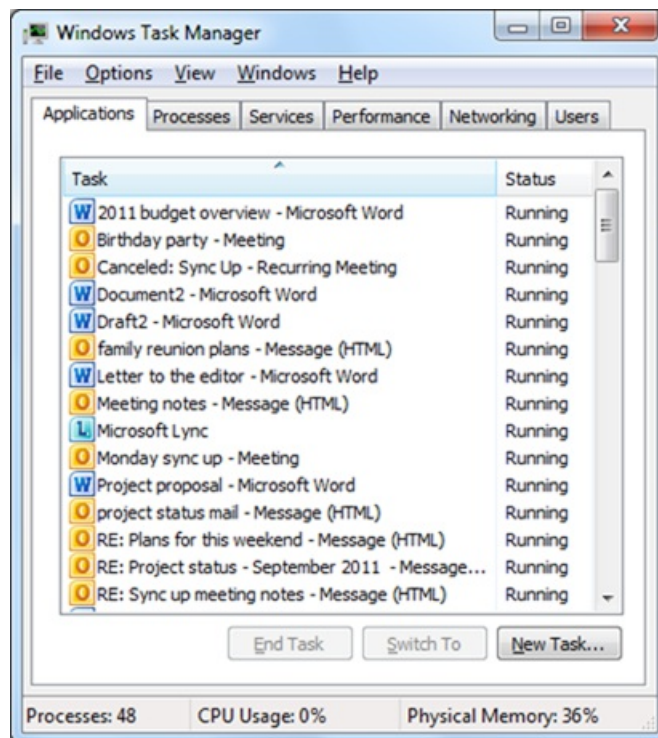


Figure 4: [Windows 7 Task Manager](#)

Because Task Manager is so widely used, we knew that any changes we made would be noticed, so of course we were both excited and cautious about the effort. At the beginning, there were a few key problems that we knew we wanted to address:

- Build a tool that was well designed, thoughtful, and modern. After all, even a technical tool can benefit from a focus on design.
- Fill some of the functionality gaps that drove some of our most technical customers to use other tools such as Resource Monitor and Process Explorer.
- Organize and highlight the richness of data available to make it more elegant and clear for those who want access to a new level of data.

How do people use Task Manager?

To really make Task Manager great at what it currently does, we wanted to first understand how people were using it. Over the years, it had grown to support many different scenarios. As of Windows 7, you could use Task Manager to close applications, to find out detailed data about your processes, to start or stop services, to monitor your network adaptor, or even to perform basic system administrator tasks for currently logged on users. That is a lot of functionality.

Because of the investments we made in [telemetry](#), we had some pretty good data to start with. We combined this with individual customer

interviews and observation in the research lab to understand what people were doing with Task Manager and why they were doing it.

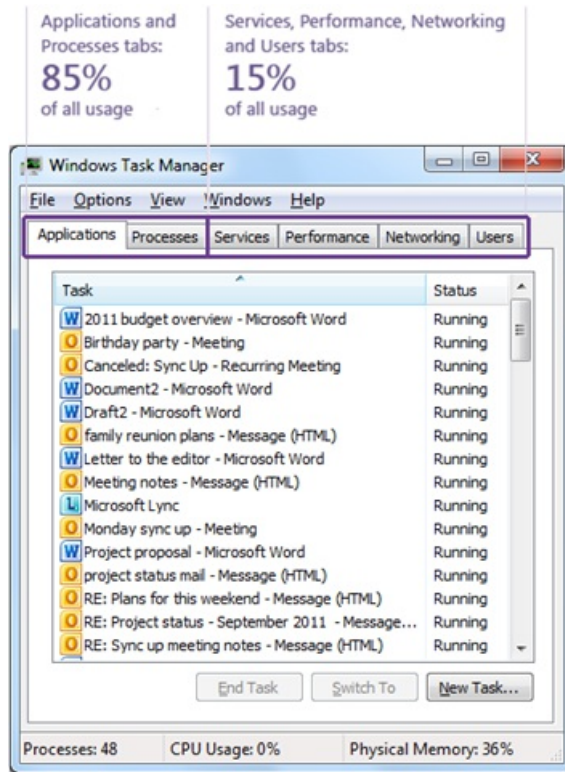


Figure 5: Which tabs are people using?

This data is pretty interesting. What it shows is that people are spending most of their time using the first two tabs, which are pivoted around views of applications and processes. Although it is not surprising, it was interesting to see that the usage was roughly evenly split between the Applications tab and the Process tab. This indicates that there must be some significant detail lacking in the Applications tab, which is causing people to go to the Process tab. So, next we looked at how people were using the Process tab to understand what they were doing there.

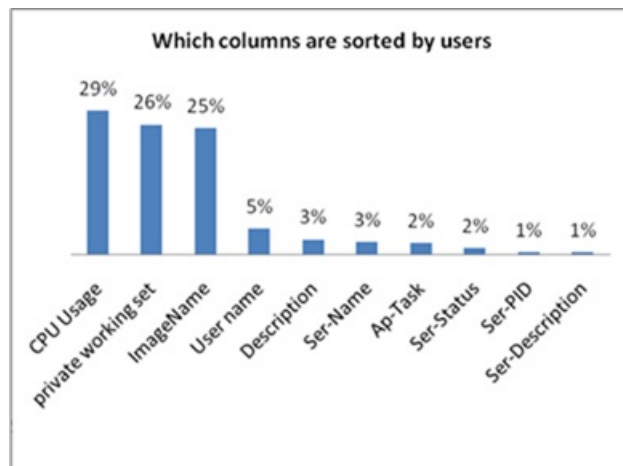


Figure 6: Many users sort the process view on resource usage

When we looked at this data, and then correlated it with interviews and observations of users in our research labs, we found that people were using the process tab either to look for something that was not on the applications list (e.g. a background or system process), or to see which processes were using the most resources.

So next we looked at what actions people take in Task Manager.

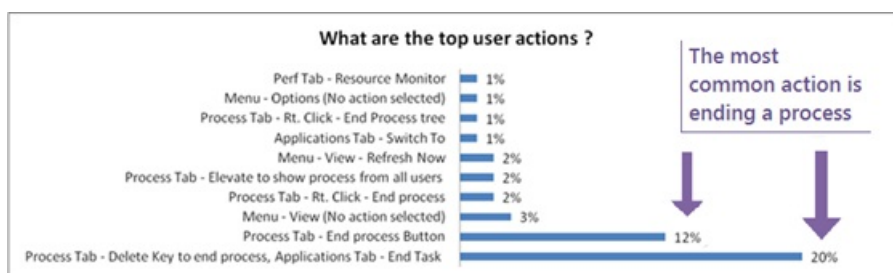


Figure 7: The goal is often to close or "kill" an app or process

Click to view a larger version of this chart

Looking at the data and talking with customers, we determined that the most common usage of the tool was to simply end or “kill” an application or a process.

Goals of the new Task Manager

Based on all of the data and our background research, we decided to focus energy on three key goals:

- **Optimize Task Manager for the most common scenarios.** Focus on the scenarios that the data points to: (1) use the applications tab to find and close a specific application, or (2) go to the processes tab, sort on resource usage, and kill some processes to reclaim resources.
- **Use modern information design to achieve functional goals.** Build a tool that is thoughtful and modern by focusing on information design and data visualization to help achieve the functional scenario goals.
- **Don't remove functionality.** While there are some notable core scenarios, there is a really long list of other, less frequent usage scenarios for Task Manager. We explicitly set a goal to not remove functionality, but rather to augment, enhance, and improve.

A key issue we intended to address was how we could add all of the interesting new functionality without overwhelming users. To solve this, we pivoted around a "More/Fewer details" button similar to the [new copy file dialog](#) model.

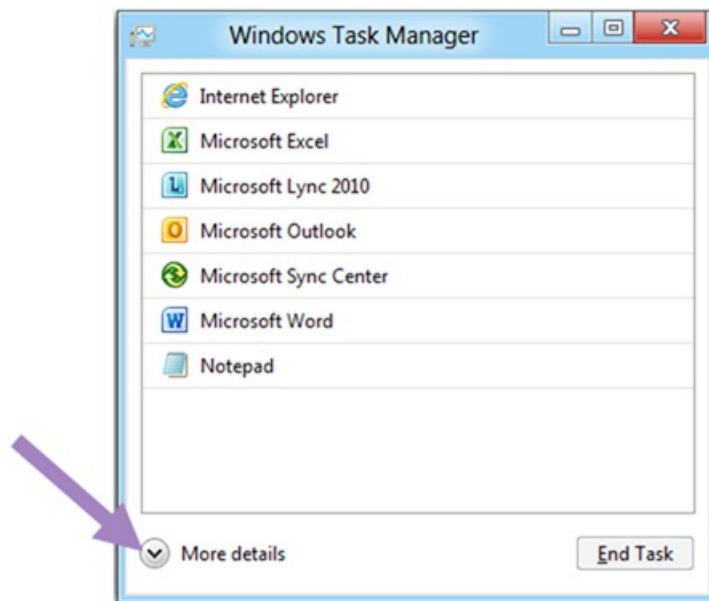


Figure 8: Fewer details view

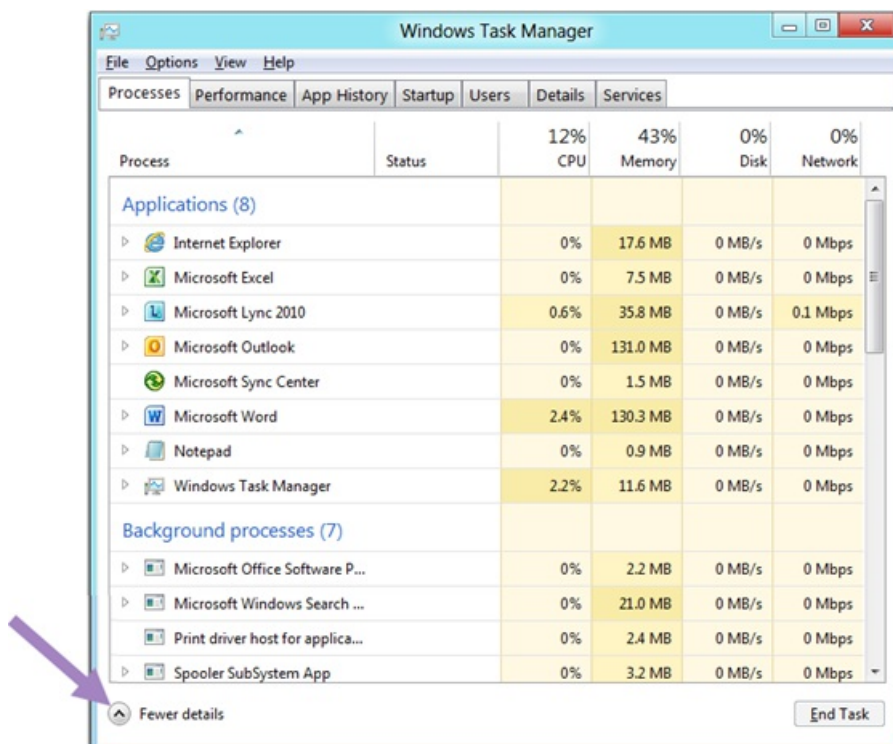


Figure 9: More details view

This model allowed us to optimize the default view (“Fewer details”) around the core scenario of finding an application and closing it. It also allowed us to add much more detail in the other view because it would only show up when someone asked for it. In the “More details” view we decided to stay with the existing tabbing model of Task Manager and focus on improving the content of each of the tabs. This would help us to augment, enhance, and improve what we already had, without removing functionality.

Scenario #1: Ending processes quickly and efficiently

We know from many third-party tools (or tools like sysinternals [Process Explorer](#)) there are many things we could add to Task Manager for power users, but we knew we had to first address the mainstream users because we didn’t want to create something that would overwhelm the majority of our customers. We will of course continue to value third-party tools as they allow for specialization and unique innovation around this and many tasks. For the default view, we designed a minimalist experience that appeals to the needs of the broadest customer base and most common scenario. When you launch Task Manager for the first time in Windows 8, you see a very clean view of your running apps. We made the default view great at one thing: killing misbehaving apps. And we removed everything that did not directly support that core scenario.

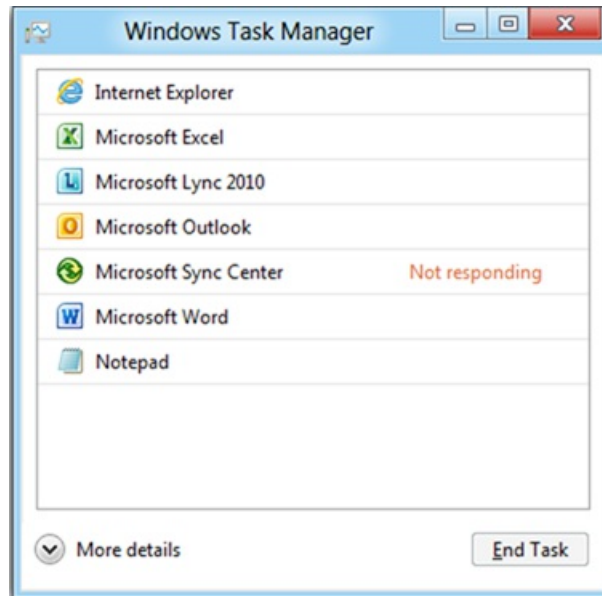
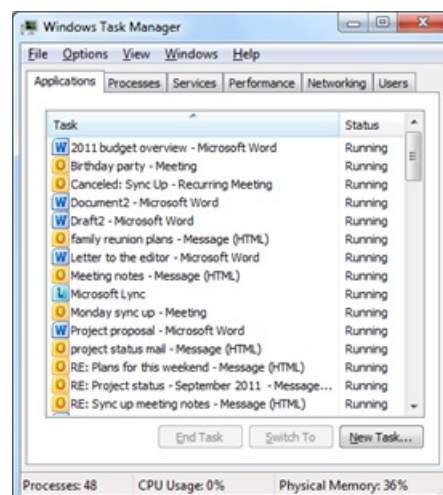


Figure 10: Default view of Task Manager in “Windows 8”

The value of the default view is all about what we took out. We removed everything not focused on the core task of killing apps, which makes the design focused and efficient. Specifically:

- We took out the tabs from this view, since they distract from the core scenario.
- We removed the menu bar from the default view.
- This view shows just the apps, and removes individual windows that can’t be killed anyway.
- We took out things that clutter the experience, such as resource usage stats and technical concepts that most users don’t understand.
- No double prompts. If you click “End task” we don’t ask you, “Are you sure?”, we just kill the app, and quickly! (But be careful, because we also won’t prompt you to save!)

Check out how much cleaner and more focused the new Task Manager is compared to the Windows 7 Task Manager with the same applications and windows opened:



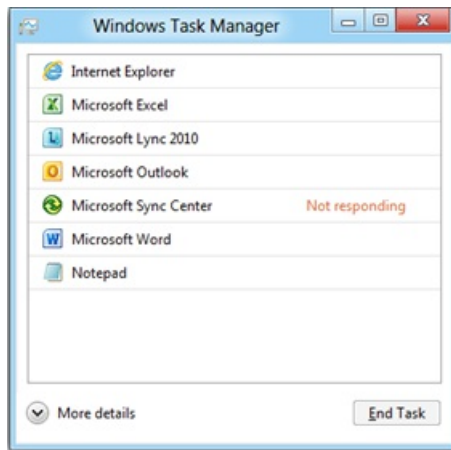


Figure 11: Windows 7 and Windows 8 Task Managers, compared

After taking out all of the extras, you are left with a tool that is great at one thing: killing a misbehaving app. And this is perfect for many users who are experiencing the pain of a “not responding” app that won’t go away using the app’s Close button.

Scenario #2: Diagnosing performance issues

A lot of what is new with Task Manager is shown only when you go to the “More details” view. This is the realm of the power user, so keep in mind that mainstream users may never want to get into this level of detail, and all of their needs should be met by the “Fewer details” view above.

Here is what you will see in this new view:

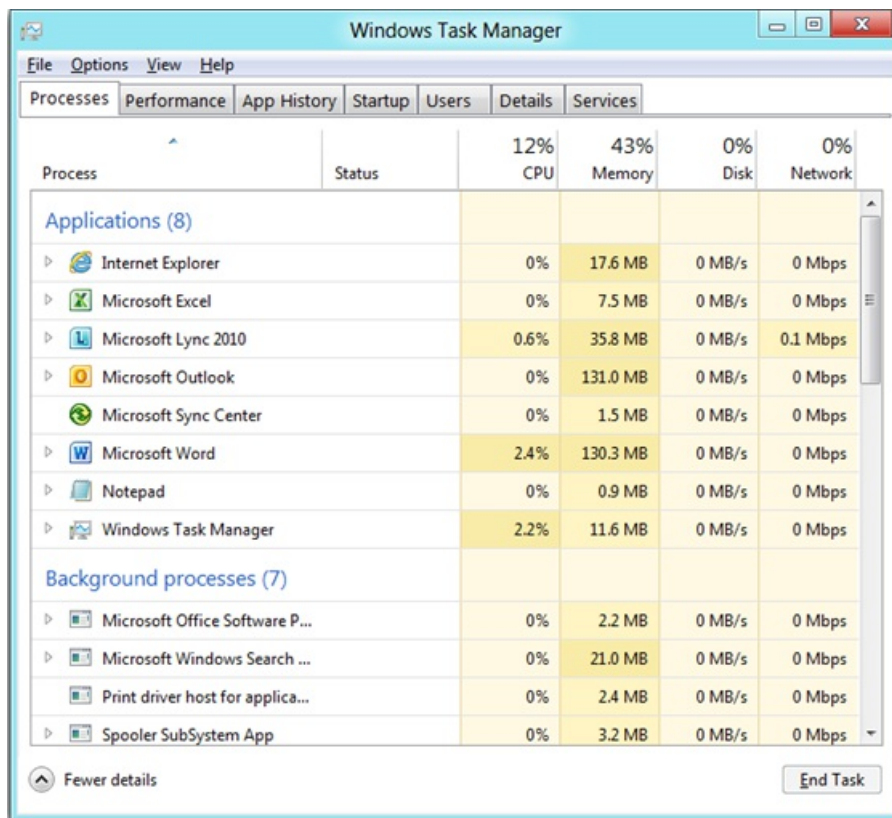


Figure 12: The new processes tab and the heat map

The heat map

The most noticeable difference in the new processes tab is the new heat map, which represents different values with color. Our telemetry data told us that it was very common for users to go to the process tab, sort by CPU or memory utilization, and then look for applications consuming more resources than expected. The nice thing about a heat map is that it allows you to monitor anomalies across multiple resources (network, disk, memory, and CPU utilization) all at the same time, without having to sort the data. It also allows you to find the hot spot instantly without needing to read numbers or understand concepts or specific units. In usability studies we used an eye-tracking system to test what users looked at when presented with various ways of visualizing this information. This helped us narrow our choices to a design that efficiently draws user’s eyes to the most significant resource problems. Below you can see the eye movement of a participant in one of our eye-tracking studies overlaid on top of a

screen shot of what he was looking at. The red dot indicates a place where his eye paused, and the lines show where his eye had quickly moved from previously.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

Network and disk counters

Many power users supplement their usage of Task Manager with other tools such as Resource Monitor simply because in the past Task Manager did not show per-process network and disk attribution. This was a gap, when you consider that a spinning disk or multiple applications competing for network bandwidth are the root cause of many perceptible PC performance issues. The new Task Manager now shows these resources at the same level of detail as memory and CPU.

Lighting up the resource usage

One of the biggest causes of PC performance issues is resource contention. When a particular resource is being used at a rate above a threshold number, the column header will light up to draw your attention to it. Think of this as a warning indicator, letting you know a good place to start looking if you are experiencing performance issues. Below, you can see that the CPU column header is highlighted to draw your attention to the fact that you may have multiple applications competing for CPU time.

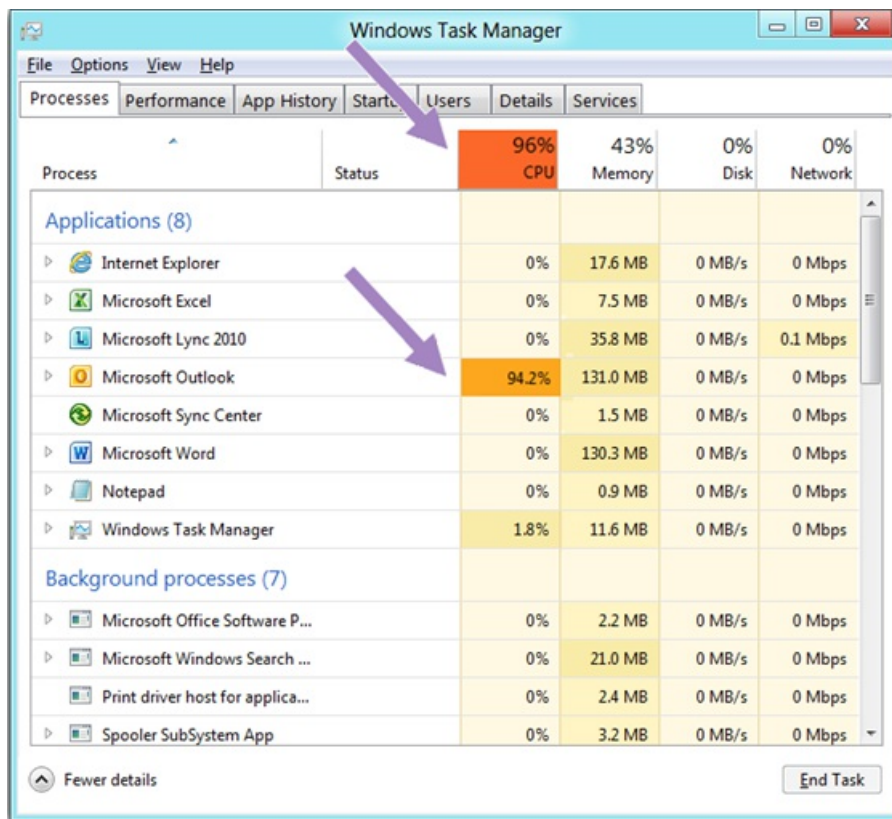


Figure 13: Resource usage indicators

Grouping by applications, background processes, and Windows processes

A big challenge with today's Task Manager is that it is hard to know which processes correspond to an application (apps are generally safe to kill), which are Windows OS processes (killing some of these can cause a blue screen), and which are miscellaneous background processes that may need to be explored more deeply. The new Task Manager shows processes grouped by type, so it is easy to keep these separated while still providing an ungrouped view for situations where you need it.

Process	Status	1% CPU	37% Memory	0% Disk	0% Network
Applications (4)					
Internet Explorer		0%	125.8 MB	0 MB/s	0 Mbps
Microsoft Word (32 bit)		0.3%	71.6 MB	0 MB/s	0 Mbps
Paint		0%	14.6 MB	0 MB/s	0 Mbps
Windows Task Manager		1.2%	13.3 MB	0 MB/s	0 Mbps
Background processes (10)					
Fast User Switching Utility Service		0%	0.5 MB	0 MB/s	0 Mbps
Media Catalog Object (32 bit)		0%	2.0 MB	0 MB/s	0 Mbps
Microsoft Windows Search Indexer		0%	49.7 MB	0 MB/s	0 Mbps
Print driver host for applications		0%	1.1 MB	0 MB/s	0 Mbps
SMSvcHost.exe		0%	0.9 MB	0 MB/s	0 Mbps
SMSvcHost.exe		0%	1.2 MB	0 MB/s	0 Mbps
SMSvcHost.exe		0%	1.0 MB	0 MB/s	0 Mbps
Spooler SubSystem App		0%	2.5 MB	0 MB/s	0 Mbps
Windows processes (28)					
Client Server Runtime Process		0%	8.6 MB	0 MB/s	0 Mbps
Client Server Runtime Process		0%	1.0 MB	0 MB/s	0 Mbps
Desktop Window Manager		0%	25.5 MB	0 MB/s	0 Mbps

Figure 14: Grouping by process type

Friendly names for background processes (and services, and everything else)

Looking at the screen shot above, do you see the line item for **"Print driver host for applications"**? In the old Task Manager, this showed up as **"splwow64.exe"**.

But if you still want to see the executable name, of course you can add it back as an optional column.

Grouping top-level windows by app

One of the most distracting parts of the old Task Manager is that the Applications tab was a flat list that included all of the top-level windows from all processes in the system. While the list of top-level windows is interesting information to have, it is often overwhelming to look at and sometimes a single window cannot be killed without closing all the other windows for that process. To address this, the new Task Manager now groups top-level windows under their parent process. It allows for a much cleaner view for typical usage, helps you focus on killable processes, process resource usage, and allows you to see which windows are owned by each process so you know what will be closed if you kill it.

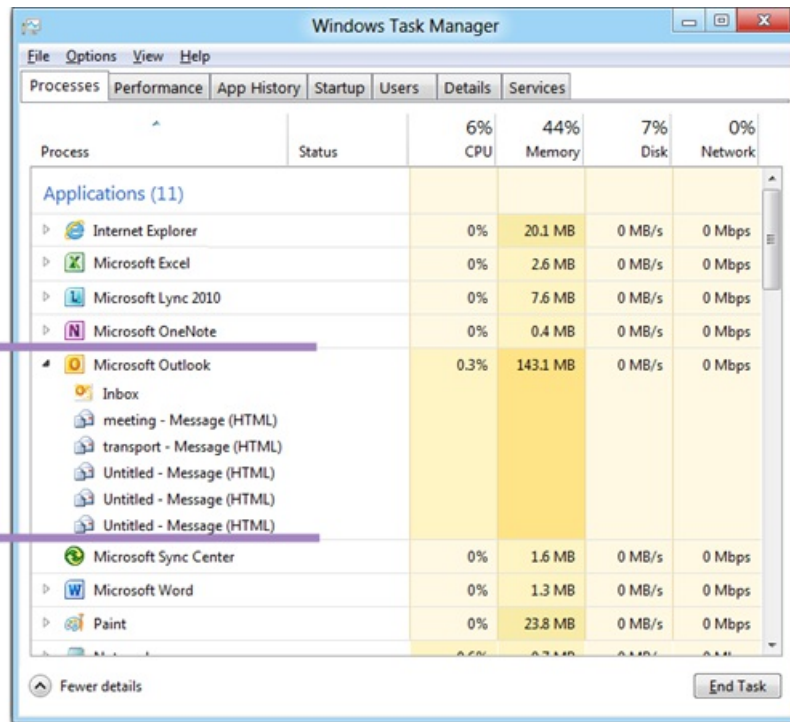


Figure 15: Grouping top-level windows by process

What's a fussvc.exe?

Have you ever looked through the process list, seen something like “fussvc.exe” and wondered what it was? Adding friendly names was a good first step to resolving this problem (fussvc.exe is actually the Fast User Switching Utility Service), but of course, to really find out what this process is, you need to search the web. The new Task Manager integrates a search context menu on right-click, so you can go directly to your default search engine (which you can customize) to see more details and relevant information. This can make a huge difference when deciding whether a background process is doing something useful or just wasting cycles.

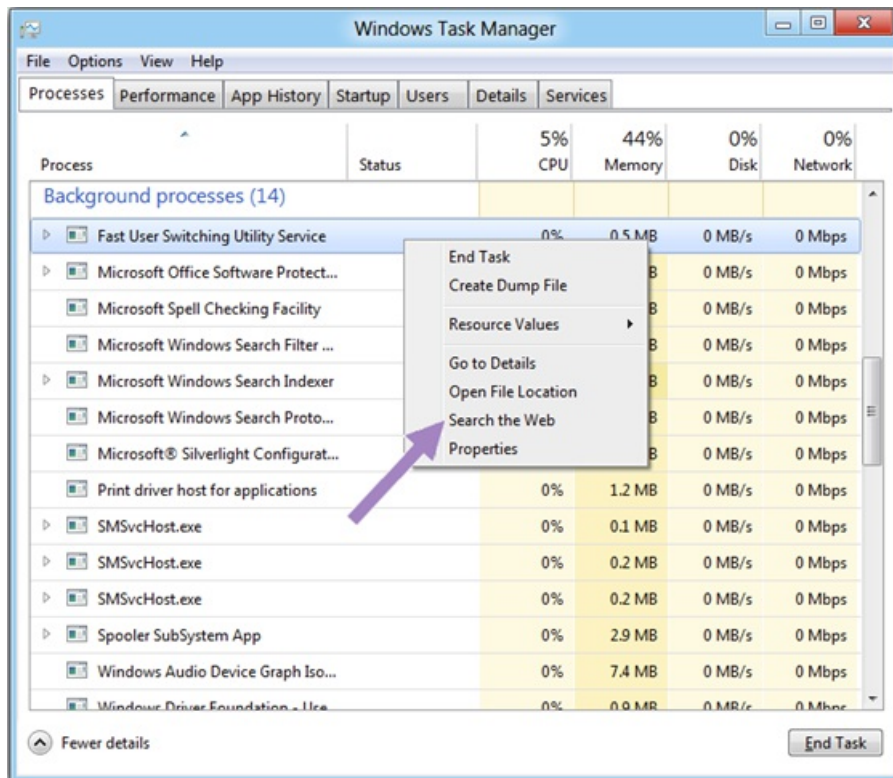


Figure 16: Search the web for details on obscure processes

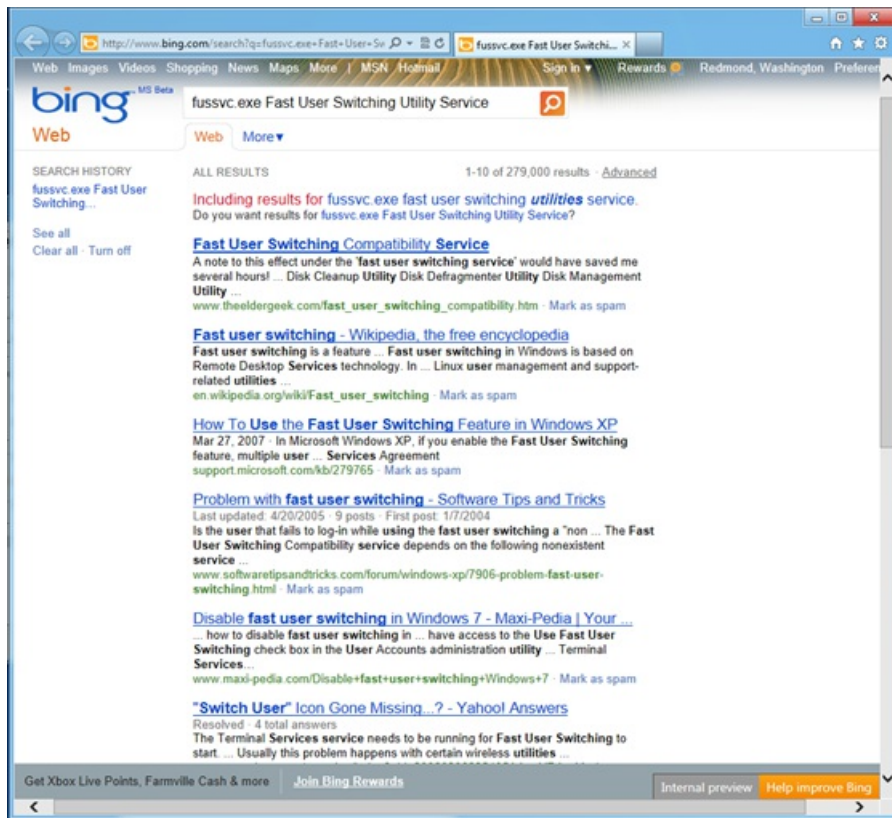


Figure 17: Search results for "fussvc.exe Fast User Switching Utility Service"

Service host details and friendly names

If you open up Windows 7 Task Manager to the Processes tab and select "Show process from all users," you will probably see eight seemingly identical instances of "svchost.exe". This is one of the most commonly noted "not very informative" sources of information we provided. Of course, some of you know that this is really just a service host process and you can add the PID column, go to the services tab, sort by PID, see which services correlate to that PID, and then reverse-look-up friendly names for each of the services... but that is a lot of work (and not everybody knows this)! With the new Task Manager, we show all of the services grouped by process with friendly names for each of them, so you instantly can see what is going on when an instance of svchost is consuming a lot of resources:

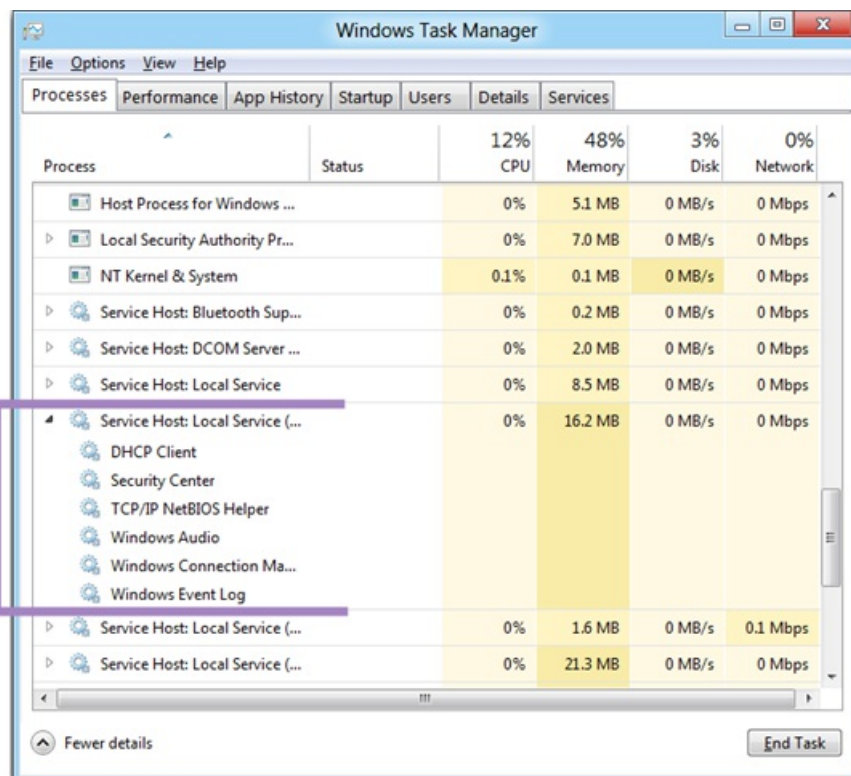


Figure 18: Service host grouping and details

As you can see, we added quite a lot to the new Task Manager (and we only showed you the first tab!). Task Manager was a unique opportunity for user experience designers and researchers working together with technical program managers and engineers to create a clean, organized, and

efficient design. We made it more streamlined for mainstream users, and more detailed for power users.

I will leave you with a quick demo where you can see what it looks like in action.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

-- Ryan

Designing search for the Start screen

Steven Sinofsky | [2011-10-18T12:00:00+00:00](#)

Given the ton of interest in the design of the new Start screen we wanted to dive deeper into the topic of search. There's a clear focus on efficiency and overall professional productivity in the comments. For professional scenarios, every keystroke matters. One new aspect of the Windows 8 platform is the ability for Metro style apps to deliver a customized search "contract." For this post we'll focus on the built-in search capabilities for files, settings, and apps, which update the Windows 7 search features. You can learn more from our [//build/ session on search](#), which provides a detailed look at the topic of this post. With that lens, Brian Uphoff, a program manager on our Search, View, and Command user experience team, authored this post.

--Steven

In our previous related posts ([Evolving the Start menu](#), [Designing the Start screen](#), and [Reflecting on your comments on the Start screen](#)) we discussed the evolution of the Start menu and the reasoning behind the design. We also discussed how organizational mechanisms and search are powerful tools that make it easier to find and launch apps. As you install more and more apps, these tools become increasingly important. For the past several releases, searching from the Start menu has been established as the quickest way to find and launch apps, particularly for keyboard users.

When planning Windows 8, we wanted to make sure the efficiency and dexterity of the Windows 7 Start menu search was carried forward into the new Start screen. Before we dive into the details of the new experience, let's take a quick look at the evolution of search from the Start menu, and how people are using it today.

Evolution of searching from Start

The search box in the Start menu as we know it today first made its appearance in Windows Vista. It became easy for users to search for programs or apps, settings, and files on the desktop and in personal folders like Documents, Pictures, Music, and Videos. The search experience aggregated different types of results in one view with programs and settings combined in a single group. The results of a query displayed a small set of items in heuristically sized groups. You needed to click "See all results" to see the rest in Windows Explorer, which aggregated everything into one ungrouped and unsorted view.

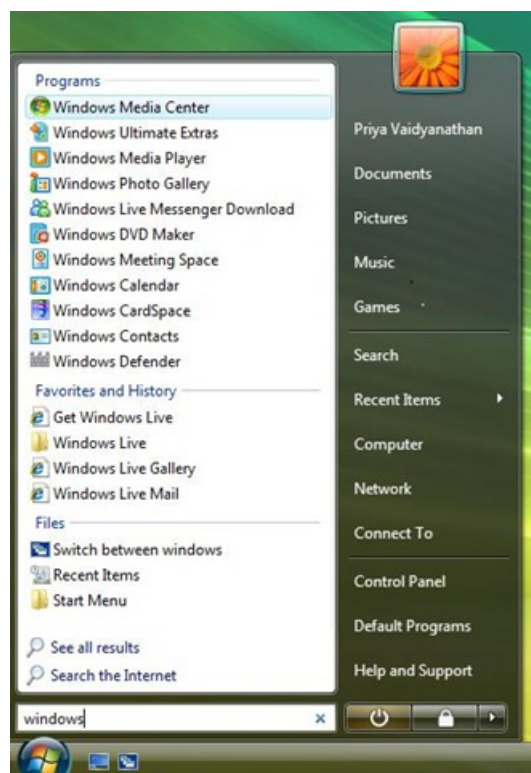


Figure 1: Start menu search in Windows Vista

In Windows 7, we expanded results to include detailed Control Panel tasks in addition to the main Control Panel pages. We also separated out Control Panel items from programs into a unique group that allowed you to more easily focus on the type of result you were looking for.

The overall experience aggregated different types of items and had a fixed limit on the number of results that could appear. This was because the result set was limited to the size of the Start menu. Clicking a group header took you to Windows Explorer for programs and files or to Control Panel for settings. Each experience had a type-specific view, though the search results order diverged from what was shown in the Start menu. Showing an aggregated view in the Start menu required compromising on performance in addition to space because we would search all programs,

Control Panel items, and files, even if you were looking for only one of these data types.

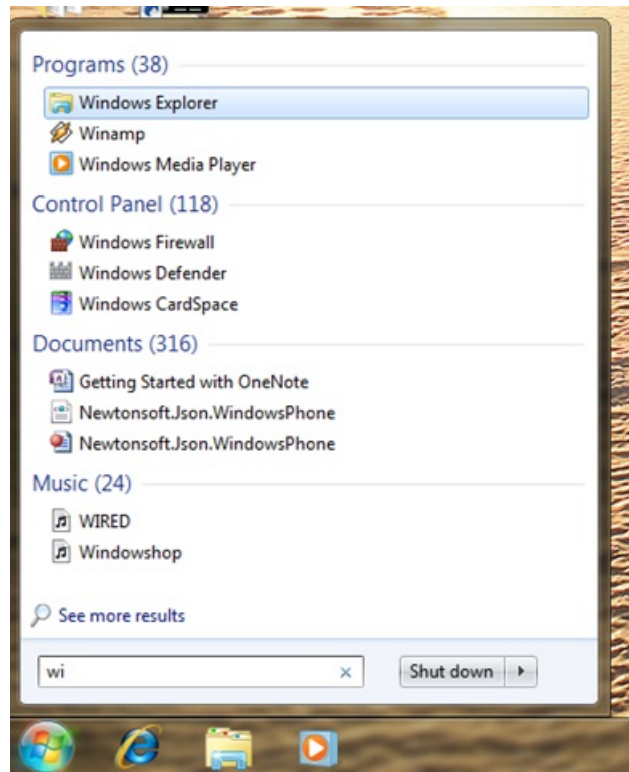


Figure 2: Start menu search in Windows 7

When we look at the usage data of how people are using the Start menu to search in Windows 7, it's clear that searching to launch programs is the most frequent and important activity users engage in with Start search.

Our telemetry data shows that 67% of all searches in Windows 7 are used to find and launch programs. Searching for files accounts for 22% of all Windows 7 Start menu searches, and searching for Control Panel items about 9%. Searching for email messages via Start Menu is very rare (less than 0.05%). The remaining 2% are searches executing the "Run" functionality.

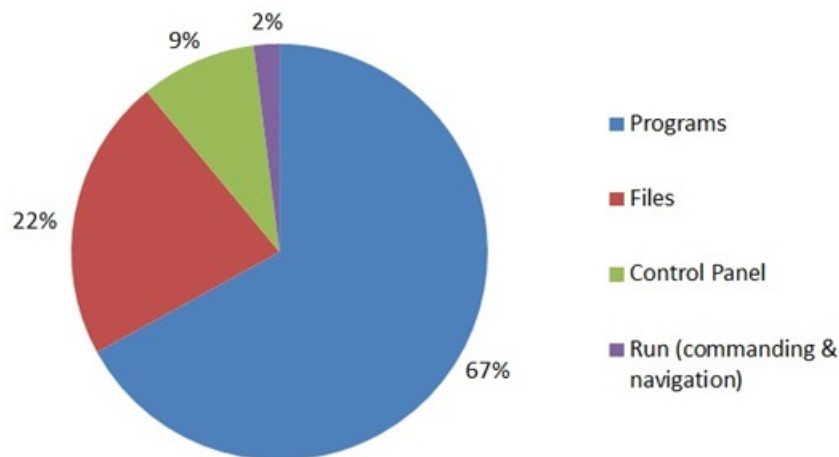


Figure 3: Windows 7 Start menu search usage data

Searching from Start in Windows 8

Searching via the Start menu has continued to evolve with each release. The Windows 8 Start search experience builds on top of search features available in Windows 7 and provides a unique view for each of the three system groups - Apps, Settings and Files. These search result views are a natural progression from the Windows 7 groups and are easily accessible from anywhere in the operating system via the Search charm or keyboard shortcuts. Separating the search results into views means we can tailor the experience for each data type. For example, the File search view provides you with filters and search suggestions while typing to quickly complete your query.

In Windows 8, we expect people will be acquiring and installing more apps than ever before. Had we continued using the Windows 7 Start menu search interface to search for a Control Panel item, you would always see app or program results before Control Panel results, displacing many Control Panel items from being the first match. This and other constraints on the existing design required us to develop a new approach—this is especially true as we consider the increasing use of larger monitors or higher DPI screens where longer menus become even more difficult to use

and navigate. In Windows 7, the total number of results that could be shown in the Start menu was limited. Depending on the number of groups with matching results, an average of 3-4 results were shown per group. Very rarely did all results for a group show up, and the organization of the results was pretty unpredictable.

With Windows 8, on the other hand, we're following an app-first model, where each app developer understands their data and users best, and knows the best way to present the information to them. Using the same model for search, we believe that always having a quick and consistent way to get directly to settings or file search results gives you precision and control over the type of results you're looking for. In Windows 8, each view is tailored for the type of content you're searching for, and shows all the results, instead of limiting them due to screen real-estate.

One change a few of you will notice is that file search results no longer include email messages and contacts. The inclusion of email search never got the generalized support from mail clients that we had hoped for, though at least one mail client did support it (one reason why email searches are rare in the Start menu <0.05% of total searches). With the app-first approach in Windows 8, Metro style email apps will use the search contract to provide a rich set of filtered search results in a view customized for email. In comparison, email clients and other apps in Windows 7 have no control over how their search results are presented.

We paid special attention to ensuring the number of keystrokes required to find and launch apps, settings, or files is at parity with or better than in Windows 7. We've introduced a set of **keyboard shortcuts** to help users quickly and efficiently get to settings search results (**WIN key + W**) or file search results (**WIN key + F**), thus reducing the total number of keystrokes needed to find and launch settings or files. We'll cover how we maintained and increased keyboard efficiency across these views in more detail later in the post.

Searching apps

App search results show the full set of apps (both their "friendly" names and executable names) for which the search term matches the name. As the number of installed apps increases, it becomes difficult to browse through a large list to find an infrequently used app. Search helps quickly filter and reduce a large list of apps down in just a few key strokes. We wanted to make sure we preserved the same keyboard usage patterns as Windows 7. You don't have to first click on the Search charm to begin searching – simply start typing in the Start screen and you'll see your list of apps filter down to the one you are looking for.

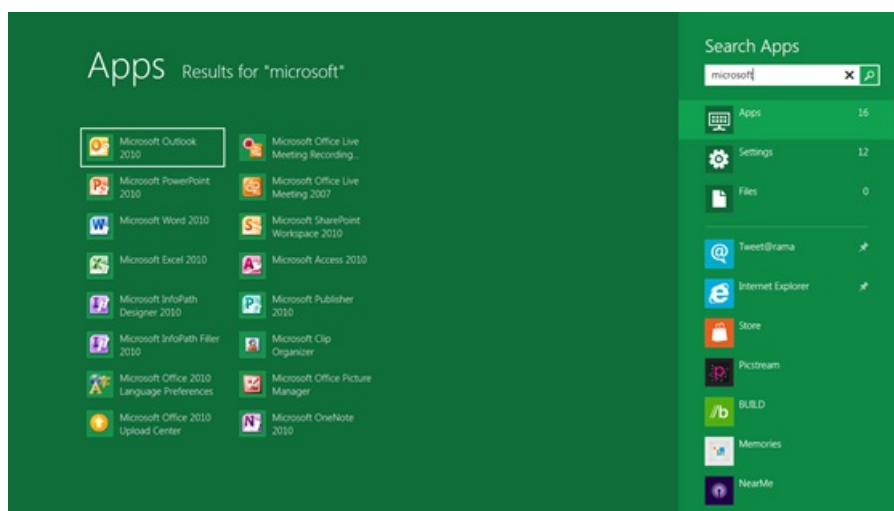


Figure 4: Full-screen app search results

Also note that the Most Frequently Used (MFU)-based ranking of app search results from Windows 7 is preserved in Windows 8. For example, if you type "paint" in the developer preview you get 2 apps back as search results – PaintPlay and Paint. If you predominantly just use Paint, it will be ranked higher than PaintPlay as you use it more often. So, launching Paint (or other apps you frequently use) becomes more efficient the more you use app search.

Some of you have pointed out that many users won't discover that they can simply type to start searching in the Start screen. Search is closely associated with typing—the most common pattern to search in the Start menu is to bring up the Start menu by using the Windows key or by clicking the Start button and typing. That exact and efficient behavior is preserved in Windows 8 as we have observed and found that pattern is what users care about most. Our experience in user tests, and even when people at //build/ tried the Develop Preview for the first time, shows that people tend to serendipitously discover this feature early in using Windows 8, and so we're confident it will not be a hindrance to usability. Nevertheless the Search charm is highly visible, and selecting it shows the Search box.

The Windows 7 Start menu also included "Run" functionality for commanding and navigating Windows. This has been carried over to Windows 8 as well—tasks like running scripts and .exes in the user's PATH are still possible and supported in App search. Search continues to support launching folders in Windows Explorer by typing in full paths. For example, typing "C:\" in Start search results in the set of folders in the C: drive appearing below the search box. Pressing the Down Arrow key moves selection through the list and autocompletes the folder name in the search box, allowing users to continue typing to further refine the path. You can do the same with UNC ([\\foo\example](#)) paths as well. And of course **WIN key + R** will switch to the desktop and bring up the classic Run dialog, just as you would expect.

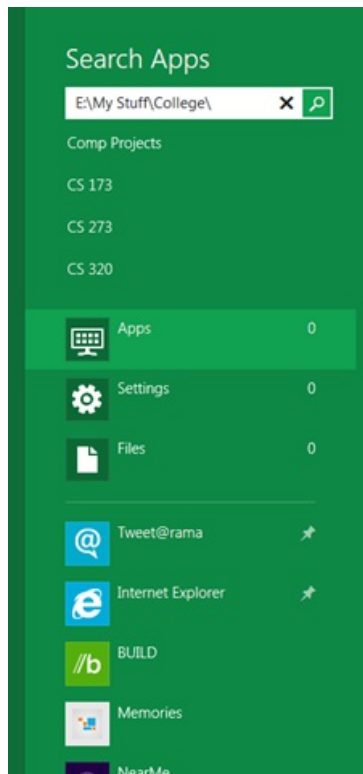


Figure 5: Typing a path into Start search

Searching settings

The settings search experience brings together all settings and Control Panel items across the system in one view. Settings search results are matched not only to the name of the Control Panel applet or task, but also to the various keywords that may describe it. We have also heard your frustration that shutdown is not available as a search result, and we will address this along with improvements to the Start user interface for shutdown (as a reminder, you can also just use the power button or close the lid).

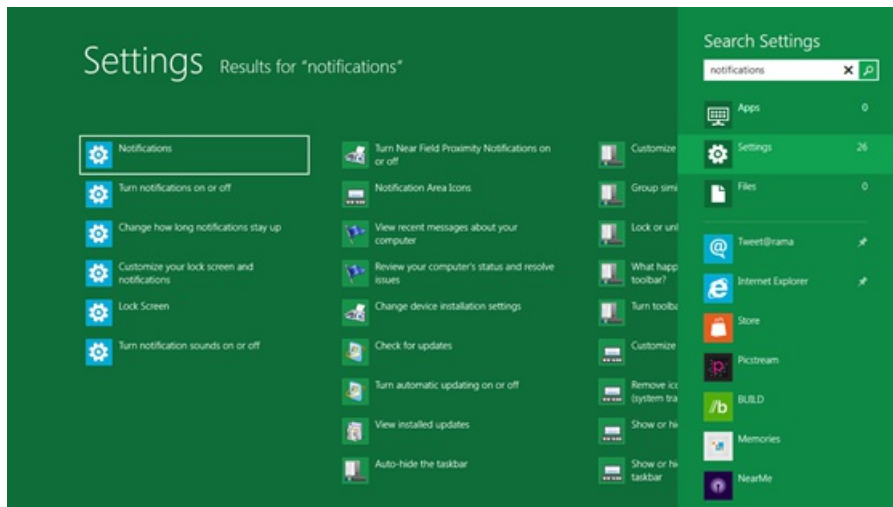


Figure 6: Full-screen settings search results

Searching files

The number of files on PCs keeps increasing over time as users continue to acquire and create more documents, music, photos, and videos. Our goal, while redesigning the file search experience, was to make it seamless and complete so you can achieve your task of quickly finding a file without having to transition to Windows Explorer.

In File search, you'll also see search suggestions as you type to help you quickly and efficiently complete the search. The indexer provides these search suggestions based on the content and properties of files it knows about. Search suggestions are a very powerful concept made popular and used extensively on the web—they help you to pinpoint relevant search terms with just a few keystrokes. In Windows 8 we built search suggestions into the file search experience and also made this feature available in the platform for all Metro style apps to use. Note, this feature also accounts for typos or spelling errors, and suggests the auto-corrected search term as you type. Using the arrow keys to choose suggestions autocompletes the term in the box. This makes it easy to add more terms to the query and quickly narrow down the set of results to find the one you want.

You can also still search using AQS ([Advanced Query Syntax](#)) from Windows 7. AQS allows for greater precision and control when constructing

the query to get targeted results. Here are some sample searches and their advanced query syntax:

Query	AQS Syntax
Find all files authored by Brian or David	author:(Brian OR David)
Find all photos with an F-stop of 2.8 where no flash was fired	f-stop:2.8 flashmode:no flash
Find all files where the file name contains a word starting with Metro and the file size is greater than 1MB	filename:\$<Metro* size:>1 mb

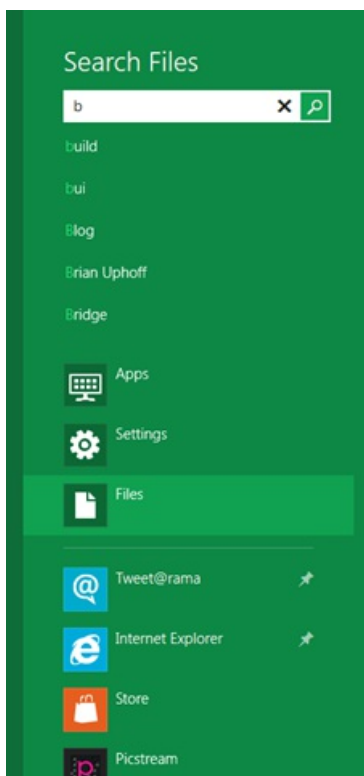


Figure 7: Search suggestions based on the content and properties of files

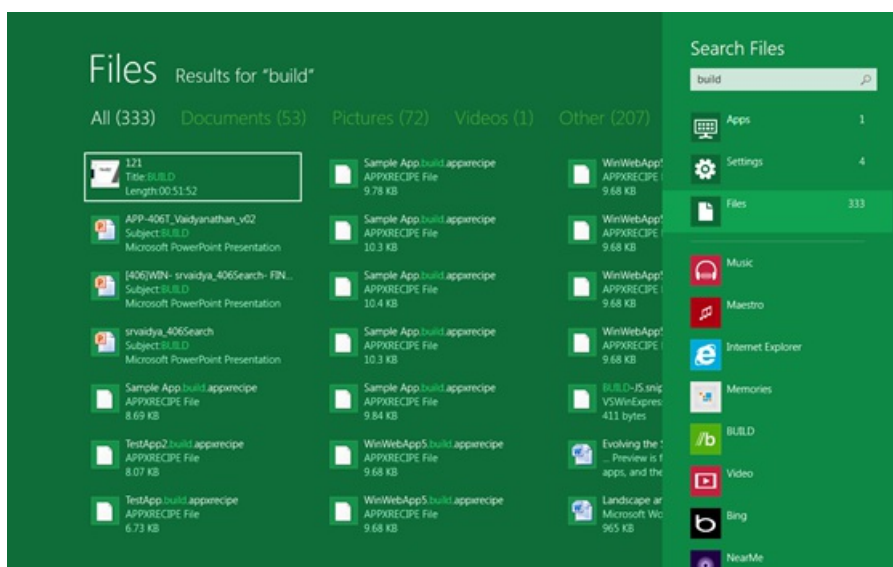


Figure 8: Full-screen file search showing results

Separating searches for apps, settings, and files into their own views allows room for each of them to evolve and breathe— this way they can each provide their own ideal display format—unlike the single list of results in previous versions, which required conformity to achieve aggregation in the

limited space. For example, the file search view also provides filters to easily refine the results based on the type of file you're searching for. Filtering by type is a powerful way to efficiently reduce the results set, irrespective of where the file is saved.

More relevant and contextual information for each file is also now displayed to make the search experience complete. This helps differentiate between similar results and also makes it clear to you why a given result was returned, by highlighting the property that matched the search term—something not possible in the Start menu before. For example, when searching for the term “performance,” the results now highlight where “performance” matched. In one result, it matched on the Title property, and is clearly indicated that way in the result. The results also show file type and size to help further disambiguate between results.

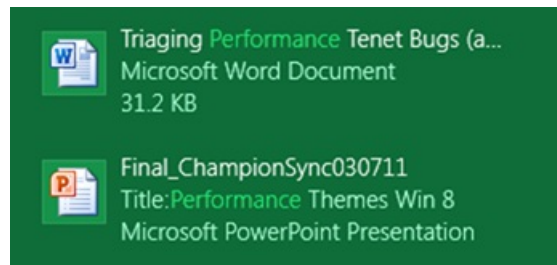


Figure 9 File search results highlight the property that matches the search term

Using a mouse, hovering over a result reveals a rich tooltip with some additional details. For example, for the video result shown below, the rich tooltip shows the duration of the video, frame height, frame width, date modified, and the full path to the file. Using touch, pressing and holding on an item reveals the tooltip.

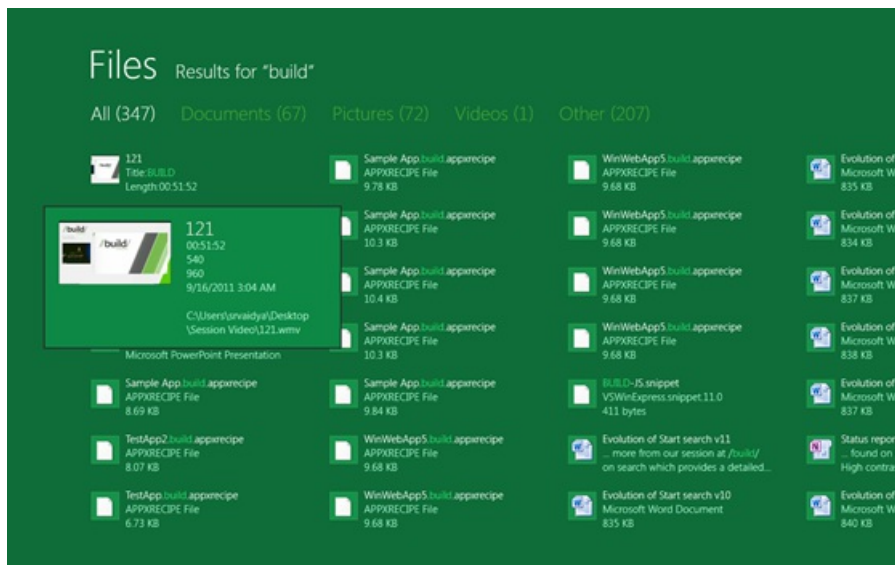


Figure 10: Rich tooltip reveals additional details in file search

Designing Start search for dexterity

Designing for efficiency and dexterity is a core goal of the Start search feature team. As such, using the keyboard to launch apps, settings, and files from search is a very important part of the Start search experience. We also put a lot of thought into preserving existing keyboard patterns, which both average and advanced users have come to rely on, and have built muscle memory around.

Our telemetry data shows that many users leverage the Start menu as a means of commanding Windows. They use specific key-combinations to efficiently launch apps. For example, pressing the WIN key, typing “calc”, and pressing ENTER launches Calculator. Many advanced users know that typing “cmd” and then CTRL+SHIFT+ENTER opens an elevated command, and that typing “notepad c:\mynotes” creates or opens a .txt file. If you watch the keyboard demonstration from the [//build/ keynote](#), you will see many of these used.

These keyboard patterns continue to work in Windows 8 just as well as in previous versions. Pressing the WIN key takes you to the Start screen. Simply start typing in the Start screen and the Search pane automatically opens with the search term in the search box, and the view filtered to show apps that match the term.

The fastest way to search settings and files from anywhere in the system is to use a set of keyboard shortcuts introduced to increase efficiency. These Windows 8 shortcuts reduce the number of keystrokes needed to launch a setting or file to a number equal to (or less than in many cases) what was required in Windows 7. Alternatively, you can also use the Search pane, which indicates the number of results matching the search in each view, to switch between apps, settings, and file searches.



Apps search



Settings search



Files search

Figure 11: Windows 8 Start search keyboard shortcuts

Based in part on the feedback here, we are working on a change that we hope to have available in our beta release, which will take you directly to app search results when you select the Search charm in the desktop.

The efficiency of using the keyboard doesn't stop at just typing to start a search. Sometimes the app, setting, or file that you want to launch is not the first result shown. You can use the arrow keys to quickly move down to the desired app in the results list, and then press ENTER to launch it. The white box that shows keyboard focus tells you which app will be launched on ENTER. This enables you to efficiently launch any app, setting, or file matching a search. In Windows 7, you could only launch one of the top 3-4 results with this kind of efficiency.

When we looked at some of the common Control Panel items people were searching for (for example, searching for power options using the term "power"), it quickly became clear that because we favored app results first in our Windows 7 design, "Power options" was the fourth result, below all the power shell app results. If you installed Office and frequently used PowerPoint, you saw PowerPoint along with Power shell (32 bit, 64 bit, and the Help file) ahead of "Power options." Extending a similar design in Windows 8 would have meant that the position of the "Power options" result would continue to fluctuate as you installed more apps on your system. This forces you to scan through increasingly large result-sets every time you search for a particular app or setting or file.

In Windows 8, we also provide the count of results per system view, so you immediately know how many apps or settings or files match the search term. Switching search views is also designed so you can easily switch views without taking your hands off of the keyboard. In the example shown below, to switch to settings search, press the Down Arrow key and the focus shifts to settings in the search list. Press ENTER and you will see settings search results. As mentioned before, you can continue to use the arrow keys to choose the desired item and press ENTER to launch it. Pressing TAB allows you to quickly switch between the search results list and the Search pane.

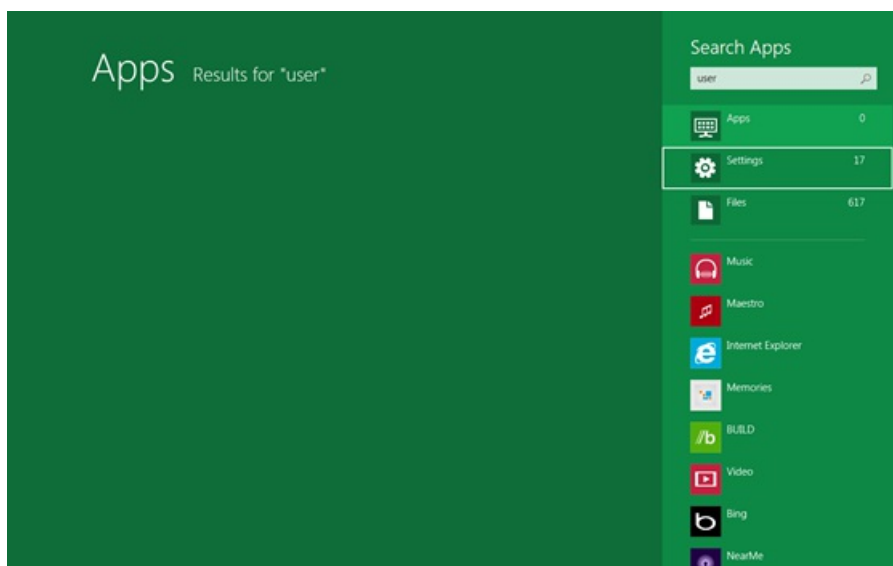


Figure 12: No results for apps – but use the arrow keys to switch to settings view, which shows 17 results

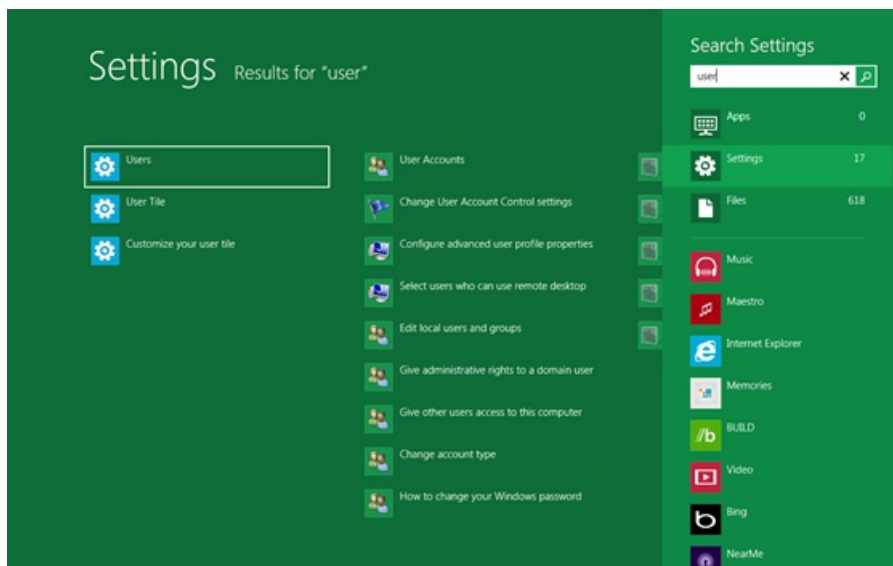


Figure 13: Use the arrow keys to choose a settings search result

To add to our earlier point on preserving search efficiency, here are some comparisons of the number of keystrokes for launching frequently used apps via search. In Windows 7, you would press the WIN key, start typing in your search term, and then press ENTER to launch the program. We count all the keystrokes end to end. In Windows 8, you can apply the same pattern for searching for apps (WIN key, type in the search term, press ENTER to launch). Launching Word, Calculator, Paint, or Media Player by pressing the WIN key and typing "word", "calc", "calculator", "paint", "player", or "media" in the search box takes precisely the same number of keystrokes in Windows 8 as it does in Windows 7.

To launch settings in Windows 7, you would press the WIN key, type in the query, arrow down to the result you wanted, and press ENTER to launch it. In Windows 8, you can use WIN key + W to launch settings search, type in a query, and press ENTER to launch. Typing WIN key + W and typing "uninstall", "device manager", or "defender" gives you the same results with precisely the same number of keystrokes in Windows 8 as in Windows 7. In some cases, it takes even fewer keystrokes than in Windows 7 (for example, pressing WIN key + W, typing "power" and then pressing ENTER to launch power options).

This dexterity-focused design isn't all we have done to make search more efficient. We have also made key performance investments across the system. In current testing of Windows 8, our search performance improvements have cut app search time in half for desktops and laptops. The improvements are even larger on netbooks.

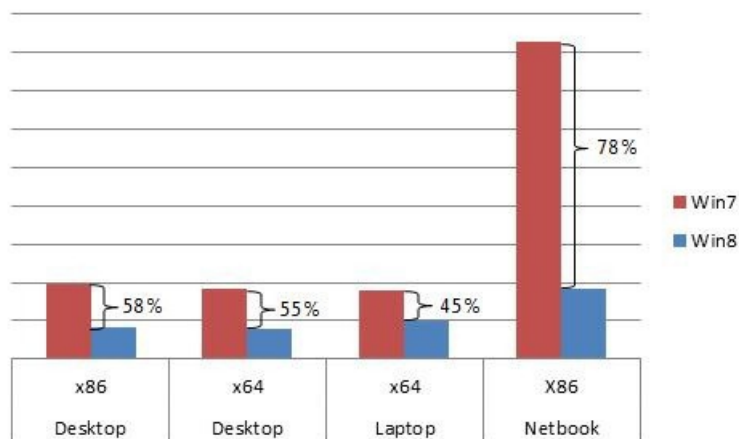


Figure 14: Performance comparison showing % decrease in execution time of app search

Designed for touch, too

We discussed the details of designing search for dexterity while using the keyboard, but this design works equally well for touch. To begin searching in Start, simply swipe the edge and tap on the Search charm. This opens the full list of installed apps. You can use the touch keyboard to search for a program to launch, but you can also use [semantic zoom](#) to zoom out, and then tap on the section that contains your app. Start search is lightweight, fast, fluid, and quickly gets out of the way as you pan through your list of apps, settings, or files.

The Search pane makes it easy to continue searching for the same term in other system views or Metro style apps with just one tap. Touch-friendly search suggestions minimize typing on a touch screen, and the search contract provides a framework for search suggestions that developers can use for their own Metro style apps as well. In addition, we designed touch-friendly filters with result counts in the file search view to help users quickly refine the search results set.

The new Start search experience makes it easier than ever to search for content in your PC or in apps from anywhere in the system. It's been designed to work seamlessly and efficiently across the range of devices that Windows will run on, and across different input mechanisms such as

the mouse, keyboard, and touch. Start search brings apps, settings, and files together with other Metro style apps that implement the Search contract, creating a unified and consistent search experience. We will talk more about the search experience and the search contract in a future post. In the meantime, you can get more information from our talk at //build/ on [the Search contract](#).

Here's a video showing how easy and efficient it is to quickly launch apps, settings, and files from anywhere in the system using the keyboard:

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

We look forward to your continued feedback as you try out the Windows 8 Start search experience!

Thanks,
Brian

Optimizing for both landscape and portrait

Steven Sinofsky | [2011-10-20T12:00:00+00:00](#)

As we have demonstrated Windows 8 in many forums, we've tended to use landscape orientation (widescreen) quite a bit. Primarily that's because often we're projecting, and it makes for a better experience that way. Another reason is that many of the early devices (such as the Samsung tablet issued at //build/ with Windows Developer Preview) are widescreen, which is ideal for showing side-by-side applications using the new Snap feature, and that tends to work well in landscape. We have done a ton of work to enable a fast and fluid experience in rotating the screen, and a great experience for people who prefer the portrait orientation, and as you will read, this is heavily influenced by our experience studying what factors contribute to a preference in either orientation. We even did work on our Visual Studio and Expressions tools to make sure developers have great tools support for building applications that work well in both orientations. David Washington on our user experience team authored this post on landscape and portrait screens in Windows 8. He also offers session [APP-207T](#) from //build/. --Steven

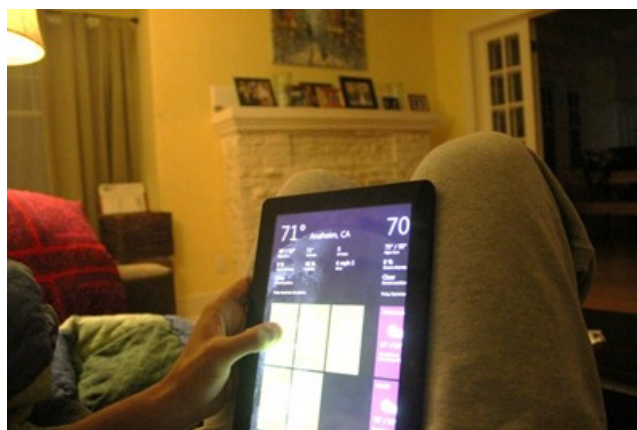
A Windows 8 PC is really a new kind of device, one that scales from small, touch-only tablets to laptops and desktops. While reimagining Windows 8, we designed it to deliver a great experience regardless of the form factor or screen orientation. Tablet devices allow for ergonomic flexibility, allowing you to hold the device in whichever orientation is most comfortable to you and best suits your content.

One of the best things about a tablet is that you can hold it in your hands. It's personal. Whether you're reading the Sunday newspaper or browsing through a stack of wedding photos, being able to hold and touch what you interact with ties you emotionally to it. In the digital age, a lot of what is most important is on devices, so when planning Windows 8 we wanted to make sure that the experience could support any orientation that the device could be held in.

As we designed the end-to-end experience on different form factors in Windows 8, we used the following principles:

- The experience tailors itself for all form factors: small screens, wide screens, laptops and desktops.
- The experience takes advantage of widescreen formats for multi-tasking and for full-screen video.
- The device can be held and interacted with in the way that is most comfortable.
- Developers have the opportunity to create one app that runs on all views and orientations across form factors with minimal effort.

We spent a considerable number of hours studying people as they used tablet devices in our usability labs as well as in their own homes. We've watched people who were already familiar with tablets, and we've watched people new to these form factors get started, and we stayed in touch with them for months afterwards. We noted grip styles, body postures, hand movements, and interactions with a variety of apps, device placements, and orientations. We saw a wide spectrum of variability, and we listened to users identify the factors influencing their choices about body and device orientation. Influences on these choices included anthropometric factors like hand size and thumb reach distances, ergonomic factors like repetition and fatigue, hardware factors like accessibility to hardware buttons, environmental factors like where they were using the tablet (e.g. in the kitchen, bedroom, or living room), and physical factors like if they were using it standing, sitting on a couch, or sitting at a desk. The number of combinations was staggering, contributing to the basic conclusion that postures, grips, and orientations change fairly frequently. Simply put, there's no one way to hold a device and people naturally seek to find a comfortable position and orientation that feels right for what they are doing with the device at the time.



We initially thought that landscape or portrait orientation was mostly influenced by personal preference. Each person that we watched rotated the device and each expected the device and UI to work with them at that moment. What was surprising was that as people become more familiar with the device and the apps they cared about, the most unique influencer on whether they rotated the device was the type of content on screen. If the content and experience felt better in landscape, people naturally used the app in landscape-mode. If the content and experience felt better in portrait, the app was used in portrait. As an example, most people prefer to watch a movie full-screen in landscape without black bars, while they prefer to read a long article or webpage in portrait, as it requires less scrolling. The preferences we heard people express were heavily influenced by their own sense of whether the app offered a great experience in that orientation. We've received questions and feedback about whether Windows 8 is "landscape first" or "portrait first." Our point of view is that both portrait and landscape orientations are important, and experiences can be great in either orientation. Rather than picking a posture and orientation for optimization, we designed an experience that makes sense

regardless of how the device is held, one that feels tailored for the app and its content.

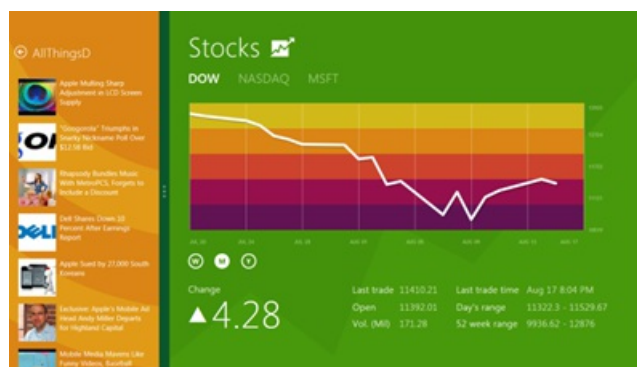
Our goals when looking at landscape and portrait were as follows:

- You can easily rotate your tablet to best suit your task or ergonomic posture.
- Rotation in Windows is fast and fluid.
- Windows rotates predictably across the system and apps – keeping the user in control.
- Developers can easily build high quality and intentional landscape and portrait layouts, depending on the experiences they want to enable.

Windows in landscape mode

Some people have asked why we showed so much of the Windows 8 user interface in landscape at the //build/ conference. Windows 8 is a reimagining of **all** PCs, and it's not just for tablets. It will run on hundreds of millions of laptops and desktops (designed for Windows 7 and new for Windows 8), many of which are and will be landscape-only. Also, in landscape and widescreen, we can offer a multitasking experience (snapping two apps side by side) and full-screen video playback without letterboxing. (In addition, for many of our larger demonstrations we are projecting to huge screens, which look better in landscape).

We've designed Windows 8 to be ergonomically comfortable in all orientations. We found that a comfortable posture for using a tablet in landscape is to hold in both hands and touch the screen with your thumbs. For this reason, we've designed the majority of the experience to be easily accessible under your thumbs. We also optimized the system to scroll horizontally, which feels fast and fluid in landscape as well as in portrait mode.



Windows in portrait mode

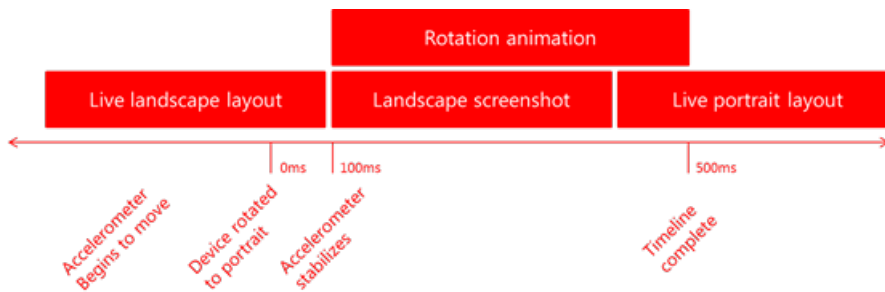
Windows is also designed to work great in portrait mode. We studied extensively scenarios like reading news in the web browser, looking at portrait photos, and scrolling through long lists of email messages, and we folded what we learned into designing a system that works seamlessly across orientations. We tuned the system experiences like the keyboard, file picker, and charms to work great in portrait as well as landscape. We wanted to make it so you don't need to relearn the system when you switch to portrait mode; it just works.



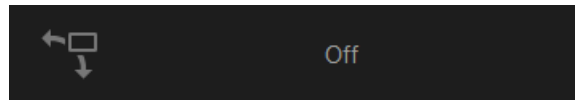
Rotation

Because one of our goals was to make the rotation transition between landscape and portrait feel fast and fluid, teams across Windows put significant work into streamlining the path of this transition, from the accelerometer hardware up through the graphics stack to the app.

An important part of the transition between landscape and portrait is the animation. The animation choreographs the appearance of a smooth transition between the two layout states. The timing of the animation is important as it had to be tuned to feel fast and responsive, while remaining smooth enough to make the transition not feel jarring. The Desktop Window Manager (DWM), which is foundational to the smooth animations in Windows 7 and Windows 8, orchestrates this animation.

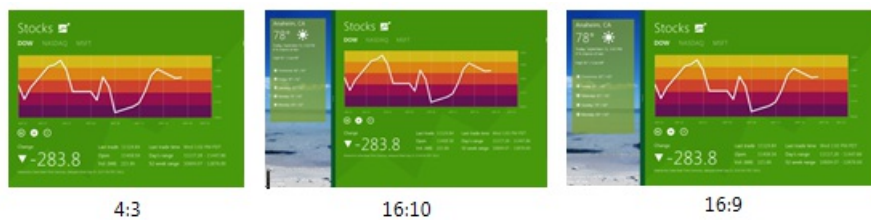


We are continually working to make the rotation as stable and predictable as possible, as we know how annoying an over-eager rotation can be. Before the rotation starts, the system waits for the accelerometer to stabilize to prevent accidental rotations. We also wanted to keep you in control of the rotation experience, so that it wouldn't be triggered accidentally. We introduced a hardware orientation lock to "override gravity" and keep the orientation the way you want it.

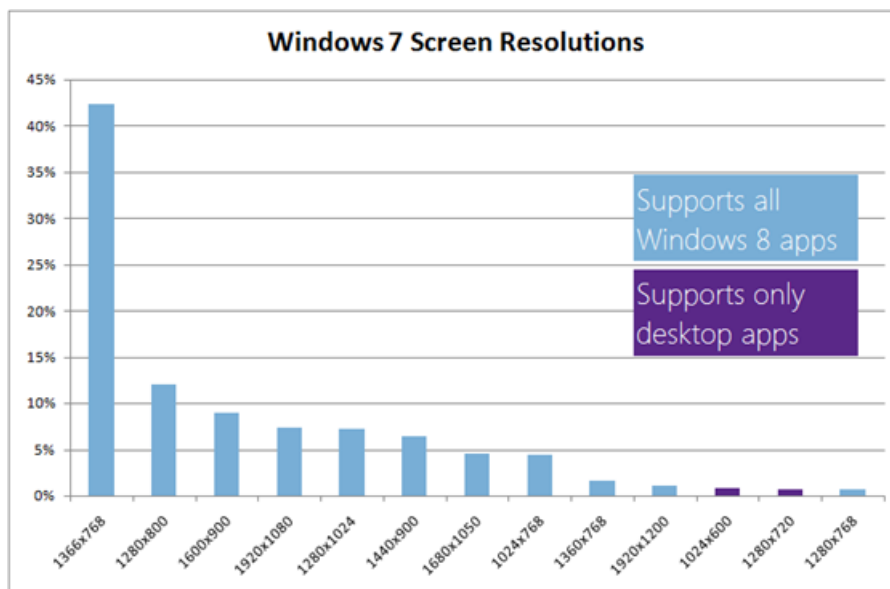


Different screens

In a future post, we'll cover in depth how Windows 8 scales to different screens from a developer perspective, but it's worth talking a little bit about screen size in the context of landscape and portrait orientations. Windows 8 will run on PCs with different sized screens and with different resolutions and different aspect ratios; from 4:3 screens that are closer to square, to widescreen 16:9 screens and everything in between. Our scaling platform enables Windows and apps to seamlessly adapt and reflow content to of these different screens and make use of the space. You can use many of these devices in either portrait or landscape. This diversity of choice is unique to Windows. You can choose the device and the orientation that best suits your needs and usage.



The minimum resolution that Windows 8 Metro style apps can run is 1024x768. We chose this size as it is a common size designed for use on the web, and a strong majority (i.e. 98.8%) of Windows users can run at this resolution or higher (see chart below).



We think it is important to have a minimum resolution for apps, as it allows all developers to design the smallest main view of their app without fragmentation across devices. This minimum also ensures that users do not see broken app layouts due to small screen sizes.

The resolution that supports all the features of Windows 8, including multitasking with snap is 1366x768. We chose this resolution as it can fit the width of a snapped app, which is 320px (also the width designed for many phone layouts), next to a main app at 1024x768 app (a common size designed for use on the web).

These limits will be enforced at runtime. We are breaking from past practices in not providing a workaround for this, given that the main motivation

is to make sure Metro style apps are designed to function fully at a specific published resolution. In the Developer Preview we did not have any useful runtime warning beyond the communication on the download site, which we will of course address for the beta. We did notice that a set of folks running on 600x800 virtual machines or 1024x600 Netbooks were inconvenienced and we're sorry about that. It is worth noting that the runrate of 1024x600 machines is very low as screen resolutions on the low end have moved to 1280x800, which supports Metro style apps without snap. This resolution is still a distant second to fully capable resolutions and we expect to see the shift to higher resolution screens continue as new PCs are brought to market.

Rotation and developers

In Windows 8, apps power the experience, so we worked to make it as easy as possible for developers to build both landscape and portrait views. As on any other platform, developers can choose which orientations they support for their apps, and how their experience is tailored. We expect most developers will provide a landscape view, as existing laptops and desktops make up the highest number of PCs in the marketplace today. However, if an app's experience could support both landscape and portrait, supporting the portrait view is just an incremental amount of layout work.

Using the same techniques used to build for the Snap feature or for different sized screens, developers can easily build portrait experiences. HTML5 developers will use CSS media queries to bind their layout style to the orientation of the system, whereas XAML developers will change their layout in response to view state events. In both HTML and XAML, all of the adaptive controls and templates that the platform provides will support both portrait and landscape. Additionally, the system automatically handles the transition animation without any additional effort from developers. If there is content in an app that is best suited for one orientation, developers can prefer one orientation or the other, and the system will keep the app in that orientation (if the device supports it).

For testing apps, Visual Studio 11 and Expression Blend allow developers to test their apps in portrait and landscape mode and on different screen sizes and aspect ratios, even if they don't have access to a tablet device.



You can choose the device that best suits your preferences, pick it up in the way that is most comfortable, and the experience accommodates that posture. Apps can take advantage of widescreen with multitasking, and still look great in portrait orientation with minimal additional work.

Here's a short video to show you the landscape-to-portrait transitions in action.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

We look forward to you trying it out!

Thanks,
David

Using Task Manager with 64+ logical processors

Steven Sinofsky | [2011-10-27T12:00:00+00:00](#)

Ryan Haveson, a group program manager on the User Experience team, wanted to update folks on some progress with Task Manager since the Windows Developer Preview. In this post you'll find the updated Task Manager tools for managing systems with a large number of logical processors. This is scalability well beyond desktop PCs, and is designed for the server and data center. A big part of Windows development is that the OS scales across a wide range of form factors and CPU architectures.

Note on comments. Please keep comments up to community standards. Just a reminder that there is no moderation of comments other than automated spam protection. –Steven Sinofsky

We talked about the [new Task Manager](#) in a previous post, and many of you have installed the [Developer Preview](#) and seen it for yourself. There was some interest on this topic so we thought we would take a moment to quickly share with you a feature that just showed up in our daily builds that you will be able to see for yourself in the future, in the Beta release.

The pictures below relate to a feature that server admins and people with access to mega-PC setups with lots and lots of logical processors often ask us about. One key thing to note up front is that here we are talking about logical processors, so if you have a system capable of hyper-threading, you will see multiple logical processors for each physical processor.

For those of you who have access to one of these many-processor systems, you know that the task manager CPU charts in Windows 7 have a few limitations:

- **Lack of real-time comparisons:** When you are looking at a CPU graph for lots and lots of logical processors, it is the anomalies that are interesting. At scale, it is pretty hard to compare moving line graphs of a 60-second window of CPU utilization to understand what is going on.
- **Tiny graphs:** When you get to the 64+ logical processor range, the graphs get pretty small. If you are trying to figure out which processors are being heavily used, you really have to squint to figure it out. When you get over 256 logical processors, you can barely read the charts at all.
- **Finding the processor ID:** If you do identify an anomalous graph, there is no easy way to get the corresponding processor ID.

Below is the Windows 7 Task Manager CPU performance tab on a system with 160 logical processors.

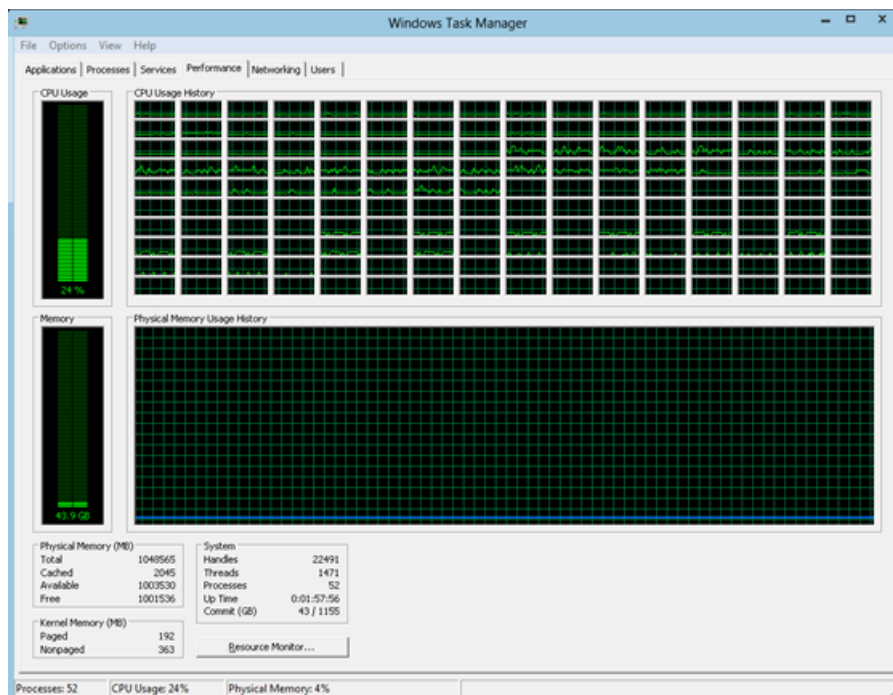


Figure 1: Old Task Manager showing 160 logical processors

As you can see, it is really hard to compare the cells in the CPU Usage History table to each other. The graphs are hard to read, and if you want to compare instantaneous CPU utilization, it is nearly impossible because each cell is showing a moving 60-second graph. Moreover, all the graphs in the CPU Usage History table look identical, so you can't easily find the processor ID for a specific graph. In our [previous post on Task Manager](#), we discussed the benefits of using a heat map as a visualization to convey and compare large amounts of numerical data. When we looked at designing the graphs for the "many-core view" of the new performance dashboard, a heat map was a natural fit.

In the screen shots below, taken from a current build of Windows 8, it is now easy to see all the logical processors at a glance and know which are

being utilized to high and low capacity.

(Note: The screen shots below show Task Manager on a system with 160 logical processors with a simulated workload.)

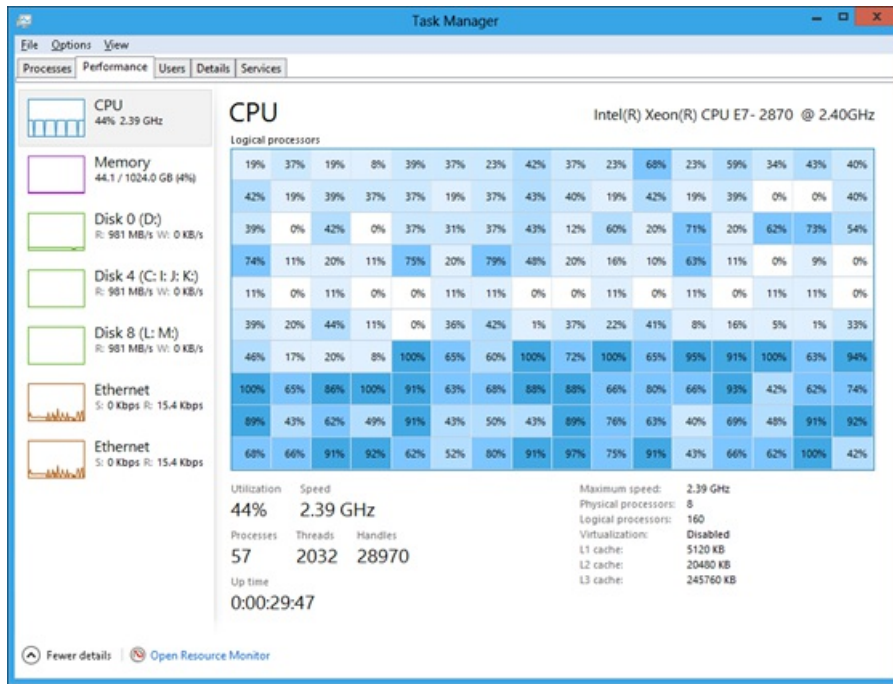


Figure 2: New Task Manager showing 160 logical processors

In the new CPU graph, you can also get the logical processor ID that maps to each entry via a tooltip, by hovering over the entry with the mouse.

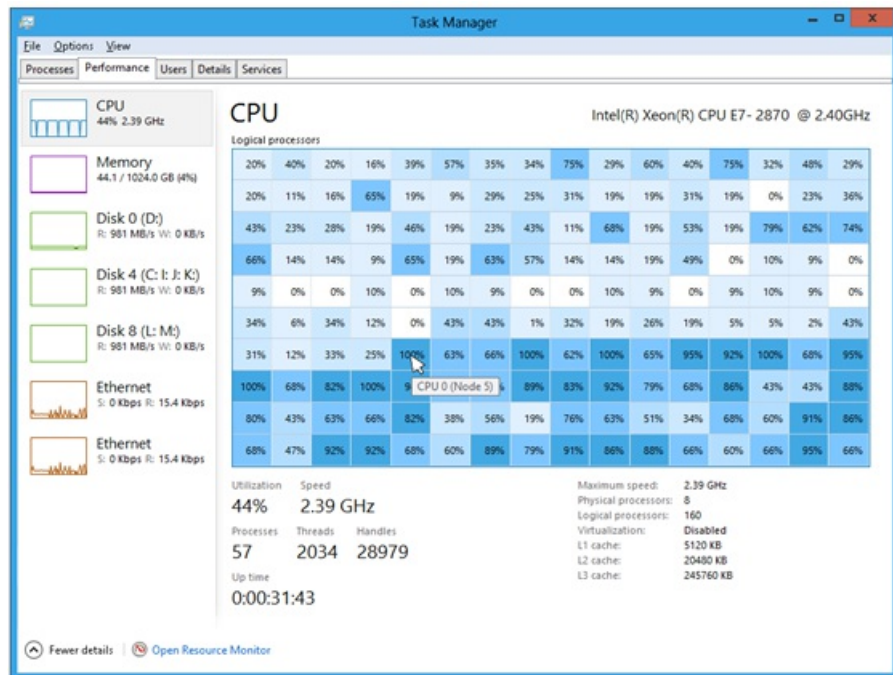


Figure 3: Tooltip showing the logical processor ID

A major benefit of a heat map is that it scales really well to large data sets. The new Task Manager will show as many logical processors as the OS supports (up to 640!). To make sure you always see the information at a meaningful size, when the data set gets too big for the window, the heat map scales to best fit, and a scroll bar appears as needed.

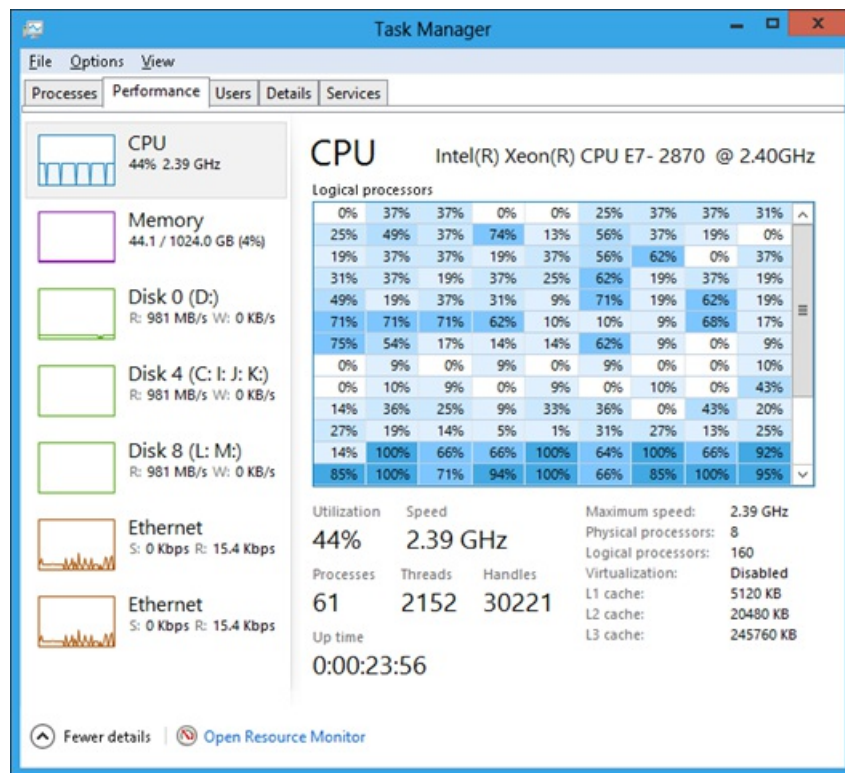


Figure 4: With 160 logical processors, the CPU graph scales using a scrollbar

For those of you who really like to (micro-) manage every last detail of your system, you can even set which logical processor(s) each of your processes can use. To do this, you first find the ID of the logical core by hovering over one of the cells in the heat map, then go to the Details tab, right-click the process you want, and click “Set affinity.”

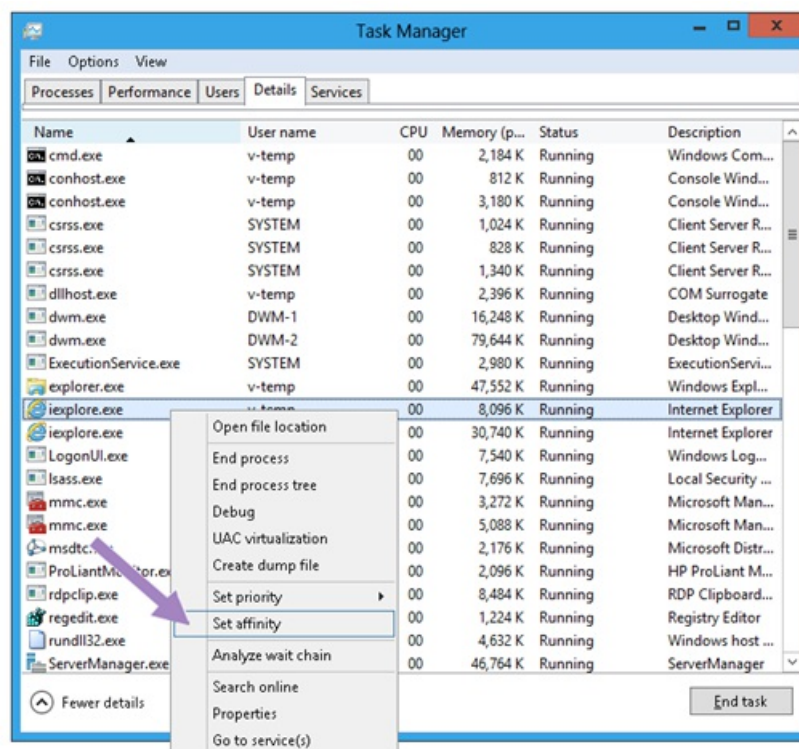


Figure 5: Set process affinity from the Details tab

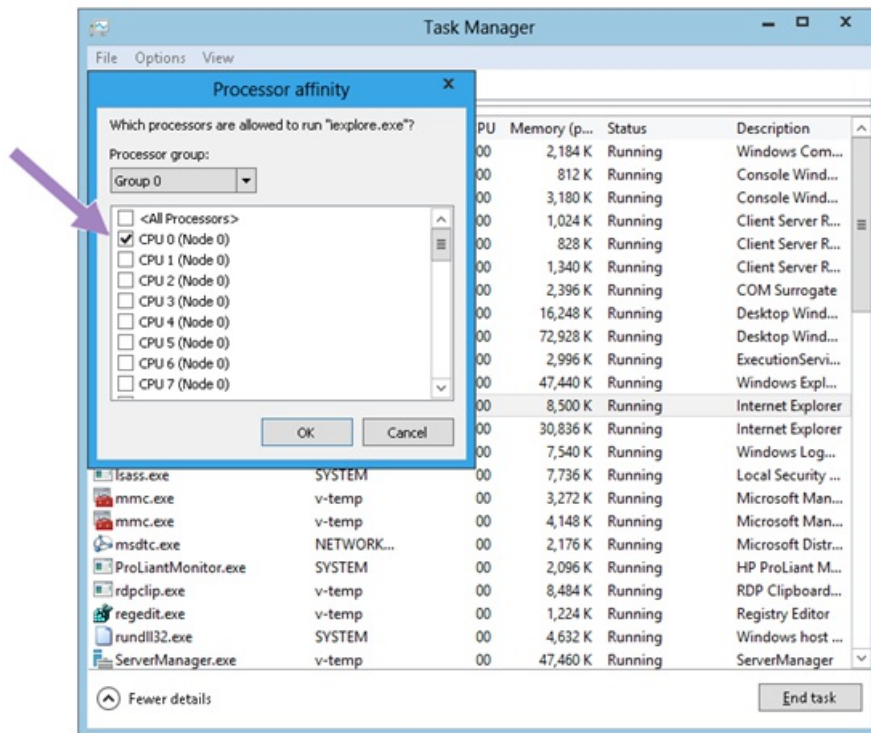


Figure 6: Select the logical processors for the process

Of course, setting processor affinity is only for the super-technical user who has a need or desire for that level of control—you can severely affect your system's power management and performance if you don't know what you are doing—so we made sure the OS would do a great job taking care of this for you. It is hard to do better than the sophisticated algorithms that Windows uses to automatically manage which processes are allocated to each logical processor based on hardware capability and topology.

--Ryan Haveson

Updating live tiles without draining your battery

Steven Sinofsky | [2011-11-02T13:00:00+00:00](#)

One thing that is becoming far more commonplace across all of our “screens” is the idea of lightweight notifications. Originally, Windows Gadgets were to offer this type of functionality—the idea is a quick heads up display for some critical information (news, weather, sports scores, or line of business events are a few examples). However, the startup time and model of Gadgets are not compatible with reducing overall power consumption (something that is important in a desktop and a laptop) or working to deliver the full-screen platform for developers. In addition, the Start screen of Windows 8 provides a much larger surface to have more of these notifications as well as a user-in-control interface for managing the updates (including usage of networking resources). In a modern experience where more and more information is available via push and in structured snippets, this provides a unique opportunity for developers and end-users. In this post, Ryan Haveson writes about the development of Metro style live tiles and how the architecture scales to large numbers of tiles while also reducing the overall power consumption and system load.

—Steven Sinofsky

We all know that performance and battery life are critically important for a successful release of Windows, and your comments continue to emphasize these attributes. @KISSmakesmeSMILE summed it up well by writing:

“...try matching or better yet, surpassing ... [competitor’s] battery runtime achievements on light/low load use.”

At the same time, we know that all modern environments (from PCs to TVs to phones) have some form of gadget, widget, or plug-in model that enables at-a-glance information consumption. Watching TV news, sports, or weather shows a structured screen of information with many sources coming together in real time. People expect to be able to quickly check their stocks, weather, email count, next appointment, line of business status, or even social networking status in a matter of seconds before getting right back to whatever else they were doing. In many ways, one could argue the PC has some catching up to do in this area compared to other devices. As we set out to design our notifications infrastructure, our challenge was in how to make the PC feel alive with activity and remain extremely efficient with respect to power and bandwidth usage. @AndyCadley’s words express the goal well:

“Treat all your “Metro” apps as if they are always running (but at zero impact to battery/performance)”

The Start screen also makes this efficient from a user model perspective by giving you a full screen heads up display without interfering with you desktop or Metro style apps while you are focused on those. In addition, not only did we want to make it efficient, we wanted to make sure that you could install as many notifying apps as you want, without having to worry about the impact on performance or battery life.

One thing we have noticed as we are using Windows 8 internally is that the ability to use the Start screen as a unified and highly readable heads up display for line-of-business applications has become a productivity enhancer. We are seeing a lot of interest in apps that are primarily about notifications. With the scalability of our new push notifications platform, Windows 8 can deliver this capability with minimal system impact, which is a big improvement over the multitude of mechanisms that exist in Windows today. It is not hard to see a scenario, especially early on, where even the most hardcore desktop-only person will find a lot of value in the Start screen as a centralized and well-presented (and controlled) notification area that is just a keystroke away.

Goals of the notification platform

Allowing hundreds of app tiles to be alive with activity, and simultaneously making sure that we don’t degrade performance makes it seem like we have contradictory goals. After all, “activity,” by definition, consumes resources: getting a notification from the cloud uses the network, and rendering the notification on a tile uses GPU/CPU resources, etc. In order to get the design right, we knew we had to stay focused on the goals we started out with:

- **Allow hundreds of live tiles** without degrading performance
- **Go beyond balloons, badges and text**, with beautiful images
- **Make it easy for developers** so they can just “fire and forget”
- **Achieve real-time delivery** so delivering “instant messages” is instant

Based on these goals, the first fundamental architectural decision that we made was that the platform would be **data-driven**, that is, no app code should run in the background to power the Start screen.

If you think about the anatomy of a notification delivery system, it involves several pieces: logic for when to connect, authentication, local caching, rendering, error handling, back-off algorithms, throttling, etc. In addition, the system has to deal with service-side issues such as knowing when you are connected or not, so it can cache undelivered content and handle complex scenarios for retrying. Can you imagine if every single app with a live tile had its own version of all that client/server code? Not only would you have different bugs in each implementation, but you would have duplicates of essentially the same code for each app loaded in memory, with code that is constantly being paged in and out to the disk. This would be really inefficient because it would mean all of your apps would be running all the time to keep the Start screen alive. Even on a machine with lots of memory, system performance would eventually grind to a crawl.

If you read Bill Karagounis’s post on how we [reduced the memory footprint](#) in Windows 8, you know that performance degrades as you increase

the number of processes, DLLs, services, etc. that are running. If each live tile was running with its own code, we would not have been able to achieve our first goal of allowing hundreds of live tiles without degrading performance.

Our solution was to build a data-driven model. This means that a developer can express their tile using a set of predefined properties and templates, in this case, using an XML schema. The XML tile data is then sent to the Windows Push Notification Service (WNS) via a simple HTTP POST and then we take care of the rest. All the code for connecting, retrying, authentication, caching, rendering, error handling, etc. is done in a uniform and power-efficient way.

Here is an example of one of the many [tile templates](#) that developers can use for their Windows 8 apps. This one consists of a text field and a single image, but there are many other templates to choose from.



Figure 1: Example template ([TileWideImageAndText](#))

Here is the corresponding XML code that describes the above tile:

```
<?xml version="1.0" encoding="utf-8"?>
<tile>
  <visual lang="en-US">
    <binding template="TileWideImageAndText">
      <image id="1" src="http://www.fabrikam.com/kickflip.png"/>
      <text id="1">First ever surfboard kickflip recorded in Santa
        Cruz</text>
    </binding>
  </visual>
</tile>
```

The decision to use a data-driven model allowed us to achieve the first two goals (performance and a high-fidelity experience), but we still had to figure out how to achieve real-time delivery and fire-and-forget-it efficiency.

There are two high-level design patterns with client/server content delivery: polling & push. **Polling** means that the client checks with the service on a regular basis (for example, every 90 minutes) to see if there is new content. **Push** means that when there is new content, the service sends the data down to the client directly.

The only way to support instant notifications with a polling model would be to poll on a sufficiently high frequency (like every 5 seconds), so if a new message arrives, you'd see it pretty much instantly. But doing so would kill our performance goals—with a 5-second poll interval, the network radio stack would never be idle, battery life would be horrible, and desktop machines would always be powered up. It would be a little like talking on your cell phone all day long—your phone's battery wouldn't last long. On top of that, it would be extremely wasteful to check the server every 5 seconds for content, since most of the time there would be nothing new. Historically, system tray notifications and desktop Gadgets introduced in Vista have been implemented using a polling mechanism. But with any polling mechanism, the interval is still not short enough for today's real-time services that are *instant*.

Thus, for Windows 8 we architected a push-based service. This was a big decision because it meant we would need to build a platform at a global scale, eventually powering the tiles for hundreds of thousands of apps and over a billion people. But the value was clear: developers would get super-efficient real-time notifications to their customers for free, without having to build or maintain their own persistent connections to the client.

The push notification platform

Let's take a closer look at the various components of the platform to explain some of the more subtle parts of the design. In the diagram below you see three key entities:

1. **Windows Push Notification Service (WNS):** This powers live tiles and toast notifications.
2. **App service:** This is the web service that a Metro style app runs (e.g. from their existing website), which sends toast notifications and tile updates via WNS. Examples of this would be the back-end service for the Weather app that shipped in the [Developer Preview](#), or a back-end service hosting photos for a social networking app.
3. **Windows 8 client platform:** This represents the actual PC and the sub-components in the OS that form the plumbing for the end-to-end experience.

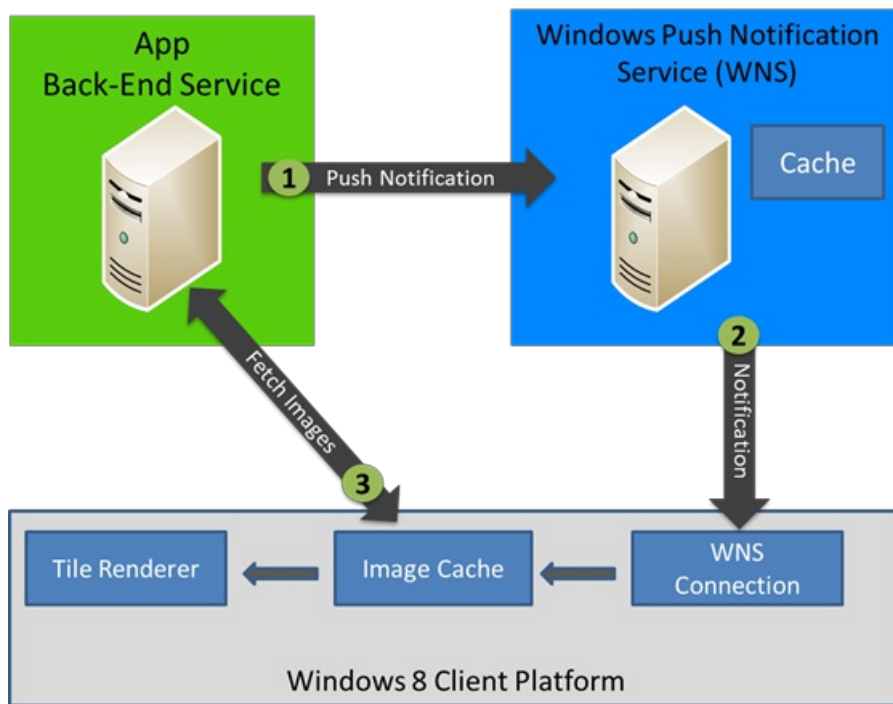


Figure 2: The push notifications platform

Let's walk through a typical usage scenario to illustrate how this works. Suppose that the app service is a social networking site that sends a tile update when someone comments on your photo (this could just as easily be a line of business app that updates me when a bug is assigned to me or an expense report needs attention for example). When there is an update, the app service sends a notification to WNS (**Step 1** in the above diagram). From there, WNS pushes the notification down to the client (**Step 2**). When it is time to show the tile update on the Start screen, the OS fetches that image from the app service based on the URL contained in the notification XML (**Step 3**). Once the notification and the image are downloaded, the app renders the live tile based on the template specified in the XML, and presents it on the Start screen.

As stated earlier, one of our goals was "fire and forget." So, to ensure that developers did not have to write complex caching and retry mechanisms for when the PC is not connected (e.g. if it is a laptop that is sleeping), we cache one notification per app in the WNS cloud until the next time that PC is online.

As we designed the client platform components, we wanted to make sure that everything was engineered for high performance and low power consumption. One of the key parts of this was separating the notification payload from the image payload. A typical notification XML is less than 1KB of data, but an image can be up to 150KB. Separating these allowed us to save significant network bandwidth for scenarios where there is a lot of duplication of the images. For example, the image for a tile may be a profile picture of a friend, which your PC can download once and cache locally to be reused. Separating the notification from the image also allowed us to be smart about discarding unused notifications before we go through the expense of downloading the image. If my device screen is off and is sitting in my bedroom while I am at work, there is no point in downloading images for tiles that will just be replaced by subsequent updates before the next time I use the device.

The authentication model

Because live tiles and notifications represent a key part of the app experience, it is important that the communications channel is authenticated and secure—all the way from the app service to the tile on your Start screen. It would be pretty bad if an app or a rogue web service could just update any tile on your machine. For that reason, we use an anonymous authentication mechanism that uniquely identifies the connection between the PC and WNS. Apps and app services also authenticate when communicating with WNS. Authenticating both connections to WNS helps to protect against abuse of live tile updates, such as spoofing attacks. The authentication mechanism used by WNS explicitly ties the application and service together in a way that keeps other applications (or nefarious individuals) from sending content to a tile that they do not own. And of course, all communication takes place over a secure channel.

All of this works regardless of whether or not you sign in to Windows using a Windows Live ID. Of course, as Katie Frigon talked about in [her post on signing in with a Windows Live ID](#), Windows 8 is best when you have a connected account as it enables a lot of enhanced experiences such as app cloud storage, roaming Windows and app settings, and single sign-on to multiple apps. Because the push notification platform uses an anonymous authentication mechanism, even if you do sign in with a Windows Live ID, the developer of the app can't use the notification pipeline to discover your Windows Live ID, system info, or location.

Building the service to scale

Earlier in this post we mentioned that we had to engineer the platform to support an incredibly large number of users and apps. To give you an idea of this scale, the graph below shows the number of notifications that apps are sending to Windows 8 per day. As of a couple weeks ago, we were already sending almost 90 million tile updates per day, and we are not even at beta yet!

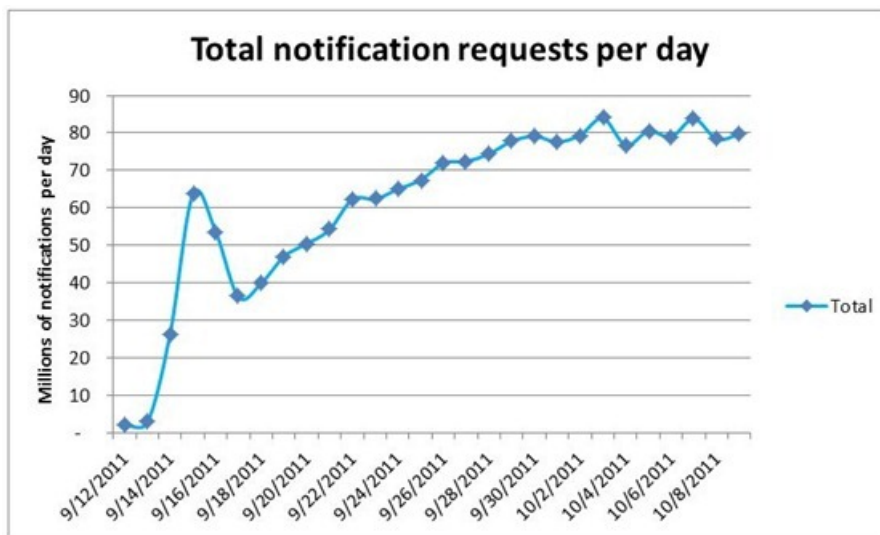


Figure 3: Notifications per day sent to Windows 8 Developer Preview build

The Stocks app is one of the popular test drive apps from the Developer Preview build. The following graph shows the total number of live tiles registered with this app in the first month since release of the Developer Preview build.

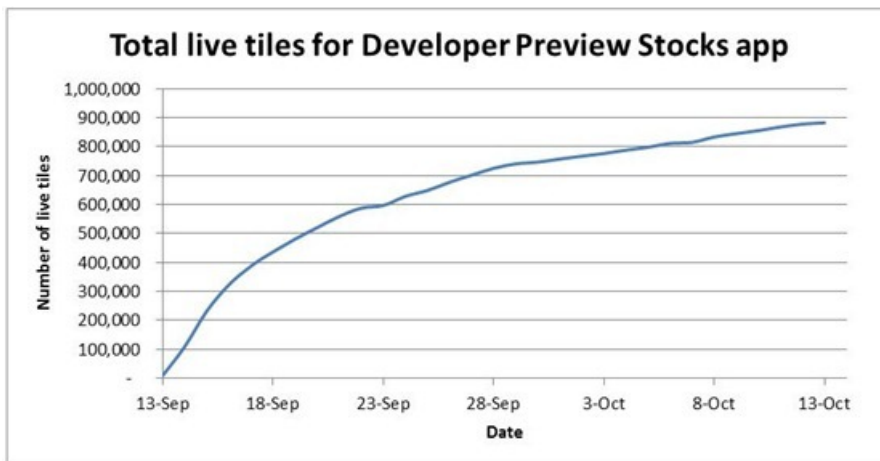


Figure 4: Live tiles registered to the Developer Preview Stocks app

When the Developer Preview was released, we started watching traffic coming through the data centers, carefully monitoring our scale-out. Here is a visualization of the actual geographic distribution of notifications in the first few days after the Developer Preview was released at //build. Note that the data represents units per square mile and was fitted to a logarithmic scale to account for a wide range of density values.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

The design of WNS is based on the Windows Live Messenger service architecture, and in fact, the service part of the notifications platform was built by the same team. There are not many teams in the world with the expertise and knowledge to be able to build a globally scalable service that can ramp up to such large numbers so quickly. Here are a few statistics to give you an idea of the scale of the Windows Live Messenger service today:

- 300M monthly active users
- 630M daily logins
- 10B daily notifications
- Over 40M peak SOC (simultaneous online connections)
- Over 3000 machines routing messages around the world

Transparency into tile resource usage via Task Manager

We were so passionate about the performance aspect of notifications platform that we added metrics in the new Task Manager to allow you to keep track of how much bandwidth the tile platform is consuming for each of your applications. In general, resource usage for tiles should be relatively low. For those of you running the [Developer Preview](#) build, go to the app history tab in Task Manager and look at the “Tiles” column to see how much bandwidth each of your live tiles has consumed over the last 30 days.

Showing total resource usage by modern applications since 9/17/2011. Last updated 10/17/2011 10:59 AM
[Reset usage data](#)

Application	CPU (Time)	Network (MB)	Metered network (MB)	Tiles (MB)
BitBox	0:00:00	0	0	0
BUILD	0:00:32	5.6	0.1	0
Control Panel	0:00:01	0	0	0
News	0:00:28	71.9	57.2	0.1
Notepad	0:00:00	0	0	0
PaintPlay	0:00:36	0	0	0
Piano	0:00:24	0	0	0
Picstream	0:00:00	0	0	0
Remote Desktop	0:00:00	0	0	0
Socialite	0:00:55	3.5	0	0
Stocks	0:02:49	1.4	0.1	0.1
Store	0:00:00	0.1	0	0
Sudoku	0:00:00	0	0	0
Tile Puzzle	0:00:00	0	0	0
Treehouse Stampede	0:00:00	0	0	0
Tube Rider	0:00:00	0	0	0
Tweet@rama	0:04:56	0.6	0	0.1
Weather	0:07:02	1.1	0.1	0.1

Figure 5: Resource usage of live tiles shown in Task Manager app history

Summary

In Windows 8, we set out to design a notifications platform that would provide at-a-glance information, without all the performance and battery life concerns that face traditional plugin and gadget-based models. To that end, every design decision we made was viewed through the lens of performance and battery life efficiency. To make it easy for app developers to participate, we built the Windows Push Notifications Service so they could create live tiles without having to write complicated network connectivity code. And because WNS uses standard web technologies, such as HTTP POST, it's easy for developers to integrate notifications based on their existing web services.

The result is a notifications platform that delivers at-a-glance information while allowing you to install as many apps as you want without worrying about the impact on performance or battery life.

--Ryan Haveson

Building a power-smart general-purpose Windows

Steven Sinofsky | [2011-11-08T15:00:00+00:00](#)

In this post, we look at the broad topic of developing an OS to reduce power consumption. We've seen an ever-increasing emphasis on power management in the OS from two perspectives. First, as Windows 8 comes to market, it is easy to see two-thirds of all PCs shipping as portable devices operating on batteries some or most of the time. And second, in the workplace, there is an increasing demand for desktop machines with a reduced carbon footprint as we look to save energy wherever we can. In all cases, this goes beyond standby/hibernate/resume performance and gets to the heart of this post which is about reducing the overall power consumption of the OS and providing OS support for power-saving capabilities in modern hardware. Pat Stemen, a program manager on our Kernel team, authored this post.

--Steven

Battery life and power consumption continue to be some of the most important topics in the computing industry. We wanted to give you a look at how we think about power management for Windows 8, and how we measure power consumption on a daily basis. We consider power management a core OS capability that is critical on any chip architecture and any PC form factor.

Our goals

We have 3 goals in mind when engineering Windows 8 power management:

- **Let the hardware shine.** We built Windows 8 such that the power efficiency of the hardware platform shines through, regardless of whether the system is a SoC-based Windows tablet or an SLI-equipped gaming PC. We designed our power management interfaces in a consistent, standardized way across all platforms. This allows our hardware partners and application developers to focus on their unique innovations and experiences instead of the differences in platform hardware and power management.
- **Continue to deliver great battery life.** Windows 7 delivered a significant reduction in power consumption and increase in energy efficiency, particularly mobile PC battery life. (In fact, you can read how we thought about it in [this e7 blog post](#).) In Windows 8, we want to maintain that same level of efficiency on existing PCs even as we re-imagine the rest of Windows.
- **Enable the smartphone power model.** One of the coolest things about the System-on-Chip (SoC) platforms you've seen us talk about at [CES](#) and [//BUILD/](#) is their capability to quickly enter very low-power idle states. We want to leverage that ultra-low idle power to bring the constant connectivity and instant-on features of the smartphone power model to capable Windows 8 PCs.

Why it matters

Most of us are probably familiar with one impact of good power management—long battery life on our mobile devices. Mobile computing devices are highly scrutinized for their battery runtime with good reason—we are using them on battery more often than ever before. Delivering consistently long battery life requires a lot of well-executed coordination between the underlying hardware, operating system, and application software. (Battery capacity and its quality over time also have a big influence on runtime.)

In addition to mobile battery life, the next most tangible impacts of good PC power management are reduced energy costs and environmental impact. The benefits of power-managing enterprise desktop PCs and servers in the datacenter are typically positioned as reducing the amount of money required to power and cool those systems as well as the resulting reduction in greenhouse gasses required to produce the energy for them. It's hard to underestimate the impact here—very small changes done well in Windows can result in very large positive environmental impact because of our scale. In many markets around the world, increasing electricity consumption is putting more demand on every aspect of the workplace to reduce power consumption. PCs are a significant source of potential savings.

Power management is also in a fine balancing act with system performance and responsiveness. As an example, we can easily reduce processor performance to save power, but as we do so, we increase the time required to process a given workload. Doing a great job of balancing power and performance is a key requirement across the Windows user experience.

We are also observing a lot of tech blogs reviewing the latest processors and hardware platforms, not just on GHz and benchmark performance, but also on the power consumed to execute the workload. These measurements combine power and performance into **energy efficiency**—how much power does it cost to execute some fixed workload. “Performance per watt” is the same thing.

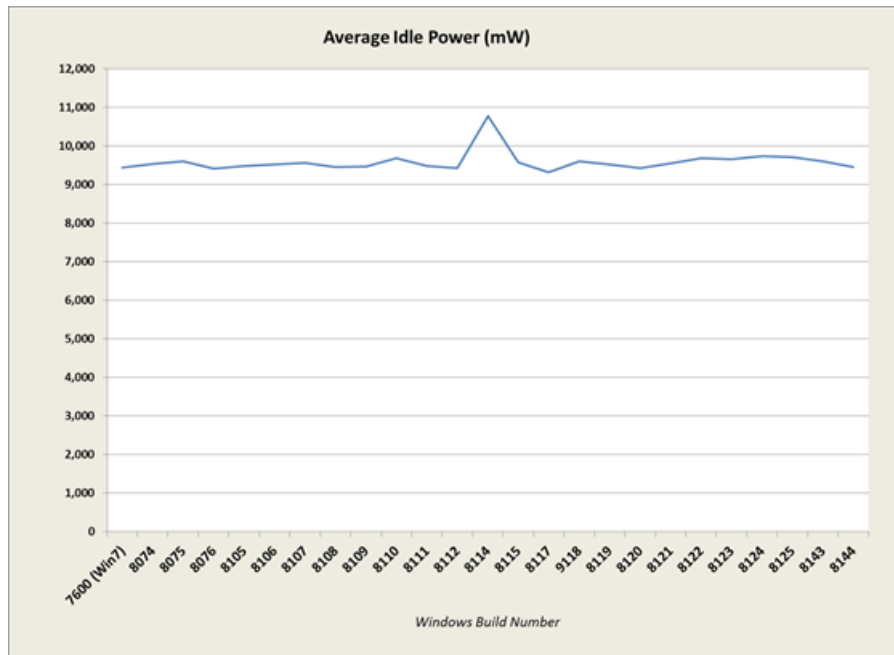
However, the thing that excites us the most is how power management is fundamental to all aspects of PC platform design. Power management directly impacts attributes including thickness, weight, acoustics (fans and their speed), skin temperature, cost, screen size, resolution, RAM quantity, etc. Hardware that is thin, light, always connected, and runs all day on battery is cool. We love being a part of enabling that for the Windows ecosystem.

Power as a resource

We think of power as a critical system resource, just like CPU utilization, hard disk activity, or memory consumption. Because it is a core resource, we track how much of it Windows consumes daily in each daily build of Windows 8. This allows us to quickly spot changes in power consumption and work closely with the development team to find the root-cause of the issue, develop, and then test any changes required. As power consumption is reduced, we get a new baseline to measure against future daily builds.

Raw power consumption (e.g. how many milliwatts) on any given Windows PC is the result of a number of platform factors including the CPU and chipset, the type and amount of RAM, the speed, type, and capacity of the storage drive, screen size, etc.

In order to have a consistent baseline, we identify a set of reference platforms and measure their power throughout the Windows 8 development process. Our reference platforms are representative of the type of machines commonly used by our customers. They include platforms from each of our major silicon chip partners. We use power consumption in Windows 7 and Windows 7 SP1 as our baseline, and compare that to measurements taken from Windows 8 builds throughout the development process.



Total system idle power measured in our lab for one reference platform. You can see a change that caused a ~1.25W increase, which was fixed in a subsequent build. Note that some variance between runs and builds is expected.

We measure the power consumption of many software workloads on each of our reference systems. The workloads we measure are similar to 3rd party battery benchmarks in that they are designed to be representative of core scenarios that we all do with our machines every day. Our core workloads include Idle, Web Browsing, Video Playback, Audio Playback, and Standby.

You might be thinking that spending time on the idle workload isn't useful—after all, few people boot their machines and just let them sit at the Start page and do nothing else. While that is true, we think of the idle workload as the floor of power consumption for the system—it's the minimum amount of activity that the system has when active. Reducing power consumption at idle reduces the base power consumption of most other workloads, including video and audio playback. Plus, a lot of workloads have significant idle time in them—from small amounts of time between keystrokes when typing, to the minutes between slide transitions when giving a presentation.

The common way to measure power for a mobile Windows PC is a battery life rundown test, in which the battery is charged to 100% and then the battery is drained to 0% while repeating the workload. This method works, but is prone to error as the battery capacity naturally degrades over multiple charge/discharge cycles. Each rundown test is an extra charge/discharge cycle, and we test every day. Thus our measurements could drift over time due to battery degradation.

We have a power measurement setup in our performance lab that allows us to provide reference platforms with direct DC power and measure the consumption. We mentioned the lab and overall capability in our [IE blog post on browser power consumption](#), but didn't mention that we have a number of real reference laptops set up to measure power consumption every day. The power supplies and their metering capability are automated with test software so that we can continuously install Windows, measure power across scenarios, and then analyze the results with each new build of Windows 8.



DC power supplies with built-in measurement capability



Reference platform instrumented for DC power supply and measurement

How software influences power consumption

Software can influence power consumption by consuming resources—CPU, disk, memory, etc.—as each of those resources has a power cost associated. Software also influences power consumption through the OS and driver software responsible for managing hardware power states.

Windows 8 features 3 key innovations to improve how software influences power consumption—the Metro style app model, idle hygiene, and a new runtime device power management framework. We will give you a brief overview of how these innovations improve power consumption in this blog post.

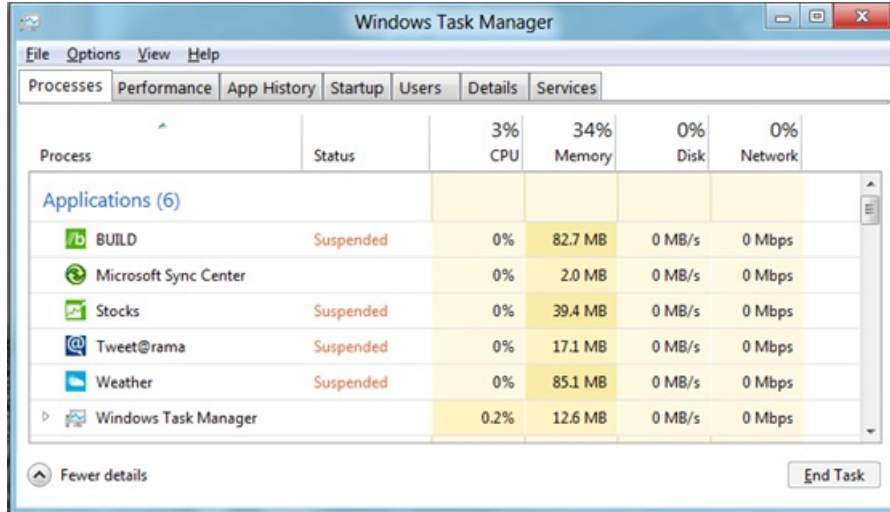
The Metro style application model

Most of us have experienced the influence of software on power consumption first-hand. It might be that you have an app on your phone that goes through battery quickly or you've heard the fan turn on in your laptop when playing a game or computing a spreadsheet. These are all examples of applications directly consuming CPU, GPU, network time, disk and/or memory.

One of the new power management innovations in Windows 8 isn't a power management infrastructure feature; it is the Metro style application model itself. The Metro style application model is designed from the beginning to be power-friendly. The power management benefit is that the model makes it easy for developers to ensure their application is running only at the right time—applications in the background are suspended such that they do not consume resources and power when not in use.

Of course, we recognize that background activity is a critical component of apps that are always connected and responsive. The Metro style application model and the underlying WinRT support background activity through a new set of capabilities called *background tasks*. (See this [Introduction to Background Tasks](#) for more details.) Background tasks make it easy to perform background activity in a power-friendly fashion. They also enable developers to continue to deliver responsiveness and “freshness” in their applications, but the mechanisms are different than the existing Win32 model because of the desire for a fast-and-fluid interface and the other key attributes of Metro style apps (see [8 traits of great Metro style apps](#)).

We’ve engineered background tasks and the overall Metro style application model to enable a new level of app responsiveness, while at the same time considering overall system attributes including power and memory consumption.



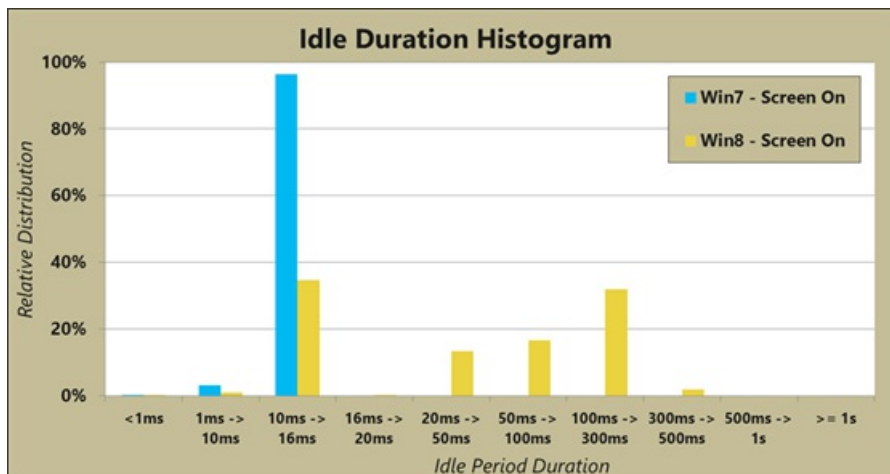
Task Manager showing suspended Metro style apps

Idle hygiene

Software can have dramatic influence on power consumption even without consuming a lot of resources through intermittent idle activity. We refer to improvements to idle activity as **idle hygiene**.

Most PC platforms feature processor and chipset idle states that allow the hardware platform to stop the clock or completely turn off power to parts of the silicon when they are unused. These idle states are absolutely critical to enabling long battery life, but they require a **minimal residency duration**—that is, you have to be idle for long enough to make the transition in and out of the idle state worthwhile in terms of power used. This is because some power is consumed on the way into and out of the idle state. Software most effectively uses these idle states when there are as few exits from the idle state as possible, and the duration of the idle state is as long as possible.

We track the idle efficiency of Windows 8 using built-in [ETW Tracing](#), some additions to the [Windows Performance Analyzer](#), and a basic histogram. Below, you can see the difference in idle durations between Windows 7 and Windows 8. When the screen is on, we’ve already moved the bar significantly from a maximum idle duration of 15.6ms in Windows 7 to 35% of our durations longer than 100ms in Windows 8! With the screen off and during Connected Standby, our idle durations are even longer, currently in the tens of seconds.



Runtime device power management

PCs attain their longest battery life when all devices, including the processor, storage, and peripheral devices enter low-power modes. Almost every device in the modern PC has some kind of power management technology, and **runtime device power management** determines how we

use those technologies seamlessly without impact to the user experience. A really good example of runtime device power management is dimming the automatic display after a timeout in Windows 7.

Just to underscore how important device power management is, we have seen many systems where not enabling a single device's power management features can easily reduce total battery life by up to 25%! (It's worth noting here that disabling a device in Device Manager is almost equally bad—most devices are initialized by firmware at their highest power modes and require a device driver to get them to a more nominal power consumption.) You can diagnose some device power management problems using the built-in **powercfg.exe** utility in Windows 7 with the /ENERGY parameter. The output of /ENERGY is an HTML file that gives you a view of which devices and software are potentially running in a power-consuming state. Of course, using the factory image for your PC that came loaded with OEM and vendor-supplied drivers is almost always the best way to ensure the devices in your PC are well-behaved for power management.

Efficient power management of devices is performed by the driver for the device, in conjunction with the Windows kernel power manager and platform firmware. The power manager makes it easy for the drivers of these devices to implement their power management routines and coordinate any power state transitions with other devices on the platform.

For Windows 8, we've built a new device power framework that allows all devices to advertise their power management capabilities and integrate them with a special driver called the **Power Engine Plug-in** or **PEP**, designed for SoC systems. The PEP is provided by the silicon manufacturer and knows all of the SoC-specific power management requirements. This allows device drivers like our USB host controller or a keyboard driver to be built once, and still deliver optimal power management on all platforms from SoC-based PCs to datacenter servers.

We are hard at work with all of our ecosystem partners to deliver the low-power and long battery life technologies we all want in our Windows 8 PCs.

--Pat Stemen

Minimizing restarts after automatic updating in Windows Update

Steven Sinofsky | [2011-11-14T10:00:00+00:00](#)

Before the Internet, updates such as service packs and "patches" were impossibly hard to come by. You ordered upgrade "media" or maybe bought a magazine with a CD in it. Of course, the Internet changed all that. In fact, when [ftp.microsoft.com](#) was first set up, among the first services was the ability to get updates for MS-DOS and Windows. With the introduction of Windows Update, we invested heavily in building not just a software delivery service, but a commitment to delivering high quality updates in a timely manner. It took some time to get to the point where customers trust these automatic updates, and we're proud of how far we've come. Today Windows Update is one of the largest services on the Internet by several measures, and of course we're using Windows 8 development as a chance to improve the experience of product updates too. This post was authored by Farzana Rahman, the group program manager of our Windows Update group.

—Steven

When it comes to Windows Update, one of the most discussed topics is the disruptiveness of restarts in the course of automatic updating. And for good reason—restarts can interrupt you right in the middle of something important.

The obvious question to ask first is why does the installation of updates even require a restart at all? Ideally, we would like all update installations to happen seamlessly in the background without a restart. But, in reality, there are situations where the installer is not able to update files because they are in use. In these cases, we need to restart your machine to complete the installation. The automatic updating experience thus needs to be able to handle cases where restarts are required.

We know this architectural challenge is one that frustrates administrators and end-users alike, but it does represent the state of the art for Windows. It is important to understand that for many updates, even if you could continue running the existing code that is already in memory, it is that very code that is a security vulnerability (for example), so the risk to the security (or reliability) of the machine would remain until you restart your machine. We'll keep working on this one. In the meantime, applications that support the [Windows Restart Manager](#) (introduced in Windows Vista) can return you to precisely where you left off after a restart.

In this blog, I want to talk about some of the improvements we are making to the automatic updating experience in Windows 8, which will make restarts a little less annoying.

First, some facts about Windows Update

Windows Update (or WU, as we like to say within the team) currently updates over 350 million PCs running Windows 7 and over 800 million PCs across all the supported Windows platforms. There are actually many more PCs updated by WU indirectly if you account for our [Windows Software Update Server](#), and for those machines (or customers) that do all updates manually for any number of reasons.

Since its genesis over a decade ago, the Windows Update experience has evolved quite a bit to adapt to a changing ecosystem, especially the changing requirements around security. And Windows Update has been quite successful in updating PCs in time to stay ahead of large-scale exploits against Windows.

Since the introduction of automatic updating, we have constantly worked to tighten the time it takes to distribute new updates to everyone who uses WU. The chart below (figure 1) shows us how fast downloads and installations occur on Windows 7, from the time of release of an update. The speed of each download is primarily determined by the internet connectivity of the PC, something that WU has no control over, so it is interesting to see below that the majority of update activity occurs in the first three days after release. This three-day number is a key one that I will come back to when we talk about improvements in Windows 8.

In one week, 90% of users worldwide who need the update have successfully completed installation, including the restart, with the number of installations pretty much flattening out after that.

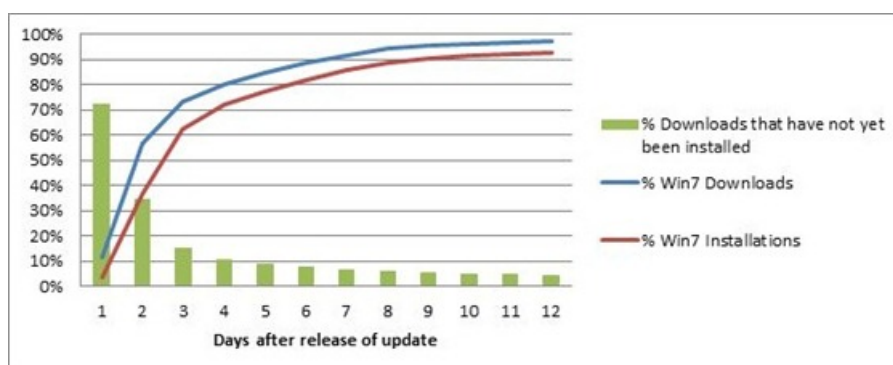


Figure 1 – Completed download and installation of updates from time of release of update

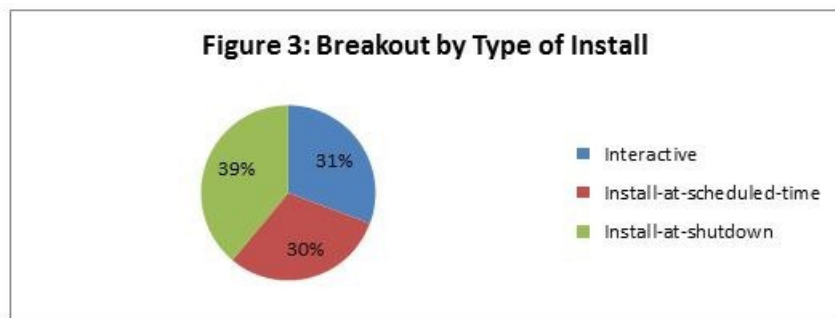
The balance of how broadly and how quickly we can update has proven beneficial to our users to the point where updating is mainly viewed as a background maintenance task (and justly so!) with nearly 90% of users choosing to update automatically on Windows 7. That's 90% of the total user base telling us to automatically install updates without showing any notifications, or asking for confirmation.

Windows 7	
Automatically install updates	89.30%
Notify me before install	2.38%
Notify me before download	3.44%
Never check for updates	4.88%

Figure 2 – Usage of various modes of automatic updating

Automatic updating and restarts on Windows 7

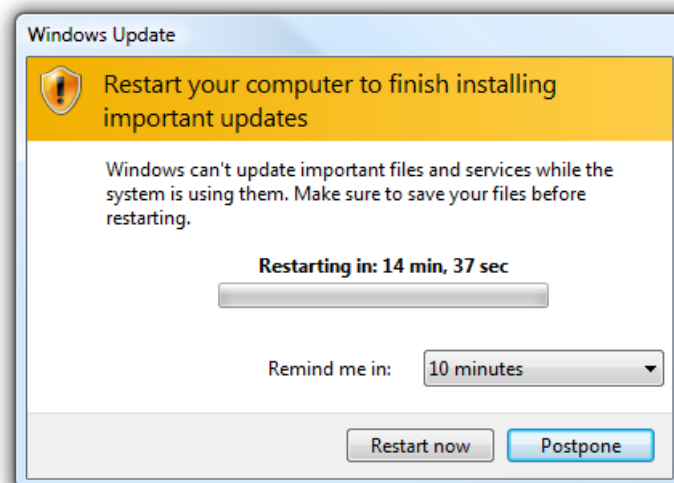
The next logical question to ask is what is the install experience for people who have chosen to automatically install updates? Below, data collected anonymously from WU gives an insight into the various modes of installation for those who have chosen to install automatically.



As you see above, there are 3 main categories of automatic update installations. Here is what we learned from analyzing each category.

Install-at-shutdown – The majority of automatic update users (39%) are updating when they shut down their systems. For these users, there is no automatic restart because the system can complete all steps of the installation during shutdown. This is the least disruptive experience for users, and so we do want to “hitch a ride” whenever we can on user-initiated shutdowns instead of inconveniencing users with a separate restart.

Install-at-scheduled-time - For the 30% who are scheduling automatic updates, their installations start at a scheduled time (the default is 3 AM in the time-zone where the PC is located) or the next time the user logs in (if we miss the 3 AM window). WU automatically completes any restarts necessary to finish the installation. To ensure that you get the chance to save any important files and data before the restart, we show you a 15-minute countdown timer before the restart.



A fifteen-minute countdown timer warns you of the restart

Allowing restarts to occur without user interaction has helped us to rapidly update a major portion of the Windows ecosystem with critical updates. On average, within a week of releasing a critical update, 90% of PCs have installed the update (see Figure 1). On the other hand, this behavior of automatic restarts has some unintended consequences for the user. Restart can occur without notice, and might occur monthly or even more often if there is an out-of-band update. This unpredictability can potentially result in loss of user data. Most of our automatic installs and the subsequent restarts happen at 3 AM, when users are not around to save any important work. We have heard a lot of painful stories of users coming back to their PCs in the morning to find that a restart occurred, and that some important data was lost. In other cases, the user doesn't lose data, but needs to restart a job that they were in the middle of (for example, a long copy job).

Interactive install - We were surprised to see 31% of users interactively installing updates; of these 31%, approximately 20% have selected to automatically install, but they manually intervene anyway. WU provides a pop-up notification telling you when updates are available if you have selected to automatically install. The notifications are clearly capturing people's attention, so they click on the notification and interactively install the updates. But this is actually reinforcing an unintended behavior. If you signed up to get automatic updates, you really shouldn't need to bother interactively installing an update every time one is available. Most installs should occur silently in the background, and WU should notify you only for critical actions (for example, a pending restart). This also matches feedback from customers, who tell us they find the constant notifications to be distracting. Their expectation when they choose automatic updating is that updating will occur automatically. This seems to be a case where making sure people are in control of their PC experience actually resulted in too much information, and ultimately the price of being in control was a feeling of a loss of control.

With these lessons learned, we set about defining a better automatic updating and restart experience for Windows 8.

Solving the challenge around updating and restarts

The question for us on the WU team is always *"What is the best way to quickly update the PC while not being intrusive to the user?"* Turns out, this is a hard question to answer, and there is no one simple answer.

The challenge we faced was to find the balance between updating with speed and giving notice to the user for upcoming restarts. Clearly, updating and securing the PC before vulnerabilities can be exploited is just as important as it ever was. However, we also want to deliver a better experience around handling restarts and avoiding data loss without compromising our goal of timely updating.

To this end, the guiding principles we used to design the experience were

- The automatic updating experience is not intrusive to users but keeps them aware of critical actions
- Minimize restarts and make them more predictable
- Continue to keep the PC and the ecosystem up-to-date and secure in a timely manner

Windows Update and handling restarts on Windows 8

Based on these principles, we made the following improvements to the Windows 8 updating experience.

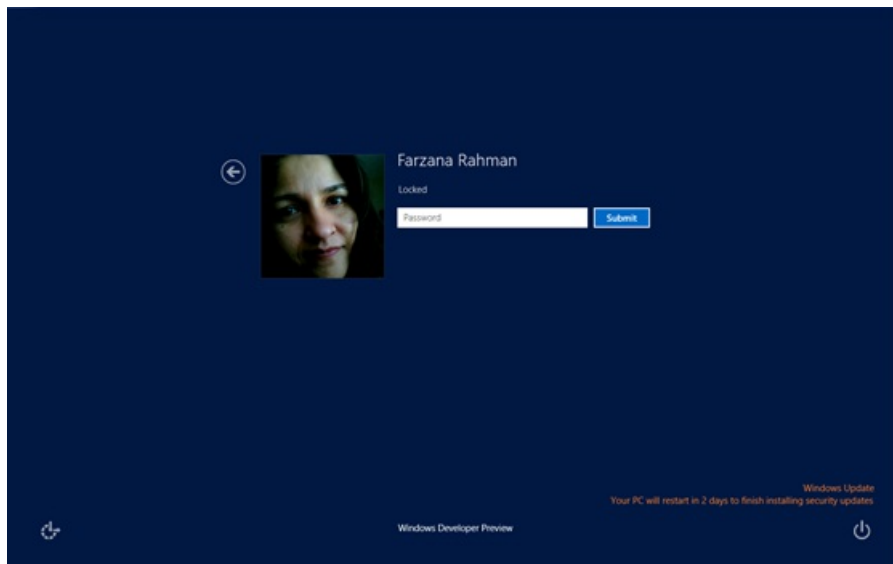
WU will consolidate all the restarts in a month, synchronizing with the monthly security release. This means that your PC will only restart when security updates are installed and require a restart. With this improvement, it does not matter when updates that require restarts are released in a month, since these restarts will wait till the security release. Since security updates are released in a single batch on the second Tuesday of every month, you are then getting essentially one restart a month. This simplification helps in three ways: it keeps the system secure in a timely manner, reduces restarts, and makes restarts more predictable.

There is one exception to the rule to wait for the monthly security release, and that is in the case of critical security update to fix a worm-like vulnerability (for example, a Blaster worm). In that case, WU will not wait, but will go ahead and download, install, and restart automatically. But this will happen only when the security threat is dire enough.

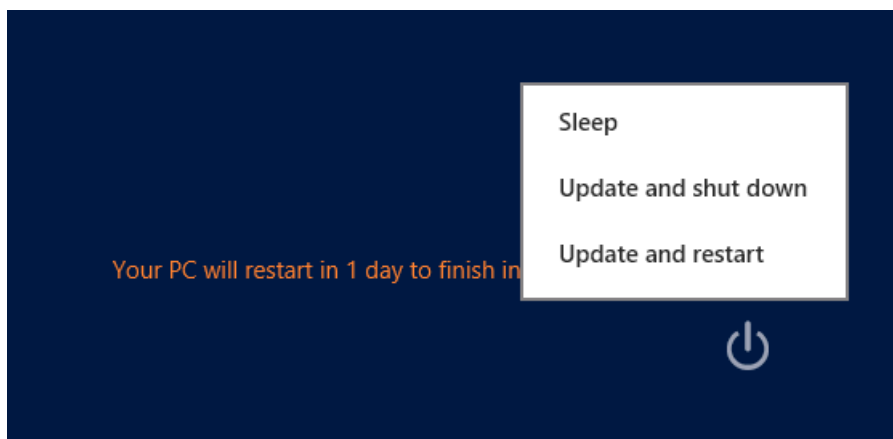
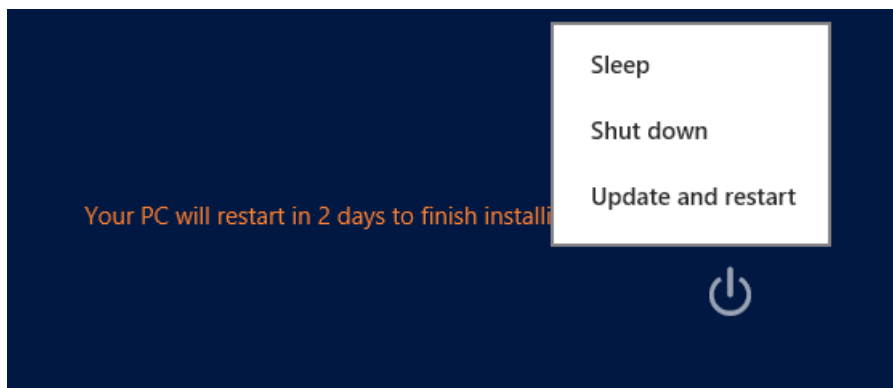
WU notifies you of any upcoming automatic restart. Let's assume that WU has already detected, downloaded, and installed security updates, and now requires a restart. Windows Update will notify you of an upcoming automatic restart through a message on the login screen that will persist for 3 days. Because the majority of update activity occurs in the first three days of the release of each update [see Figure 1], we wanted to give you 3 days to allow you to restart at your own convenience. You would restart by selecting "Update and shutdown" or "Update and restart" on the login screen, or by going to Windows Update in the Control Panel. You will no longer see any pop-up notifications or dialogs about pending restarts. Instead, the message appears in a more visible and appropriate place (the log-in screen). The use of the login screen has become ubiquitous even in home environments, as more and more machines become portable.

Here is a timeline view of that experience:

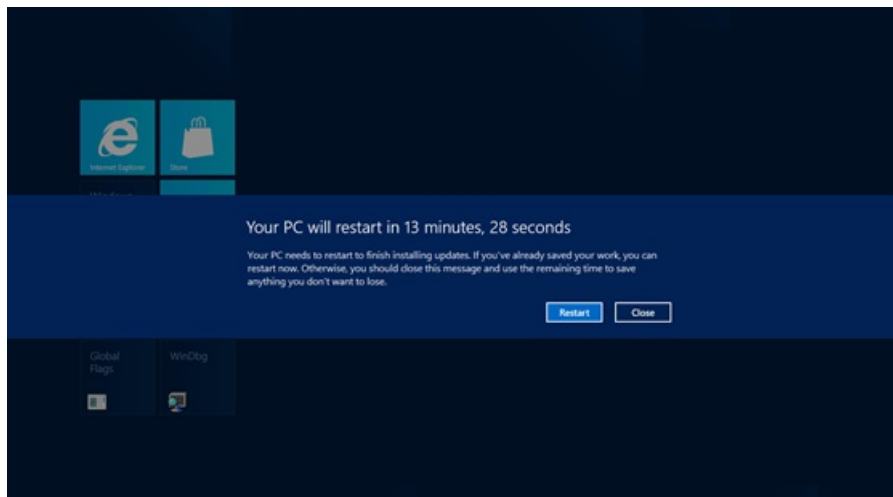
1. A message about the upcoming restart is shown in the login screen for three days or until the PC is restarted (whichever is sooner). This means you now have three days to restart the PC at your convenience. All you need to do is see the login screen once in 3 days to see the message about the upcoming restart and by default the lock screen will appear after 15-minute idle timeout.



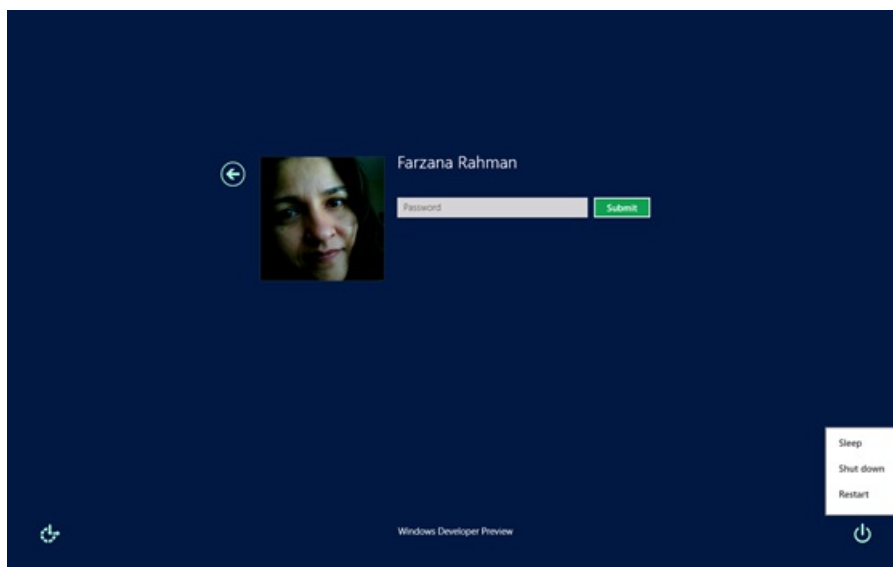
2. In addition to the restart notification on the login screen, the Power options on the lock screen will change to “Update and restart” immediately after the update occurs, and will include “Update and shutdown” on days two and three, to make the message even more apparent to you. This allows you to restart your PC at your own convenience.



3. If after three days, the restart still has not occurred, then WU will automatically restart your PC for you. In this case, the automatic restart will happen either at the end of the three-day grace period, or, to prevent data loss if WU detects that there are critical applications open at the end of the three-day grace period, it will wait to automatically restart the next time you login. I’ll address this behavior in more depth in the next section.



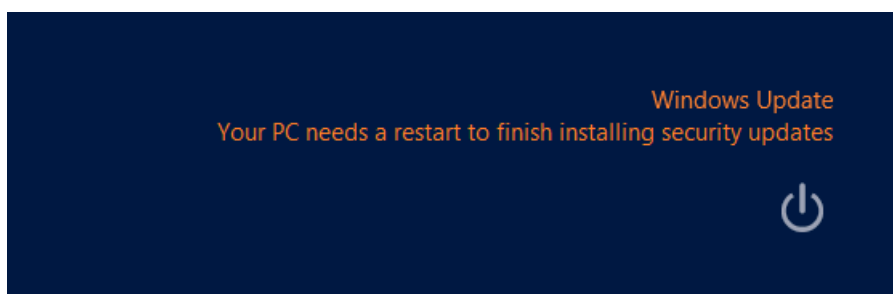
4. After the restart has occurred, the message on the login screen will go away and the power options will revert to the original choices. We know people would like Windows to automatically log in after the restart, but we strongly advise against doing so, given the potential security issues with this configuration.



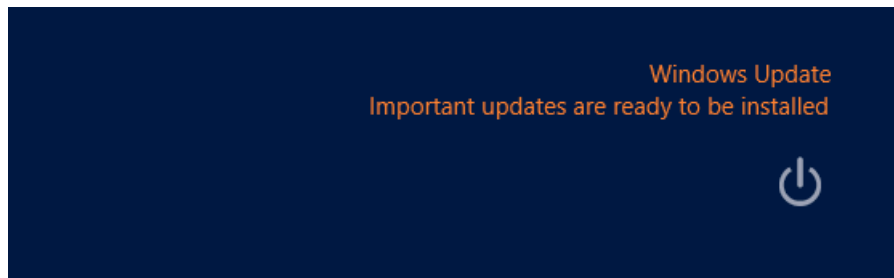
Delay the automatic restart if there is potential of losing user data. If the PC has hit the three-day deadline and still needs an automatic restart, WU will only automatically restart the machine if there is no chance of losing the user's data. That means, if you are not at your PC (i.e. it is locked), if you have applications running in the background, or if there is potentially unsaved work, WU delays the automatic restart until the next time you come back to your machine and log in. At log-in, you will be asked to save your work, and you'll see a warning that the machine will be restarted within 15 minutes.

Ensure minimal interruption to user activity. Having a restart notification or dialog pop up in the middle of an important presentation, a game or a movie is not a pleasant situation, to say the least. When attempting to automatically restart the PC, if you are in presentation mode, playing a game, or watching a movie full-screen, WU detects this state, and delays the automatic restart until the next available opportune moment or the next time you log back in to the PC.

The experience for business users. For PCs in an enterprise setting, if no policy has been set by the IT administrator, the updating experience is exactly the same as it is for home users. However, an IT administrator can set a policy to prevent auto-restart after automatic installs (just as in Windows 7). If they set this policy, there will be no three-day countdown and no automatic restart. Instead, users will see a message on the login screen indicating that the PC needs to be restarted, and the message persists until the restart occurs. This informs users that a restart is required while keeping them in control of when to restart.



The experience for users in “notify mode.” I also want to address the experience for users who have chosen to be notified before downloading or installing updates (5.82% of the WU user base from Figure2). For a user in this “notify mode,” a message will be shown on the login screen. If you choose to be notified before downloading updates, you will see the login screen message saying “Important updates are ready to be installed” when updates are ready to be downloaded. If you choose to be notified before install, you will see the same login screen message after updates are downloaded, but before they are installed. In either case, you won’t see the message about a pending restart on the login screen since installation is not automatic.



Cumulatively, these improvements help us achieve the balance we are striving for with Windows Update - keeping the PC (and PC ecosystem) up-to-date, without an intrusive experience.

What about updating 3rd party applications?

Lastly but not the least, I want to address the feedback from users who would like WU to update their 3rd-party applications. People clearly find the experience with multiple updaters on the system less than optimal (and we agree!) Each application updater gives you a different experience, you have to remember to go visit each updater to install updates, you never know when or how updaters will run and what they might do, and so on. People would like one updater for the entire system. On the other hand, users have also told us that they trust the quality of updates distributed by WU and hence are comfortable with choosing to automatically update their systems. We would not want to do anything that might reduce trust in the system by encouraging people to take on this management task manually and exposing their PCs to potential vulnerabilities for even short times.

Through WU and the “Microsoft Update” option (opt-in) we also offer updates for Microsoft products and for 3rd-party device drivers, with a common set of setup tools for each. All of these updates are carefully screened, and must adhere to the Windows conventions for updates regarding rollback and recovery, and overall system impact. As an example, drivers we publish through Windows Update go through tests run by the [Windows Logo Program for Hardware](#), which after validating the updates, signs them for authentication. And, we are continuously working to improve the validation system, to deliver better and higher quality drivers. The wide variety of delivery mechanisms, installation tools, and overall approaches to updates across the full breadth of applications makes it impossible to push all updates through this mechanism. As frustrating as this might be, it is also an important part of the ecosystem that we cannot just revisit for the installed base of software.

However, as we discussed at the [//build/](#) conference, the new Windows Store will provide a one-stop shop for (free and paid-for) Metro style apps, with an integrated update service to help ensure apps are maintained in a consistent manner. Because of the vetting process for these apps and the commitment from developers to deliver value to customers, we’re able to bring you this improvement as well. We’ll have more on this topic in future posts as soon as the Store becomes available to you for public testing.

Looking forward to your feedback.

Farzana

Improving the setup experience

Steven Sinofsky | [2011-11-21T12:00:00+00:00](#)

Installing Windows is a complex operation that provides an incredibly unique capability—the ability to run a new version of Windows on a vast array of hardware configurations and combinations that were designed with no knowledge of a future Windows, even a version with substantial re-architecture of the Kernel. While most people do not experience the full code path of setup/upgrade (because they buy new PCs and choose to get a new version of Windows that way), even orchestrating the new PC “out of box experience” (OOBE) is a complex technical challenge. Our aim in improving setup is to reduce the time from start to finish so that customers can get to Windows and use the full power of Windows to further customize and ultimately enjoy their new Windows experience. This post was written by Christa St. Pierre on our Setup and Deployment team.

—Steven

(Note, we’re taking a break for the US Holiday)

Setup is something that gets a lot of attention from us in any Windows release. It needs to just work reliably across a huge number of variations of hardware and software. This is true whether you are upgrading your own laptop, or you’re an IT pro who is migrating 10,000 desktops in an enterprise using broad deployment tools. For Windows 7 our main focus was on improving successful install rates, and we did a lot of work to improve reliability and deal with many tough (but relatively rare) cases that had caused problems in setting up earlier versions of Windows. This work gave Windows 7 a more reliable setup experience than in any previous Windows release, as measured by lab testing, customer support incidents, and setup telemetry.

For Windows 8, our goal was to continue to improve reliability while also improving the installation experience and raw performance. Not only did we want it to be rock solid, but also faster and easier to use.

A big challenge

Although millions of people choose to upgrade their existing PCs, most people choose to get a new version of Windows preinstalled on a new PC. In the past that often had to do with increasing system requirements in new Windows releases, and the need to purchase new PCs with more power to run the new version. With Windows 7 however, we made a commitment to work on many more existing PCs by keeping system requirements low and maintaining compatibility. We’ve continued that commitment with Windows 8, so many of you with existing PCs can simply upgrade. Looking just at Windows 7 customers, there are currently more than 450 million PCs that will be able to run Windows 8, but we expect that many systems running Windows Vista and even Windows XP will also be eligible.

Support for these PCs running different Windows versions is a big challenge in terms of testing all possible upgrade paths, languages, service packs, architectures, and editions. When you think about it, it is a rather remarkable achievement that hardware designed for one OS can be supported on an OS that did not exist when the hardware was created, especially considering that connecting hardware to software is a fundamental role played by the OS.

There are always complexities involving hardware support. Sometimes PCs are equipped with peripherals that require updated drivers for Windows 8, and in other cases, for any number of reasons, a PC maker decides that a particular model or configuration is not supported on a new version of Windows. There are also complexities in getting software to work seamlessly upon upgrade, particularly utilities that hook into the lowest levels of Windows such as anti-virus, disk format and defrag, or virtualization. While we have a massive test and ecosystem effort, ultimately the final say on support on a new version of Windows for a PC, peripheral, or software package is determined by the maker of that product. Our commitment to keeping things running and bringing forward software is industry leading and continues with Windows 8. At one recent team meeting, a member of our team showed Windows 8 running Excel version 3.0, which is the 16-bit version of Excel from 1990!

Perceived as “difficult”

During planning for Windows 8, we wanted to hear from customers who chose not to upgrade to Windows 7 even though their PCs would run it. In 2010 we commissioned a study of how people make PC purchase decisions, and talked to customers in three global markets to find out more. While the list of reasons as to why a customer chose not to upgrade varied by market, we have received notable feedback that upgrading the PC was perceived as difficult. So even though many customers *wanted* to upgrade, the current setup experience might be something that just wasn’t easy enough to make them feel confident in doing so.

Different customer needs

Hearing that some customers think it is too difficult really highlights the fact that we have many different customer needs we need to fulfill with setup. Most customers who buy a Windows upgrade from a retailer just want it to be fast and easy, but a few also want to be able to do some more complex things, such as setting up in a multi-boot configuration. And of course, we also have the IT Pro customers, who need to take full control over configurations, install from network as well as media, and add customizations to the setup image. The advanced user’s needs are a lot like those of the IT Pro, both because they require more fine-tuned control and because it’s hard for us to predict exactly which controls they may want to manipulate. For this reason, we have not created a “super advanced setup” mode, but we encourage people who want to create unattended setup configurations for home or work to use our standalone deployment tools. In Windows 7, we provided a [Windows Automated](#)

[Installation Kit](#), and in Windows 8 we have enhanced that with additional tools in the [Windows Assessment and Deployment Kit](#), which is available for download to [MSDN subscribers](#).

For this post, I'll talk mostly about the interactive GUI setup experiences, since that's where we have the most changes. We sought to maintain very high backwards compatibility with existing unattended installation configurations that IT Pros or advanced users have spent time on for Windows 7, so you can expect those to work consistently for Windows 8 as well, without having to start over. So rest assured that your custom deployments continue to be fully supported as before.

Streamlining the end-to-end experience

Leaving aside automated installations and just looking at the typical GUI scenarios, we still wanted to serve two distinct customer groups in the setup user experience:

- People who want an easy way to upgrade to the new release with an absolute bare minimum of hassle
- People who want to do a clean install, and want more control of setup options, disk layout, and partition configuration

The way we approached these needs was based on the realization that the first group typically runs setup in the UI of their current Windows OS (i.e. they launch it like an app), while the second group typically runs setup from boot media. So, rather than trying to rationalize two fairly different experiences and customer requirements, we chose to maintain two setup user experiences: a **streamlined setup** that you reach by running an .exe from the DVD or via web delivery, and an **advanced setup** that runs when you boot off of a DVD or USB key. The streamlined setup is a new experience, optimized for ease-of-use, upgrades, and web delivery via download. Advanced setup is the home of all things familiar to the advanced user, including full support for unattended installation, partition selection, and formatting. Under the covers they share all of the same setup engine components. So both experiences benefitted from our ability to focus on a common codebase for performance and reliability enhancements.

Shifting towards web delivery methods

Before going into the detail on the user experience changes, there's one big change that is important to call out. In the past, if you wanted to buy an upgrade for Windows, it involved purchasing a boxed product from a retail outlet, taking it home, (sometimes being infuriated while trying to open the box,) and inserting a DVD. However, buying boxed software is quickly becoming the exception rather than the rule, with more and more software being purchased online as broadband penetration increases and large-size media downloads become more common. While we will continue to offer boxed DVDs, we are also making it easier than ever to purchase and install online. This includes starting the setup experience online as well, and having one continuous integrated experience from beginning to end. There is also one big advantage that is a favorite of mine. With our web setup experience, we actually "pre-key" the setup image that is downloaded to a unique user, which means that you don't have to type in the 25-digit product key when you install!

Streamlining - Reducing repetition and integrating experiences

More than 20 million customers downloaded and ran the Windows 7 Upgrade Advisor during the first six months of availability. Many customers also ran Windows Easy Transfer during this same time period. A reasonable (and often recommended) installation experience for Windows 7 followed a flow like this:

1. Download and install Windows 7 Upgrade Advisor
2. Run Windows Upgrade Advisor
3. Run Windows Easy Transfer to save files and settings
4. Run Windows 7 setup and clean install
5. Run Windows Easy Transfer to restore files and settings

This end-to-end experience included 4 different web and client experiences and required the average customer to walk through 60 screens to complete. The primary reason for the high screen count was the repetition of information. We can visualize it something like this:



A common Windows 7 installation experience: Upgrade Advisor, Windows Easy Transfer, and Setup

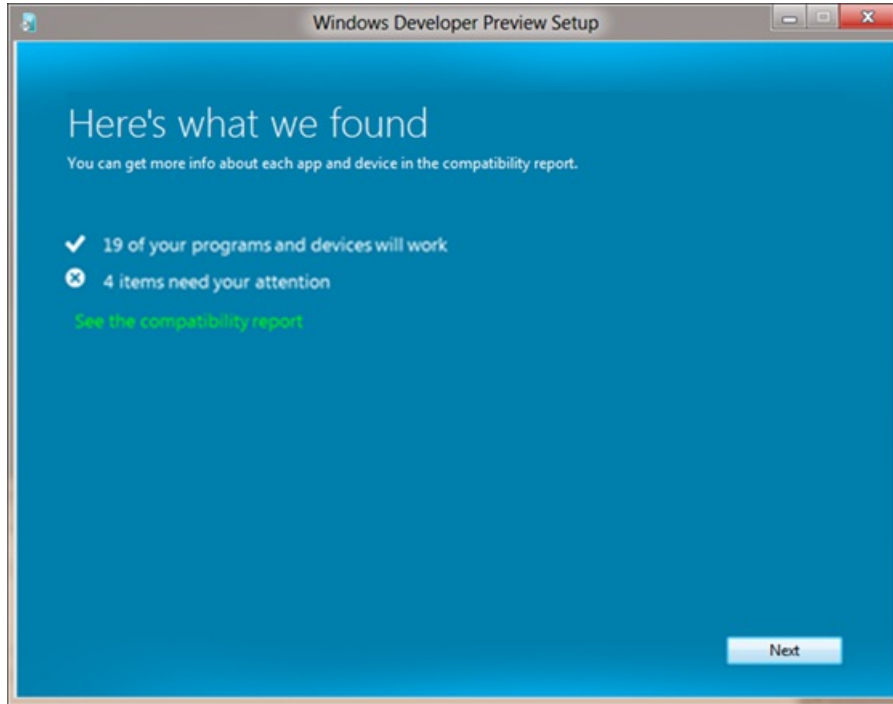
In Windows 8, rather than having Upgrade Advisor, Setup, and Windows Easy Transfer as separate apps or features, we've folded them together into one fast and fluid experience in which we first determine if your PC, apps, and devices will work in the new OS, note which things you want to keep (apps, files and/or settings), and then install the new OS.

We've also added the capability for setup to resume automatically after certain actions (such as resolving a blocking compatibility problem), which in the past would have required restarting setup again from the beginning.

Here's what to expect when you launch the new setup experience from the web :

Determining compatibility

The first thing we do is scan the PC to determine compatibility, resulting in a summary report such as this one:



Windows 8 setup compatibility summary

It provides information on the apps and devices that will work in Windows 8, those that won't work, and any other system information that is useful to know when determining whether or not to purchase and install Windows 8. A detailed compatibility report is also available if you want to print or save the information, or desire more detail about what to expect once you get to Windows 8, including which apps or devices will require updates.

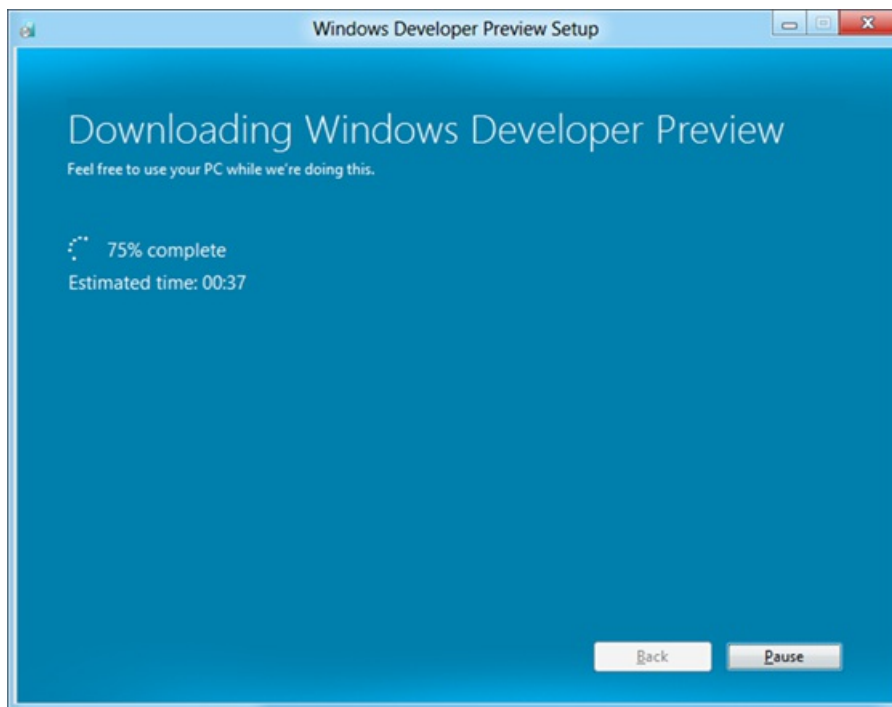
The compatibility data behind the report covers hundreds of thousands of applications and devices, including retail software, OEM preinstalled software, and peripherals. If an application or device ran on Windows 7, our goal is that it should run on Windows 8 too, but in some cases it may require an update or other support from the OEM or vendor. Some applications also have custom installation logic – installing certain components or settings depending on the OS you're upgrading from (this is particularly true of system utilities and software that is tightly connected to hardware and peripherals). You may need to uninstall and reinstall these types of apps. (This is also a reason to be careful of 3rd party “app mover” applications, which claim to move apps from one OS to another, as the end result can be unpredictable or broken.) As a reminder, the best drivers for any system are the ones available directly from the PC manufacturer for embedded hardware and from the device manufacturer for peripherals.



Detailed compatibility report

Downloading Windows 8

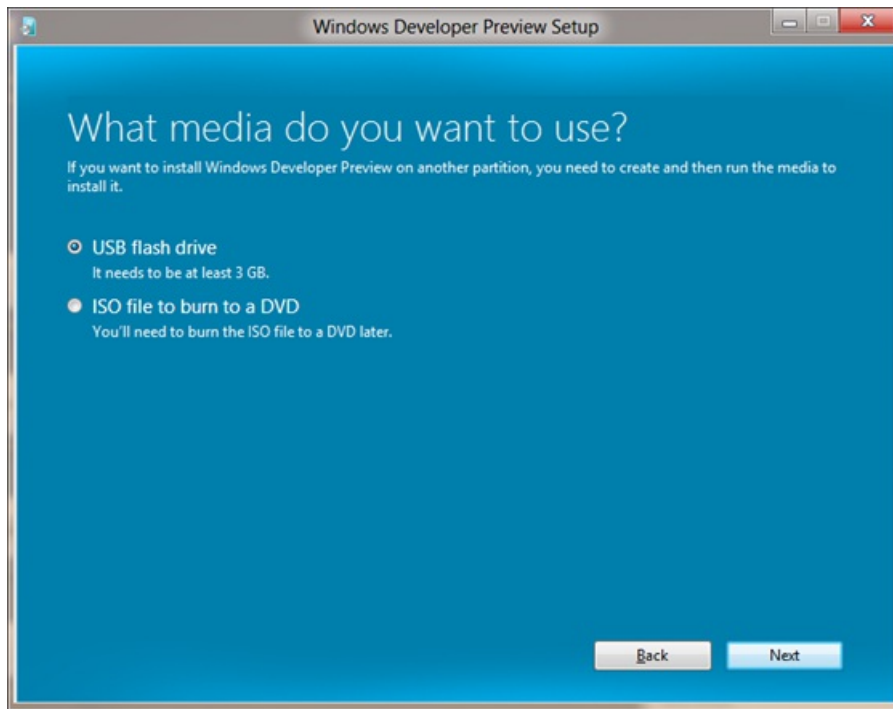
Next, an integrated download manager provides time estimates, data validation, the ability to pause, resume, and re-download only parts of the file if something goes wrong. Additionally, because we have already scanned the PC to determine compatibility we know which version of Windows 8 to download – eliminating the need to ask questions such as which language or OS architecture to choose.



Downloading the Windows image from the web

Continuing with installation or creating bootable media

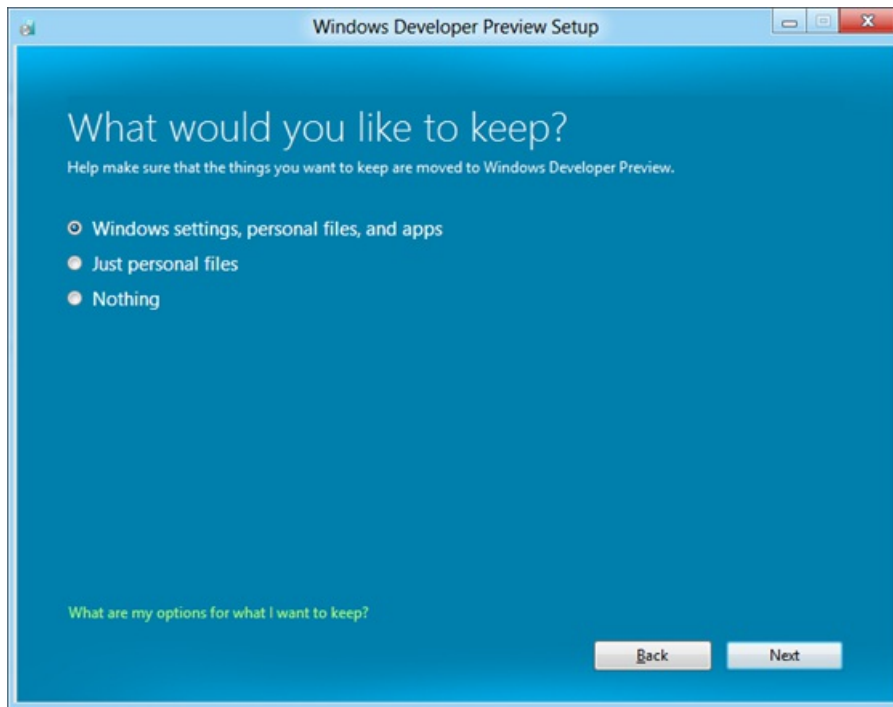
Once the download is complete, you are presented a choice to continue the installation, or install on another partition. The latter option takes you to advanced setup, and allows you to save an ISO or create a bootable USB drive before completing other advanced setup options. (This is the option you'll need to choose if you want to dual boot, for example.)



Creating bootable media from web-based setup

Choosing what to keep

Next is the upgrade choice. You can choose to keep all, some, or none of your personal data depending on the OS you're upgrading from, and your personal preferences.



Windows 8 setup options for upgrade and migration

The “Windows settings, personal files, and apps” option is akin to the existing “upgrade” option in Windows 7 and Windows Vista, where an in-place upgrade is performed over the current OS, retaining the apps that were previously installed as well as settings and user files on disk.

The “Just personal files” option is a new functionality, which allows you to get a clean install, but still keep your data without a separate tool such as Windows Easy Transfer.

The upgrade options that you might see in the screen above depend on which version of Windows you are upgrading from. Here’s the list of what you can migrate based on your currently installed version of Windows:

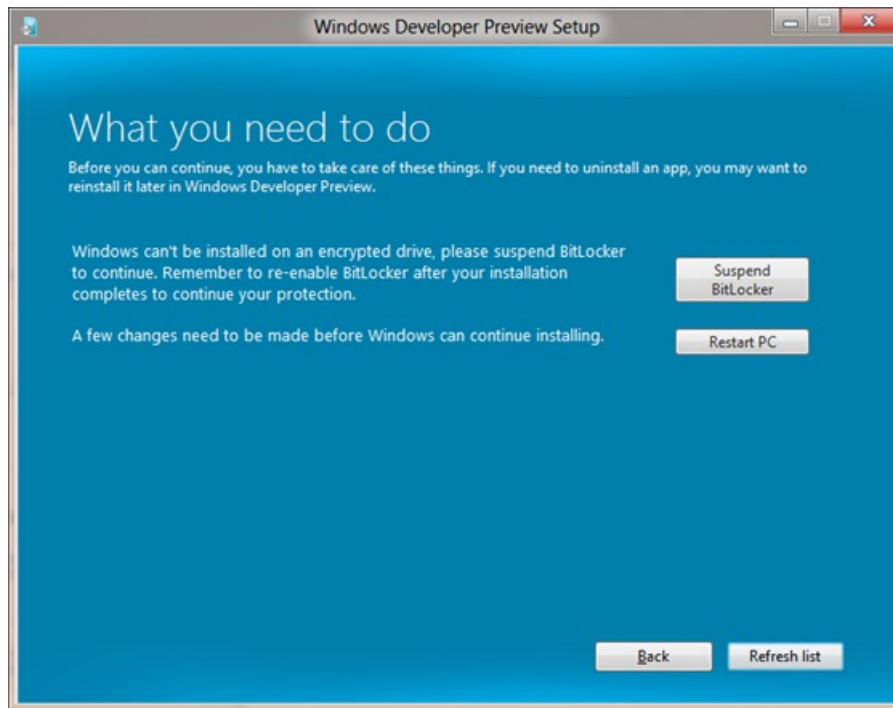
<i>You can transfer these...</i>		<i>When upgrading from...</i>	
	Windows 7	Windows Vista	Windows XP

Applications	x		
Windows settings	x	x	
User accounts and files	x	x	x

Clean install is supported across all versions.

Resolving blocking issues

Often you may need to make changes to your PC before you can continue with the installation. Common requirements include things like uninstalling an application, freeing up disk space, or suspending BitLocker. When encountering this scenario in Windows 7 setup, you would simply see a warning message and then you'd have to exit, take care of the clean-up that was listed, and resume setup again from the beginning. In Windows 8, most items listed in the actionable compatibility report (shown below) include a button to help you directly resolve the blocking issue. For example, if an app needs to be uninstalled, clicking a button in this report automatically launches the uninstaller for that particular app. Once the app is uninstalled the report automatically refreshes, and setup continues without having to start again.

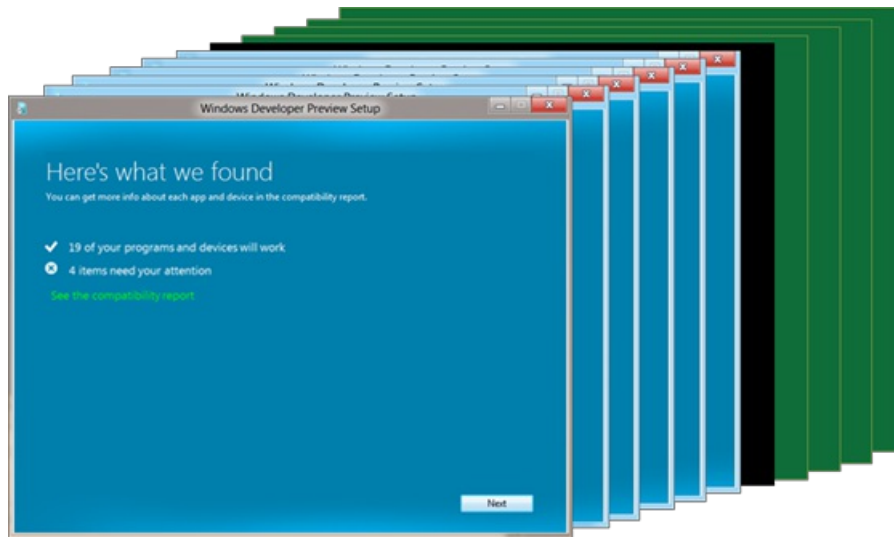


Resolving blocking issues directly from the setup experience

This also works in the case where a reboot is needed. For example, if the blocking app requires a reboot after it is uninstalled, setup will resume from where it left off before the reboot.

The result

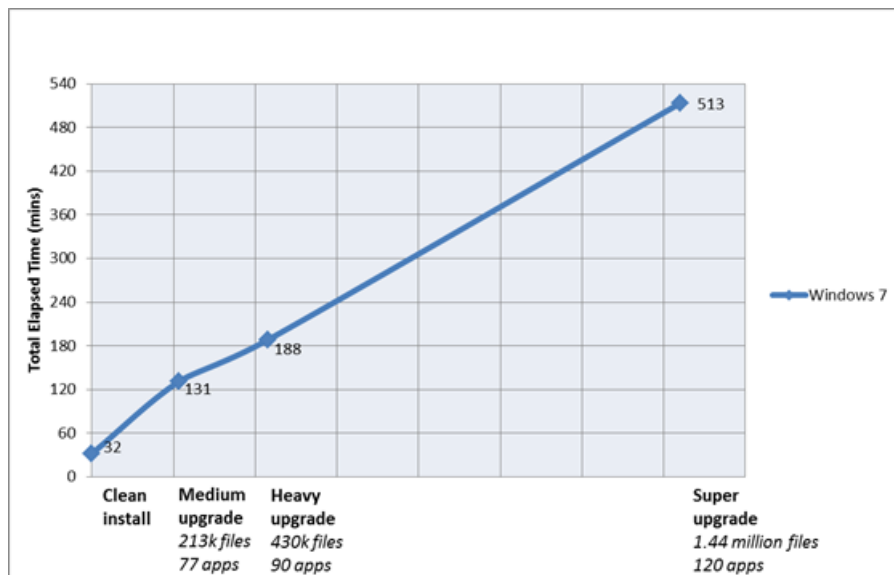
The scenario we presented at the beginning that included **four** different wizards and up to 60 screens in a Windows 7 upgrade can now be accomplished in **one** end-to-end experience and as few as **11** clicks, an improvement of **82% fewer clicks** in Windows 8. The exact number of steps you need to take to complete the installation varies based on your existing OS, migration choices, install method, and number of blocking issues you need to resolve to get the PC ready for installation, but the experience is greatly simplified for everyone. We accomplished all of this with no loss of functionality or customization—we simply streamlined the existing experience.



The typical Windows 8 installation experience, with integrated advisor, migration, and setup

Improving upgrade performance

If you had a large number of files on your system, you may have seen that installation times in Windows 7 didn't scale very well. In fact, as you can see in the diagram below, the more user files there are on a PC being upgraded (regardless of the size of the files) the longer the upgrade takes to complete.



Windows 7 time to upgrade in relation to the number of files on a PC

Note: Time in this graph represents time to complete the upgrade once the installation is initiated, and does not include time to download or read files from media.

The reason for this is that in Windows 7, the upgrade process preserved the customer's applications in the Program Files folder and their files in the Users folder by moving each file to a transport location (so that the original folders can be deleted to make way for the newer installation), and then moving them back again to complete the installation. With music and photo collections, it's not unusual to have hundreds of thousands of files, so even relatively fast move operations can really add up.

To address this in Windows 8, we have made several modifications to the upgrade engine to reduce the impact on upgrade times.

Moving whole folders

In the past, each file that was preserved across upgrade was moved individually. In Windows 8, instead of moving things file-by-file, we move entire folders, drastically reducing the number of file operations required. This goes a long way towards shrinking the variation in upgrade times due to the amount of data the customer has on the machine.

At a high level, the logic for whether or not we need to move a given folder is:

- Every file in the folder (and its sub folders) is preserved (there are no exclude rules removing some of the files, for example).
- The entire folder is placed on the target OS unchanged.

- The target destination doesn't already exist (i.e. we don't have to merge an existing folder on the destination OS with one from the source OS). There are a few exceptions to this rule however – for example, every folder has a desktop.ini file, but we have logic that allows the source folder to overwrite this file, as in many cases the file is only a cache and can be regenerated.

Simplifying the transport

In Windows 7 the transport (this is the place where we store the files and settings being preserved between the old and new operating systems) was comprised of two folders: “Windows.~q” and “Windows.~tr”. In Windows 8 we have simplified this to just one folder. We have repurposed the “Windows.old” naming convention for consistency with clean install (which creates a “Windows.old” folder containing the previous OS in order to be able to roll back should the installation fail). Merging the transport folders into the single Windows.old folder speeds up the upgrade process, as it removes the need to move files between the ~tr and ~q folders.

Switching to hard links

In upgrades to Windows 7, files were moved between the old OS, the transport, and Windows 7 by using file move operations. In upgrades to Windows 8, we use hard link operations instead. This means we can link to the actual data on disk in the transport location without having to physically move the file, which has a significant performance gain. And if something goes wrong with setup and we have to roll back, we just need to delete the hard links, and the files are completely unaffected on disk.

Removing the down-level gather phase

In Windows 7, the files and settings to be preserved across the upgrade were calculated while the previous OS was still running. The registry values and data collected by our upgrade logic were also gathered while running on the old OS. The content of the files was then gathered offline during the Windows Pre-Installation Environment (Windows PE) phase in order to avoid file-in-use issues.

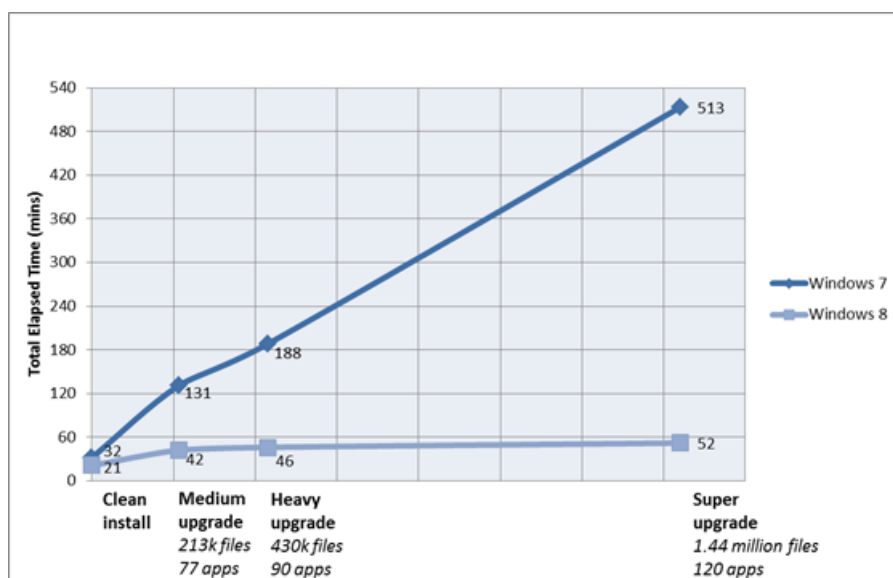
Most of this work has been removed in Windows 8. The gather rules no longer run during upgrade; instead, we just move the following folders into Windows.old when the PC is offline:

- Windows
- Program files
- Program files (x86)
- Users
- Program data

This means that during the “apply” phase of upgrade (once we are running in Windows 8), everything we need to preserve can be extracted from the Windows.old folder (as we touch no other folders during the upgrade), eliminating the need for a gather phase. Speaking of the Windows.old folder, we have also added a new feature that automatically deletes that folder 4 weeks after a successful install, so you don't have to worry about removing it. Of course, you can still use the Disk Cleanup tool to remove it immediately if you prefer.

The result

In our labs we compared Windows 7 upgrade times to upgrading to a recent Windows 8 build, and found that the variation in upgrade times based on number of files has been virtually eliminated, as shown in the diagram below.



Windows 7 vs. Windows 8 time to upgrade

Note: Time in this graph represents time to complete the upgrade once the installation is initiated and does not include time to download or read files from media.

Additional optimizations for web delivery

As I mentioned above, Windows 8 setup has been designed for online delivery, in addition to the local delivery from a DVD or USB drive. While downloading Windows has been possible in the past, it was primarily a physical media experience made available for download. In Windows 7 upgrades, for example, two copies were created of the download content on the customer's drive—the compressed download and the extracted contents—requiring ~5 GB. This could be very problematic on space-constrained systems. Additionally, both the compressed and extracted download contents remained on disk, even after a successful installation.

For Windows 8, in addition to the setup experience improvements for web delivery, we also optimized other aspects. Our goal was to minimize the time it takes for the download to complete, verify the integrity of the bits that are downloaded, minimize disk space requirements, and ensure a resilient download experience for the customer. The two main areas of improvement for Windows 8 are constructing optimized download packages, and making sure that downloading is flexible and resilient.

Constructing optimized download packages

The Windows 7 media layout for x86 consists of 874 files and 200 folders, with a number of redundant files both in the media and compressed within install.wim and boot.wim. To efficiently store and transfer the contents of installation media, we typically use ISO files. For example, an ISO created from the x86 client media is 2.32GB. In order to optimize for download in Windows 8, we take the required subset of files for the specific version of Windows being downloaded. After eliminating duplicates and compressing resources, the single-file size is 2.10GB (as compared to 2.32GB), a savings of 9.5%. After this optimized package is created we compress it using an improved compression algorithm specifically for Windows 8 setup, which provides an additional 28% savings. In this example (using the Windows 7 x86 ISO) the size of the download would be reduced from 2.32GB to 1.51GB.

Downloading is flexible and resilient

The download manager included in Windows 8 setup downloads the optimized package containing the new OS and reconstructs the layout required to run through the install process, without leaving duplicate files on the system. The download manager leverages the Background Intelligent Transfer Service (BITS) as the default transport protocol to transfer files from the Internet to the local machine and provides the ability to pause, resume, and restart. It verifies the bits that are downloaded in 10MB increments. If verification fails for a particular increment, the download manager has the ability to re-request only that specific block of data without having to restart the entire download.

The result

In Windows 8, customers do not have to install a separate download manager, mount the ISO to begin the installation, check the hash of the file for verification post-download, manually clean up unneeded files, or restart a download from the beginning should connectivity be interrupted. Setup takes care of all of these steps automatically, providing a fast, resilient, and easy setup experience. And again, this is true whether you just want to run a quick upgrade on an existing installation, or to create boot media for an advanced setup experience – either with GUI or unattended.

Advanced setup for IT Pros

I said at the beginning that this post wouldn't go into a lot of detail about our automated installation using the Assessment and Deployment Kit (formerly the WAIK) but I know that many readers of this blog may be interested in learning more about it. So, here are a couple of handy configurations that will make it easy for you to customize a bootable USB drive (that you can create as part of our download experience) and automate your installation. For full details about all of the configurations that are possible, check out the [Windows 8 ADK](#) (for MSDN subscribers only). If you aren't an MSDN subscriber check out the [Best Practices for Authoring Answer Files](#) article on TechNet. These best practices still apply for Windows 8 and all of the tools and documents referenced in the article are available in the Windows 7 WAIK or the Windows 8 ADK.

Here's a video demonstrating an advanced setup from a USB flash drive. Note that this experience is not yet available in the Developer Preview build, but will be there in the final release.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

Key injection

You may have noticed in the screenshots and video above that none show the familiar "type your product key" experience. In the web setup scenario you won't see those screens because of key injection from the server, but if you boot from media and choose to do an advanced setup, you'll have to type it in. With "unattend" settings though, you can do your own key injection, so that you can skip this step. This is handy if you're reinstalling after changing some system components or replacing a drive.

The specific setting that you will need to configure to do this is the [ProductKey](#) setting.

Here's a sample:

```
<UserData>
  <ProductKey>
    <Key>12345-12345-12345-12345-12345</Key>
    <WillShowUI>Never</WillShowUI>
  </ProductKey>
</UserData>
```

Automating install

You can also automate other parts of the experience so that you don't have to manually click through the screens. Here are a couple of other settings that are useful when automating your install:

You can choose the [UI language](#) used for Windows:

```
<InputLocale>0407:00000407</InputLocale>
<SystemLocale>de-DE</SystemLocale>
<UILanguage>de-DE</UILanguage>
<UserLocale>de-DE</UserLocale>
```

There are additional settings for every UI choice, so you can script it to the point that the install is essentially hands-off from start to finish.

Dual boot configuration

Unattend is also useful when you want to automatically configure the system for booting multiple operating systems. You could do this all manually in the Advanced Setup GUI and BCD configuration, but why do that when you can script it? The unattend framework is very flexible and you can instruct Setup to format, create, or modify partitions on the PC's disk(s).

Using the [DiskID](#) setting you can create and modify partitions. You can then specify the [PartitionID](#) setting to install to a specific partition—one that is different from an existing OS partition.

Here's a sample to install to a specific partition:

```
<ImageInstall>
  <OSImage>
    <InstallFrom>
      <Path> Z:\sources\install.wim </Path>
    </InstallFrom>
    <InstallTo>
      <DiskID>0</DiskID>
      <PartitionID>1</PartitionID>
    </InstallTo>
    <WillShowUI>OnError</WillShowUI>
    <InstallToAvailablePartition>>false</InstallToAvailablePartition>
  </OSImage>
</ImageInstall>
```

Where to save your unattend answer file

Once you have your answer file configured to your liking you can copy it to the root of your USB media. (Remember, if you use setup from the web, you can still create a bootable USB drive or save to an ISO.) You can also include the file at the root of the DVD media where you burned the ISO, if you chose that route instead. Even cooler, the unattend file doesn't even have to be on the installation media. In fact you can place the unattend file at the root of a USB flash drive, plug in the flash drive before starting setup, and setup will automatically find it and use it.

An improved setup experience

With Windows 8 setup we have greatly improved both speed and ease of use, while still retaining all of the advanced setup functionality that many customers will demand. We have integrated what was once many separate steps for people to perform when preparing and starting their setup into a streamlined user experience, with a fast and reliable setup engine under the hood. Customers who choose to install Windows from an online source will have a greatly improved experience over what we've delivered in the past, with smaller and faster downloads, as well as increased resiliency and control. We hope that you will find these improvements to be a great way to start your experience using Windows 8.

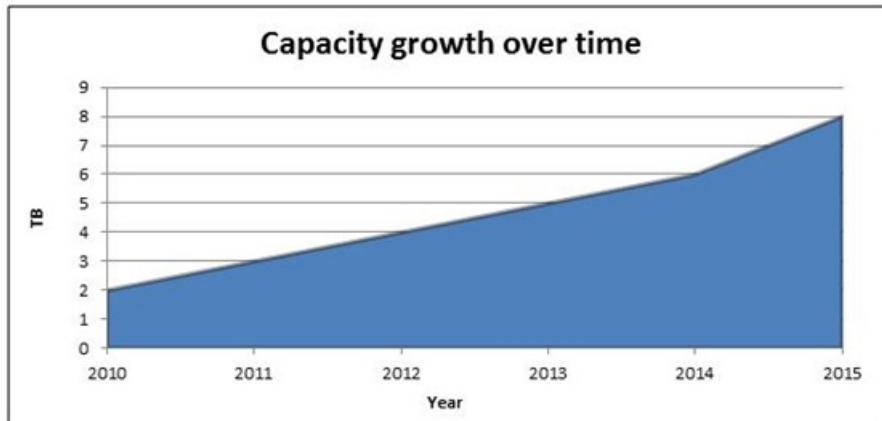
--Christa St. Pierre

Enabling large disks and large sectors in Windows 8

Steven Sinofsky | [2011-11-29T15:30:00+00:00](#)

One of the most basic services provided by an OS is the file system, and Windows has one of the most advanced file systems of any operating system used broadly. In Windows 7 we improved things substantially in terms of reliability, management, and robustness (for example, [automating completely the antiquated notion of "defrag"](#)). In Windows 8 we build on this work by focusing on scale and capacity. Bryan Matthew, a program manager on the Storage & File System team, authored this post.
--Steven

Our digital collections keep growing at an ever increasing rate – high resolution digital photography, high-definition home movies, and large music collections contribute significantly to this growth. Hard disk vendors have responded to this challenge by delivering very large capacity hard disk drives – a recent IDC market research report estimates that the maximum capacity of a single hard disk drive will increase to 8TB by 2015.



*Maximum capacity growth over time for single-disk drives
(Source: IDC Study# 228266, Worldwide Hard Disk Drive 2011–2015 Forecast:
Transformational Times, May 2011)*

In this blog entry, I'll discuss how Windows 8 has evolved in conjunction with offerings from industry partners to enable you to more efficiently and fully utilize these very large capacity drives.

The challenges of very large capacity hard disk drives

To start you out with a little bit of context, we will define “very large capacity” disk drives as sizes > 2.2TB (per disk drive). The current architecture in Windows has some limits that makes these drives somewhat tricky to deal with in some scenarios.

Even as hard disk drive vendors innovated to deliver very large capacity drives, two key challenges required focused attention:

- Ensuring that the entire available capacity is addressable, so as to enable full utilization
- Supporting the hard disk drive vendors in their effort to deliver more efficiently managed physical disks – 4K (large) sector sizes

Let's discuss both of these in more detail.

Addressing all available capacity

To fully understand the challenges with addressing all available capacity on very large disks, we need to delve into the following concepts:

- The addressing method
- The disk partitioning scheme
- The firmware implementation in the PC – whether BIOS or UEFI

The addressing method

Initially, disks were addressed using the CHS (Cylinder-Head-Sector) method, where you could pinpoint a specific block of data on the disk by specifying which Cylinder, Head, and Sector it was on. I remember in 2001 (when I was still in junior high!) we saw the introduction of a 160GB disk, which marked the limit of the CHS method of addressing (at around 137GB), and systems needed to be redesigned to support larger disks. *[Editor's note: my first hard drive was 5MB, and was the size of a tower PC. --Steven]*

The new addressing method was called Logical Block Addressing (LBA) – instead of referring to sectors using discrete geometry, a *sector number* (logical block address) was used to refer to a specific block of data on the disk. Windows was updated to utilize this new mechanism of

addressing available capacity on hard disk drives. With the LBA scheme, each sector has a predefined size (until recently, 512 bytes per sector), and sectors are addressed in monotonically increasing order, beginning with *sector 0* and going on to *sector n* where:

$$n = (\text{total capacity in bytes}) / (\text{sector size in bytes})$$

The disk partitioning scheme

While LBA addressing theoretically allows arbitrarily large capacities to be accessed, in practice, the largest value of “n” can be limited by the associated *disk partitioning scheme*.

The notion of disk partitioning can be traced back to the early 1980s - at the time, system implementers identified the need to divide a disk drive into several *partitions* (i.e. sub-portions), which could then be individually formatted with a file system, and subsequently used to store data. The Master Boot Record partition table (MBR) scheme was invented at the time, which allowed for up to 32-bits of information to represent the maximum capacity of the disk. Simple math informs us that the largest addressable byte represented via 32 bits is 2^{32} or 2.2TB. Of course, in the 1980s, this seemed a perfectly legitimate practical limitation to impose, considering that the largest consumer disk available then was a whopping 5MB and cost well over \$1500!

As early as in the late 1990s, system implementers recognized the need to enable addressing greater than the 2.2TB limit (among other requirements). A group of companies collaborated to develop a scalable partitioning scheme called the GUID Partition Table (GPT), as part of the Unified Extensible Firmware Interface (UEFI) specification. GPT allows for up to 64-bits of information to store the number that represents the maximum size of a disk, which in turn allows for up to a theoretical maximum of 9.4 ZettaByte (1 ZB = 1,000,000,000,000,000,000 bytes).

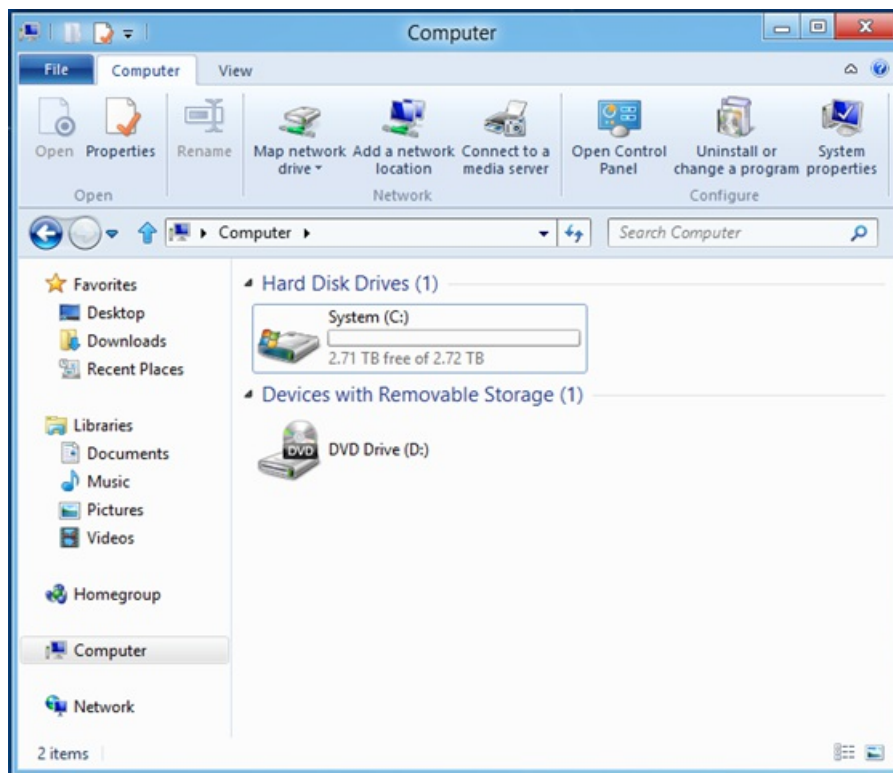
Beginning with Windows Vista 64-bit, Windows has supported the ability to boot from a GPT partitioned hard disk drive with one key requirement – the system firmware must be UEFI. We've already talked [about UEFI](#), so you know it can be enabled as a new feature of Windows 8 PCs. This leads us to the topic of firmware.

Firmware implementation in the PC – BIOS or UEFI

PC vendors include firmware that is responsible for basic hardware initialization (among other things) before control is handed over to the operating system (Windows). The venerable BIOS (Basic Input Output System) firmware implementations have been around since the PC was invented i.e. circa 1980. Given the very significant evolution in PCs over the decades, the UEFI specification was developed as a replacement for BIOS and implementations have existed since the late 1990s. UEFI was designed from the ground up to work with very large capacity drives by utilizing the GUID partition table, or GPT – although some BIOS implementations have attempted to prolong their own relevance and utility by using workarounds for large capacity drives (e.g. a hybrid MBR-GPT partitioning scheme). These mechanisms can be quite fragile, and can place data at risk. Therefore, Windows has consistently required modern UEFI firmware to be used in conjunction with the GPT scheme for boot disks.

Beginning with Windows 8, multiple new capabilities within Windows will necessitate UEFI. The combination of UEFI firmware + GPT partitioning + LBA allows Windows to fully address very large capacity disks with ease.

Our partners are working hard to deliver Windows 8 based systems that use UEFI to help enable these innovative Windows 8 features and scenarios (e.g. Secure Boot, Encrypted Drive, and Fast Start-up). You can expect that when Windows 8 is released, new systems will support installing Windows 8 to, and booting from, a 3TB or bigger disk. Here's a preview:



Windows 8 booted from a 3 TB SATA drive with a UEFI system

4KB (large) sector sizes

All hard disk drives include some form of built-in error correction information and logic – this enables hard disk drive vendors to automatically deal with the *Signal-to-Noise Ratio (SNR)* when reading from the disk platters. As disk capacity increases, bits on the disk get packed closer and closer together; and as they do, the SNR of reading from the disk decreases. To compensate for decreasing SNR, individual sectors on the disk need to store more *Error Correction Codes (ECC)* to help compensate for errors in reading the sector. Modern disks are now at the point where the current method of storing ECCs is no longer an efficient use of space, – that is, a lot of the space in the current 512-byte sector is being used to store ECC information instead of being available for you to store your data. This, among other things, has led to the introduction of larger sector sizes.

Larger sector sizes – “Advanced Format” media

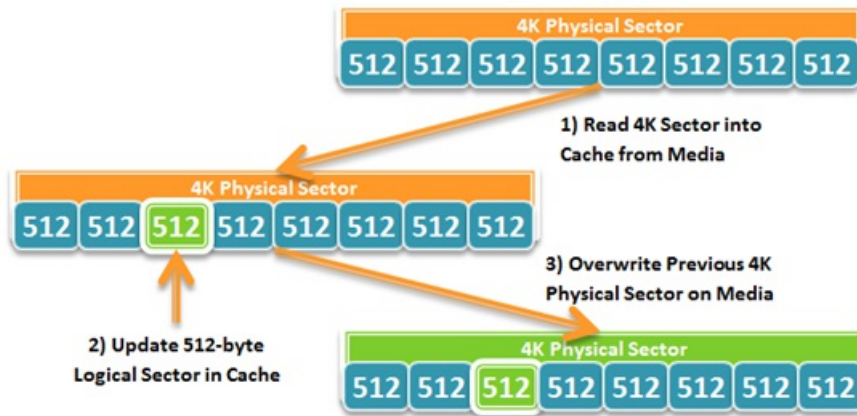
With a larger sector size, a different scheme can be used to encode the ECC; this is more efficient at correcting for errors, and uses less space overall. This efficiency helps to enable even larger capacities for the future. Hard disk manufacturers agreed to use a sector size of 4KB, which they call “Advanced Format (AF),” and they introduced the first AF drive to the market in late 2009. Since then, hard disk manufacturers have rapidly transitioned their product lines to AF media, with the expectation that all future storage devices will use this format.

Read-Modify-Write

With an AF disk, the layout of data on the media is physically arranged in 4KB blocks. Updates to the media can only occur at that granularity, and so, to enable logical block addressing in smaller units, the disk needs to do some special work. Writes done in units of the physical sector size do not need this special work, so you can think of the physical sector size as the unit of *atomicity* for the media.

As illustrated below, a 4KB physical sector can be logically addressed with 512-byte logical sectors. In order to write to a single logical sector, the disk cannot simply move the disk head over that section of the physical sector and start writing. Instead, it needs to read the entire 4KB physical sector into a cache, modify the 512-byte logical sector in the cache, and then write the entire 4KB physical sector back to the media (replacing the old block). This is called Read-Modify-Write.

Disks with this emulation layer to support unaligned writes are called 4K with 512-byte emulation, or “512e” for short. Disks without this emulation layer are called “4K Native.”



As a result of Read-Modify-Write, performance can potentially suffer in applications and workloads that issue large amounts of unaligned writes. To provide support for this type of media, Windows needs to ensure that applications can retrieve the *physical* sector size of the device, and applications (both Windows applications and 3rd party applications) need to ensure that they align I/O to the reported *physical* sector size.

Designing for large sector disks

Learning from some issues identified with prior versions of Windows, AF disks have been a key design point for new features and technologies in Windows 8; as a result, Windows 8 is the first OS with full support for *both* types of AF disks – 512e and 4K Native.

To make this happen, we identified which features and technology areas were most vulnerable to the potential issues described above, and reached out to the teams developing those features to provide guidance and help them test hardware for these scenarios.

Issues we addressed included the following:

- Introduce new and enhance existing API to better enable applications to query for the physical sector size of a disk
- Enhancing large-sector awareness within the NTFS file system, including ensuring appropriate sector padding when performing extending writes (writing to the end of the file)
- Incorporating large-sector awareness in the new VHDX file format used by Hyper-V to fully support both types of AF disks
- Enhancing the Windows boot code to work correctly when booting from 4K native disks

This is just a small cross section of the amount of work done to enable across-the-board support for both types of AF disks in Windows 8. We are also working with other product teams within Microsoft and across the industry (e.g. database application developers) to ensure efficient and correct behavior with AF disks.

In closing

NTFS in Windows 8 fully leverages capabilities delivered by our industry partners to efficiently support very large capacity disks. You can rest assured that your large-capacity storage needs will be well handled beginning with Windows 8 and NTFS!

/Bryan

Windows Store event and blog

Steven Sinofsky | [2011-12-06T17:54:00+00:00](#)

Today at our Windows Store Preview event in San Francisco, Antoine Leblond revealed the business terms and app policies for anyone developing apps for our new Windows Store. He showed some of the great apps that will be available in the Store at Windows 8 Beta, and [demonstrated the design, capabilities, and flexibility of the Store as a platform](#).

Antoine has also just started a new [Windows Store for developers blog](#), where he will provide info and updates related to the Store for anyone interested in developing apps for Windows 8. The new blog is the place where he will announce and discuss any information about the Store and have an ongoing dialog with app developers about terms, policies, revenue opportunities and platform considerations. It will supplement the conversation we're having here and B8 will continue to be the primary place for our ongoing conversation about engineering Windows 8.

-- Steven

Protecting your digital identity

Steven Sinofsky | [2011-12-14T12:00:00+00:00](#)

We live in a world of usernames, passwords, and PINs when it comes to using our computing devices connected to the Internet. These are very important elements of the digital economy and providing the infrastructure for these in Windows is serious business. This work starts with the most basic step of signing in to Windows, and then includes the technologies used to secure the myriad of accounts you will come to use over time. In this post we take a look at the architectural improvements to Windows that enable even more secure management of your many passwords. Dustin Ingalls, the author of this post, is a group program manager on the security and identity team.

–Steven

One of the challenges that we spent a lot of time thinking about while planning Windows 8 was how to help you manage your digital identity in a way that is both convenient and secure. In today's world, there are a number of very interesting details with respect to digital identities, how they are used, and how they are protected.

Currently, the most common way people verify their digital identity is by using a password. Passwords are used to sign in to your computer, to your bank, to web merchants, and lots of other places. Our research has shown us that the average person using a PC in the United States typically has about 25 online accounts.⁽¹⁾ That's a lot to keep track of! In fact, the data also shows that the number of unique passwords across those 25 accounts is only about 6. For folks who spend time thinking about security, that's a worrisome finding as it shows that the average person reuses the same password quite frequently across accounts. Additionally, given that different websites have different password policies (some require alphanumeric with special characters, some disallow special characters, some have minimum password lengths, some don't, etc.), it's likely that the number of unique passwords across accounts would be *even lower* if websites actually had the same password policies.

On the one hand, that's completely understandable. Remembering a bunch of different passwords is difficult, especially for accounts that we don't use frequently. On the other hand, password reuse is very useful to hackers...they know that if they can learn your password for one site, it's highly likely that you use the same password on other sites. Even worse, an attacker can often use your sign-in information to reset the password for other accounts where the password actually is different. For example, if an attacker can somehow gain access to the password for one of your accounts, there's a strong probability that you use the same password for one of your web email accounts. Given that there are only a handful of major web email providers, finding yours is often pretty easy. Once an attacker gains access to your email, they can go to other common sites (major banks, major online merchants, etc), and use the "lost password" functionality to send a password reset link to the email account that they've already taken over.

(As an aside, the Hotmail team has spent a great deal of effort in redesigning the password recovery process for Hotmail. There are many ways that "bad guys" attempt to compromise online accounts (from all providers) and Hotmail is no different. When your account becomes compromised (or you legitimately forget your password), we have in place a number of security steps to make sure that you, and only you, can restore your account. While these might seem inconvenient, consider the relatively small amount of information you provided in order to sign up. That's why [we encourage people](#) to add either a secondary email account, or even better, a mobile phone number to their account information. The latter is especially hard to duplicate or hack. If you do find yourself with a compromised Hotmail account, you can [reset your password](#). And for those of you using public terminals or untrusted environments to access Hotmail, we encourage you to [use a single-use password](#) sent to you via SMS.)

Clearly, the overall user name/password framework leads to a set of interesting challenges. We all want the web to be frictionless, easy, and safe. Having to remember a whole bunch of complex passwords generally isn't perceived as frictionless. However, using the same easy-to-remember password across multiple sites isn't safe. The ideal solution here involves somehow finding a way to make it both easy and safe to use all of your different digital identities.

In thinking through this challenge, there are two basic approaches to making it both easier and safer to manage your digital identity. One approach is to enable Windows to help you manage your passwords. If you could have complex, unique passwords for each website you visit without having to remember them all, that would certainly be easier than having one easy to remember password – at the same time, the complex password would make the business of compromising your identity much more difficult for hackers. Another approach is to use something other than a password to help protect and establish your identity. There have been a number of alternatives to passwords available for many years—technologies such as One Time Passwords (OTP), certificates, smart cards, etc. However, despite some of the superior security properties of these password alternatives, they haven't exactly caught on for mainstream use—mostly because they're just not as easy to use as a password.

With Windows 8, we provide support for both the safe storage of username/password combinations, and technology to support alternate authentication; that is, we try to make it easier for you to enhance the security of your passwords, and easier to use newer and stronger techniques for protecting your digital identity.

Shortcomings of passwords

There are a number of different methods that attackers use to try to obtain your password. The most common methods are:

Phishing. Phishing involves tricking a user into revealing their password directly to the attacker. Common forms of phishing include "please reset

your account” emails that either ask you to send in your password, or link to a website that looks like a popular website and ask you to enter your password.

Guessing. Given people’s natural preference for easy to remember passwords, attackers can often gain access to an account by simply running through the top 10 or 20 passwords most commonly in use on the Internet. Attackers can also make use of public information (perhaps based on your public social networking profile) to find other easy to guess passwords based on things like your favorite sports team or favorite pet.

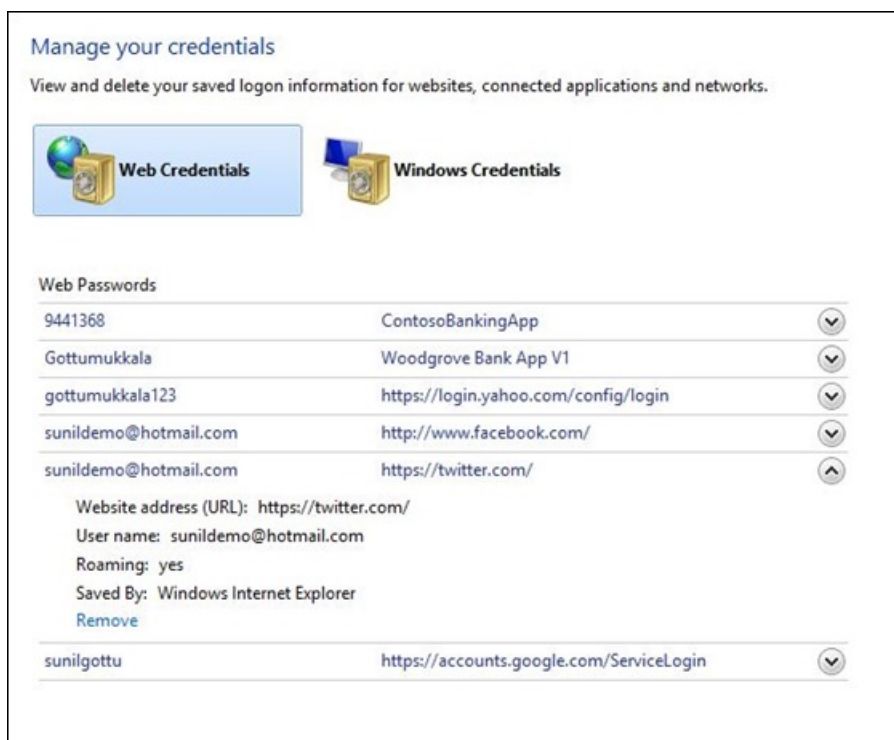
Cracking. In certain situations, an attacker can capture a snippet of data (usually the password’s hash value) and use it to derive your password. There are freely downloadable resources on the Internet that enable attackers to derive passwords less than 8 characters in length very quickly.

Keylogging. If an attacker can successfully install a keylogger on a device, they can record each time you hit a key on your keyboard, and therefore easily pick up name/password combinations. This is an especially common attack on public PCs or kiosks. (That’s why, for example, using the single use code instead of a password for Hotmail is a good idea in such situations!)

Improving the security and usability of passwords

There are a number of important steps you can take to help protect against all of these types of attacks. One of the most important steps is to keep your PC clean and free of malware (to help against phishing and keylogging). Windows 8 includes a number of substantial features in this area that we’ve already covered in prior blog posts ([Secure Boot](#), [SmartScreen and Windows Defender enhancements](#), etc). However, some attacks (like guessing and cracking) rely only on password strength, so it’s important to [use strong, complex passwords](#) that are unique to each account.

Windows 8 simplifies the task of managing unique and complex passwords in two important ways. The first is by providing a way to automatically store and retrieve multiple account names and passwords for all the websites and applications you use, and do so in a protected manner. Internet Explorer 10 uses the credentials that we store to remember names and passwords for websites you visit (if you choose). In addition, anyone building a Metro style app can use a direct API to securely store and retrieve credentials for that app. (It is important to note that IE respects instructions from websites about saving your credentials – some websites specifically request that passwords not be saved.)



Windows 8 allows you to securely store and manage all of your sign-in credentials

The second important investment in this area was covered in an earlier post by Katie Frigon, [Signing into Windows 8 with a Windows Live ID](#). One of the great things you get when you sign in to Windows with your Windows Live ID is the ability to sync the credentials you’ve stored to all of the Windows 8 PCs that you register as your “Trusted PCs.”

When you store credentials in conjunction with signing in to Windows with your Windows Live ID, Windows enables you to set your password for each account to something that is both complex and unique; since Windows 8 will automatically submit the credential on your behalf, you’ll never need to remember it yourself. If you need to see the actual password at some point later, you can view it in the credential manager shown here, from any of your Trusted PCs.

The same principles that keep your credentials safer on websites and applications also apply to how you sign in to your PC. The password you use to protect the account on your PC must be resilient to guessing and cracking. Windows 8 helps with this, helping you to set a very strong password for sign-in, while at the same time enabling a number of “convenience” sign-in methods such as Picture Password and biometrics. This makes it easy to sign in to your PC, without sacrificing security. We will cover Picture Password and other sign-in methods in more detail in a future post.

It is worth reiterating that signing in to your PC with a Windows Live ID, in addition to making sign-in easier, also offers improved sign-in security and gives you a clear path to recovery if you forget your Windows password. With a local password, if you forget your password, you're in a tough spot – if you didn't create a password recovery USB stick, you're stuck rebuilding your machine from scratch. However, if you sign in to your PC with a Windows Live ID, you can [reset your password](#) from another PC. If your Windows Live ID password was stolen somehow, you still have the benefit of a number of Windows Live safety features that are designed to detect compromise and limit your account usage until you can successfully prove that you are the rightful owner of your account and recover your account. The account recovery workflow leverages two-factor authentication features (secondary account proofs) that you set up earlier, such as a mobile phone number or secondary email address (if you haven't already set these up, we'll ask you for them the first time you use your Windows Live ID with Windows 8). Also, even if your Windows Live ID is in a compromised state, you will still have full access to your PC since Windows will cache your last “known good” sign-in password (encrypted, of course) and allow you to use that to continue to sign in.

Creating an easy to use alternative to passwords

While a complex and unique password can be highly resistant to guessing and cracking, because it is what we refer to as a “shared” or “symmetric” key, it is still always vulnerable to phishing and keylogging. Since the key is shared between you and whatever you are signing in to, if the attacker can somehow gain access to your secret key, the game is up. However, there are alternatives that offer strong protection against these types of attacks.

One alternative is **public/private key pairs**. Secure Sockets Layer or Transport Layer Security (SSL/TLS) certificates are an example of this – these are the most commonly used methods for protecting network traffic on the Internet today. Public/private key pairs differ from passwords in that they are an “asymmetric” key – the private key and the public key are different, and knowledge of the public key doesn't enable the attacker to derive the private key. Put very simply, in a public/private key sign-in scheme, when you want to sign in to a service, the service sends you a sign-in request, you sign the request with your private key, and the service then uses your public key to read the signature, proving cryptographically that the sign-in request was signed by whomever holds the corresponding private key. This is referred to as “proof of possession”. So long as you haven't lost your private key, there is strong cryptographic proof that you are the real account holder signing in to the service. Since the actual private key is never exchanged, both keylogging and phishing no longer work. There are no keystrokes to log, and worst case, if a user is tricked into using their private key to sign an authentication request for a fake website, nothing useful is provided—the bad guys can't re-use this information to sign in to the legitimate website.

Although this technology is used extensively on the Internet today, it still hasn't replaced conventional password sign-in. Why not? The main reason is that strong protection of a private key typically requires dedicated hardware (typical examples of this are hardware security modules (HSMs) and smart cards), and historically, use of such hardware hasn't been very convenient—if you lose the hardware or don't have it with you, you can't sign in.

Windows 8 has a number of new features that make it much easier for both users and application developers to make use of public/private key methods. Windows already provides fairly extensive support for use of key pairs and certificates; but strong protection of the private key, as I mentioned earlier, typically relied on HSMs or smart cards. Windows 8 includes a new Key Storage Provider (KSP), which provides easy, convenient use of the Trusted Platform Module (TPM) as a way of strongly protecting private keys. A TPM is a trusted execution environment found on many business-class PCs today (and we expect much broader availability of TPMs when Windows 8 ships), which enables a PC to securely store cryptographic keys. Metro-style apps have APIs that make it easy to automatically enroll and manage keys on your behalf. The Windows Dev Center provides a [sample banking app](#) that shows developers how to use this API.

The KSP feature is particularly useful for banking and commerce applications, since it provides very strong resilience against the most common types of identity attacks on the Internet today while leveraging hardware inside your PC to prevent malware from stealing your private key.

For organizations and businesses that already use smart cards, we've implemented a new feature that overlays the TPM KSP feature and enables the TPM to function as a “virtual smart card.” This solution is more convenient and economical because you don't need a physical smart card reader, but deployment is also easier because the virtual smart card functionality works with existing smart card applications and management solutions. The virtual smart card feature can be used in place of existing smart cards with any application or solution that is smart card compatible – no server- or application-side changes are required. Also, Windows 8 continues to support cards compliant with the Personal Identity Verification (PIV) standard or the Generic Identity Device Specification (GIDS) standard. By using these standards, deployment of smart cards is made much easier in Windows 8. All of these options are available for signing in to Windows (on domain-joined PCs), apps, websites – anything that was previously accessible using a physical smart card. This short video shows this in action after it is set up via policy or logon script by your administrator.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

In a world that is becoming increasingly dependent on maintaining a secure digital identity, we are very passionate about finding ways to make your digital life safer and more secure, without making it more complex. We've spent a great deal of time and focus on this in Windows 8, and we are very much looking forward to hearing your feedback!

- Dustin Ingalls

(1) Source: Dinei Florencio and Cormac Herley, A Large Scale Study of Web Password Habits, Microsoft Research. 2007

Optimizing picture password security

Steven Sinofsky | [2011-12-19T09:00:00+00:00](#)

We wanted to talk a bit more about the security of picture passwords in a follow up post based on some of your comments. Jeff Johnson, the Director of Development for the User Experience team, is particularly interested in the math and security of this feature and authored this post on how to optimize the security of the picture password. Since this is a new form of logging on and concerns over security (especially with mobile devices) as well as new authentication techniques (fragility of facial recognition for example or the challenges we've seen with biometrics) it is no surprise folks took to thinking about potential pitfalls in the approach. Our goal was to provide a convenient mechanism that was clearly no less secure than text passwords (all that math Jeff provided). Below Jeff talks about why this is a robust solution in general. Keep in mind in reading this that over the years many "best practices" have been established for typed passwords (policies such as numbers+letters+mixed case, length, inability to recycle passwords, no dictionary words, etc.) as well as important cautions (such as avoiding public internet terminals with potential for overhead cameras or keystroke loggers) -- these types of practices all have analogs in the use of picture password as you can imagine. Jeff outlines some of these and the logic behind the security of the model. --Steven

A question we've been asked several times in one way or another is "I care about keeping my machine secure; what are the best practices for creating the most secure sequence of login gestures?" This leads to an interesting (at least to me, as a math guy) analysis. It involves game theory, but first I'll distill it down to the following best practices.

- Pick a photo that has at least 10 points of interest. A point of interest is an area that can serve as a landmark for a gesture – a point that you would touch, places you would connect with a line, an area you would circle.
- Use a random mixture of gesture types and sequence. While a line is the gesture that has the most permutations, if you always use 3 lines, that actually makes it easier for an attacker, as they can rule out trying sequences with the other gesture types.
- If you choose to use a tap, a line, and a circle, randomly choose the order of those gestures; this creates 6 times the number of combinations as a predictable order.
- For circle gestures, randomly choose whether you draw it clockwise or counterclockwise. Also consider making the size of the circle bigger or smaller than the "expected" size.
- For line gestures, your instinct may be to always draw from left to right, but it is more secure if you randomly choose the direction with which you connect the two points.
- As with all forms of authentication, when entering your picture password, avoid allowing other people to watch you as you sign in.
- Keep your computer in a secure location where unauthorized people do not have physical access to it. As with any password entry, be aware of line of sight and potential recording devices that intrude on your screen.
- Be aware that smudges on the screen could potentially identify your gestures. Clean your screen thoroughly on a regular basis. Although this increases the risk if you clean, sign in, and then do nothing, the buildup of oils from repeated use is generally easier for an attacker to see (plus, who likes using an oily device?). Note that buildup is more of an issue for entering numeric PINs, when the device is frequently turned on and off and you enter the sequence dozens of times a day (oils can build up in those locations). Periodically look at your screen at an oblique angle while on the picture password login screen and see if there appears to be a pattern pointing to your gesture sequence. If so, either clean your screen or add a handful of additional smudges in the picture password area (which effectively increases the POIs discussed below)

If you follow these tips, you will substantially increase the security of your computer.

As several comments suggested, we also considered shrinking the size of the image and displaying it at random positions and slight rotations on the screen to minimize any risk from smudges. We knew from usability feedback that decreasing the size of the image both increased the difficulty of properly entering the gesture and made the login experience feel less immersive; however, if there were a significant improvement to security, we wanted to consider the costs and benefits. What we discovered was that while shifting the image could reduce the buildup of smudges in specific spots, there were even more prominent "clouds" of taps, lines and circles that were identical relative to each other. With this information, an attacker could easily figure out the gestures relative to each other. With that information, it was a simple exercise to move them around the picture until they appeared to coincide with significant elements of the picture. There wasn't a noticeable improvement in security and we were able to measure significant degradations to the fast and fluid user experience. In reality, using smudges is very difficult. When we took tablets that had been used for a number of days by folks, there were typically too many smudges to even begin to deduce their gesture set. Even when we were given their login sequence and knew what to look for we had limited success. We included this analysis because we feel it is important that whenever any innovative new technology is introduced that potential attack vectors are disclosed and the technical community can reach a general consensus of the degree of a threat and its potential mitigations. Of course we also have confidence that screen technologies will continue to improve and smudges will someday seem quaint.

The analysis

It is also interesting to compute the odds of an attack succeeding in various scenarios. As discussed in the [previous blog post](#), gestures are based on a 100 x 100 grid, giving even the simplest gesture (the tap) a potential of 10,000 values (given proximity matching, this number is effectively reduced to 270). In reality, the number of points of interest (POI) is much lower than that – there are only so many memorable locations in a given photograph.

Although there are other ways to structure an analysis, for the purposes of this discussion we will assume that there are a small number of POIs, and all gestures involve only those points. We assume that taps are directly on a POI, circles only come in two sizes (say, small around the point, and larger around the point) and two directions (clockwise and counterclockwise), and lines always connect two POIs. Because this isn't strictly true, the number of permutations is actually even greater.

Windows provides additional protection for picture passwords (and PINs) by disabling the login mechanism after 5 incorrect tries (you then have to use your conventional password). With this in mind, it is interesting for a given scenario to frame the relative security in two ways.

First, what are the odds that an attacker with full knowledge of your gesture selection methodology would be able to sign in to your machine before the lockout is triggered (we will refer to this as Odds1). If there are x equally likely gesture sequences, then the odds of guessing it in five tries before lockout are $5/x$.

The second interesting view is assume you were given 100 machines each with a password picked randomly according to the rules of the scenario (we will refer to this as Odds100). What are the odds that an attacker could log in to at least one of those machines? Since these are independent events, the odds of this are:

$$1 - \left(\frac{x-1}{x}\right)^{100}$$

Base scenario

Let's assume a horribly insecure scenario: Your "picture" is entirely black with a single white dot in the middle of it. Because there is only one POI, only the tap and circle gesture can be used (there is nowhere to connect a line to). Obviously, if I used only the tap gesture, an attacker would have 100% success as the only valid sequence would be three taps on the white dot. Let's assume we only use circles and no points. There are 4 possible circles we can randomly choose for each gesture. This gives us a total of $4^3 = 64$ possible gesture sequences. For this scenario, Odds1 is 7.81% and Odds100 is 99.97%. It's surprising that for a single machine the odds of a successful sign in with my picture password is less than 8% (my intuition would have guessed a higher number), though you can see it is a virtual certainty that with 100 machines, at least one of them would be compromised. While some users might be comfortable with these odds, most security conscious folks and IT admins who manage a population of machines would find this unacceptable.

Let's now augment the scenario by saying we will randomly choose for each gesture whether it is a tap or a circle. It is tempting to say that this doubles the complexity of each gesture, but it does not. There are 4 possible circles and 1 possible tap, so there are 5 unique gestures giving a total of 125 sequences.

Let's say that we choose to implement our new "random" methodology as follows: flip a coin to determine if it's a tap or a circle. If it's a circle, we'll randomly decide which of the four possibilities it will be. While this seems nice and random, it is actually *less* secure than just using only circles. This is because half the time we will pick a gesture for which there is only one possibility (the tap). An attacker would focus their attack on gestures that featured two or three taps and achieve higher success. An ideal attack strategy (there are others with identical odds) would be to test for 3 taps, and then test for two taps followed by each of the four circle types for the 5 attempts before lockout. Instead of the apparent Odds1 of 4% (an improvement over the previous 7.81%), an attacker would actually achieve Odds1 of 25%, more than three times *worse* than just using circles. Statistics can be tricky!

Fortunately, there is an easy fix to this scenario. For each gesture, we pick a random number between 1 and 5. If it is a 1, we use a tap. Otherwise we use the value to pick one of the 4 circle possibilities. This does yield an Odds1 of 4% (almost twice as good as the first scenario), but the Odds100 is still an abysmal 98.31%.

A slight improvement

Let's make just a small improvement to our methodology. This scenario involves a picture with only two POIs (it's really hard to imagine a real photo this simple, so we can pretend it's a black canvas with two white dots). This allows us to add the line gesture, but there are only two possibilities for it: drawing from the first dot to the second, or from the second to the first.

Learning from the previous example, we will not randomly pick the gesture type and then the gesture. We will sum up all possible gestures and then pick a random number to map with equal probability onto each possible gesture. There are 2 possible taps, 8 possible circles, and 2 possible lines. The total number of gesture sequences is $12^3 = 1728$. This gives us an Odds1 of .29% and Odds100 of 25.2%. It is somewhat remarkable that so simple of a picture with only 2 POIs would have odds this low for a successful attack. Even if you had 100 machines to attempt to break into, you would only succeed getting into at least one machine 1 out of 4 tries.

Ramping it up

Let's assume there are now 5 POIs in your picture. I can begin to imagine some very simple pictures where this might be the case. We now have 5 possible taps, 20 possible circles, and 20 possible lines. This gives us $45^3 = 91,125$ possible sequences. Odds1 is now vanishingly small at 0.0055% and Odds100 is also very low at 0.55%. For many users, these odds are sufficient to protect their data.

To the max

Let's assume you are very security conscious and choose a picture with 10 POIs. There can be debate as to how many POIs a particular photo

contains. However, it doesn't matter how many POIs are "obvious" as long as you pick 10 points that are identifiable to you to randomly choose gestures with. Actually, if some of the points aren't obvious (but you can still reliably target them), that is a security plus.

We now have 10 possible taps, 40 possible circles, and 90 possible lines. This is a very robust $140^3=2,744,000$ sequences. Odds1 is vanishingly small at 0.0002%. In fact, you are more than 50 times more likely to win \$10,000 with a \$1 ticket in the Washington State Select 4 Lottery than you are to have your machine broken into using a picture with 10 POIs! The Odds100 has dropped to 0.018% and even Odds1000 is only 0.18%.

Social engineering

Social engineering is one of the most significant threats to sign-in security of all types, whether password, PIN, or picture password. Using a randomizer to help construct your sign-in sequence is equally useful for each of these methods.

For the technical enthusiast, it is possible to implement the above schemes with a small amount of programming or the use of Excel. However, it would be useful to have a lower tech way of creating a gesture sequence that a larger audience could employ. Of course, we should not be under any illusions that the number of people who seek out these tools and procedures will be any greater than the number who would voluntarily pick strong text passwords if not required by site admins.

Roll of the dice

As a whimsical exercise, I thought it would be fun to come up with an analog way of generating a random gesture sequence. To do this, I chose to employ a six-sided die (D6 for hard core gamers :-)) to generate a 6-POI gesture sequence. In addition to mapping nicely onto the die, a 6 POI picture has the useful property that the number of possible lines (30) exactly equals the number of taps (6) plus circles (24), so it is easy to bifurcate the gesture type as well.

Repeat the following steps for each of the three gestures:

1. Roll the die.
The number indicates which of the six POIs to use for the gesture (for a line it will be the starting POI).
2. Roll the die again.
 - o If the die is even, the gesture will be a line
Roll the die again.
If the number matches the first roll to pick the initial POI, reroll until you get a different number.
This number is the second point for the line.
 - o If the die is odd, the gesture will be a tap or circle
Roll the die again.
Use the roll value list below to determine the gesture.
 - 1 - The gesture is a tap
 - 2 - The gesture is a small clockwise circle
 - 3 - The gesture is a small counterclockwise circle
 - 4 - The gesture is a larger clockwise circle
 - 5 - The gesture is a larger counterclockwise circle
 - 6 - Reroll

As expected, the complexity provided by 6 POIs is between the numbers for 5 POIs and 10 POIs. Odds1 is 0.0023% and Odds100 is 0.23%.

We hope you enjoy using the new picture password sign-in as much as we have enjoyed creating it!

--Jeff Johnson

Refresh and reset your PC

Steven Sinofsky | [2012-01-04T10:00:00+00:00](#)

The power of personalization is something we all love about PCs, but sometimes there is good reason to want to roll back to an earlier state. Most consumer electronics devices today can be reset to some factory state, and so we built this capability into Windows 8 too. Desmond Lee is a program manager on the Fundamentals team and authored this post about “push-button reset.”

--Steven

Many consumer electronic devices these days provide a way for customers to get back to some predefined “good” state. This ranges from the hardware reset button on the back of a wireless network router, to the software reset option on a smartphone. We’ve built two new features in Windows 8 that can help you get your PCs back to a “good state” when they’re not working their best, or back to the “factory state” when you’re about to give them to someone else or decommission them.

Today, there are many different approaches and tools to get a PC back to factory condition. If you buy a PC with Windows preinstalled, it often comes with a manufacturer-provided tool and a hidden partition that can be used for that specific model of PC. You might also use a third-party imaging product, Windows system image backup, or the tried and true method of a clean reinstall from the Windows DVD. While these tools all provide similar functionalities, they don’t provide a consistent experience from one PC or technique to another. If you are the “go to” person for your friends, relatives, or neighbors when they need help with their PCs, you may find that it’s sometimes necessary to just start over and reinstall everything. Without a consistent experience to do this, you might end up spending more time finding the recovery tool for a specific PC than actually fixing the problems, and this gets even worse if you’re helping someone over the phone.

With Windows 8, there are a few key things that we set out to deliver:

- Provide a consistent experience to get the software on any Windows 8 PC back to a good and predictable state.
- Streamline the process so that getting a PC back to a good state with all the things customers care about can be done quickly instead of taking up the whole day.
- Make sure that customers don’t lose their data in the process.
- Provide a fully customizable approach for technical enthusiasts to do things their own way.

As we began planning for Windows 8, we asked ourselves: “Wouldn’t it be great if you could just push a button and everything is fixed?” We really wanted to focus on the concept of “push button”, which translated into a design goal that represents a simple to use, predictable, and fast solution. We also wanted to build on the process many people already use today when they need to start over: back up your data, reinstall Windows and apps, and restore your data. The strength of this approach is that you start over from a truly clean state, but you still get to keep the things you care about. With that as the basis of the solution, our goal was to make the process much more streamlined, less time-consuming, and more accessible to a broad set of customers.

Our solution in Windows 8 consists of two related features:

- **Reset your PC** – Remove all personal data, apps, and settings from the PC, and reinstall Windows.
- **Refresh your PC** – Keep all personal data, Metro style apps, and important settings from the PC, and reinstall Windows.

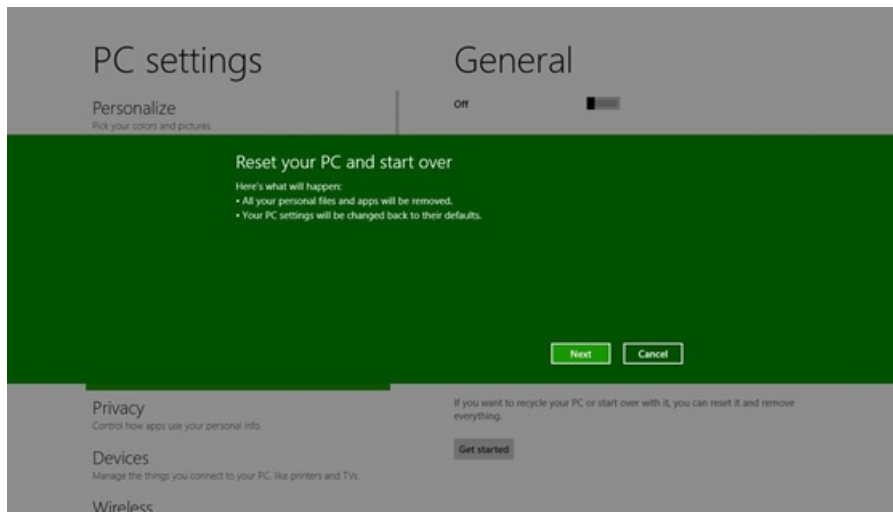
Reset your PC to start over

In some cases, you might just want to remove everything and start from scratch manually. But in other cases, you’re removing your data from a PC because you’re about to recycle or decommission it. For both of these situations, you can easily reset your Windows 8 PC and put the software back into the same condition as it was when you started it for the very first time (such as when you purchased the PC).

Resetting your Windows 8 PC goes like this:

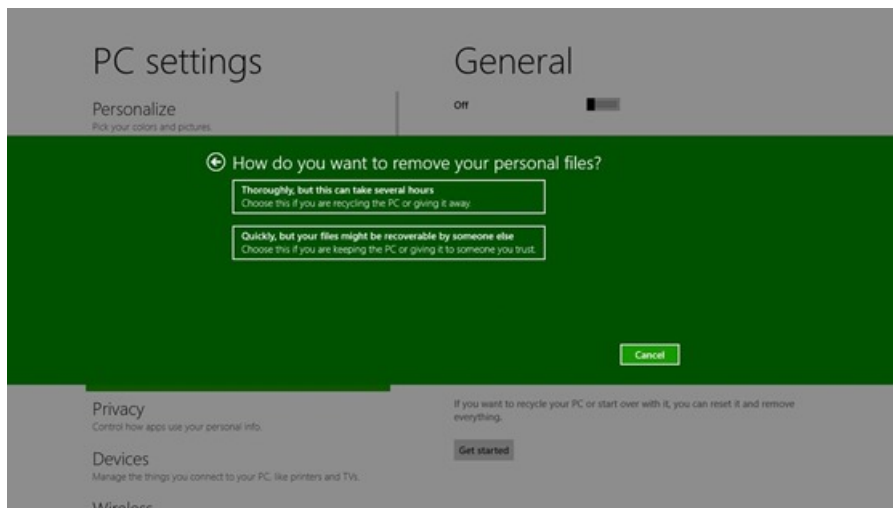
1. The PC boots into the Windows Recovery Environment (Windows RE).
2. Windows RE erases and formats the hard drive partitions on which Windows and personal data reside.
3. Windows RE installs a fresh copy of Windows.
4. The PC restarts into the newly installed copy of Windows.

(Note that the screenshots below reflect changes that we’re making for Beta, some of which are not yet available in Developer Preview)



Resetting your PC

For those of you who worry about data that may still be recoverable after a standard reset, especially on PCs with sensitive personal data, we also will be providing an option in Windows 8 Beta to erase your data more thoroughly, with additional steps that can significantly limit the effectiveness of even sophisticated data recovery attempts. Instead of just formatting the drive, choosing the “Thorough” option will write random patterns to every sector of the drive, overwriting any existing data visible to the operating system. Even if someone removes the drive from your PC, your data will still not be easily recoverable without the use of special equipment that is prohibitively expensive for most people. This approach strikes a good balance between security and performance – a single pass through your hard drive offers more than enough security for typical scenarios such as donation to a local charity, but does not bog you down for hours or days with multi-pass scrubbing operations that might be required for regulatory compliance if you are dealing with highly confidential business and government data.



Choosing how your data should be removed

Refresh your PC to fix problems

Resetting your PC can take you back to square one if you encounter a problem, but that's clearly a very heavy weight solution, something you'd only do as a last resort. But what if you could get the benefit of a reset – starting over with a fresh Windows install – while still keeping *your stuff* intact? This is where Refresh comes in handy. Refresh functionality is fundamentally still a reinstall of Windows, just like resetting your PC as described above, but your data, settings, and Metro style apps are preserved. We have a solution to help you with your desktop apps, too, which I'll talk about a little later.

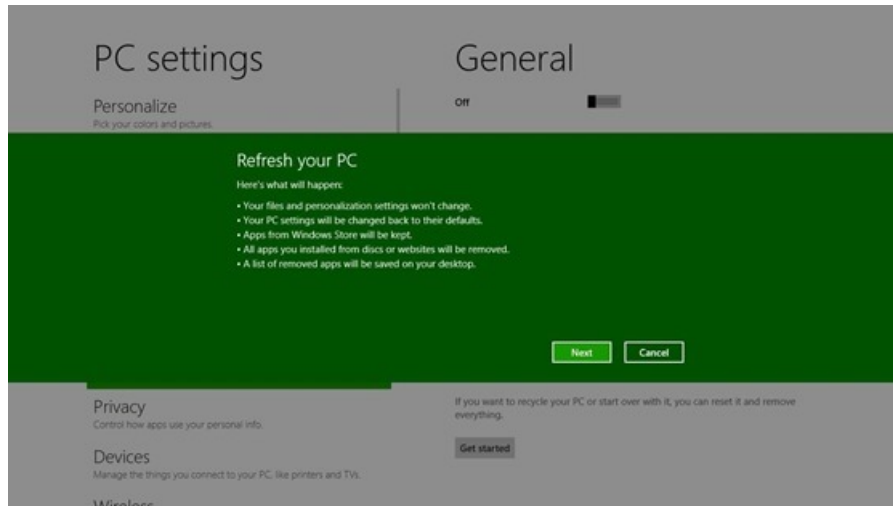
The coolest part about Refresh is there's no need to first back up your data to an external hard drive and restore them afterwards.

Refreshing your PC goes like this:

1. The PC boots into Windows RE.
2. Windows RE scans the hard drive for your data, settings, and apps, and puts them aside (on the same drive).
3. Windows RE installs a fresh copy of Windows.
4. Windows RE restores the data, settings, and apps it has set aside into the newly installed copy of Windows.
5. The PC restarts into the newly installed copy of Windows.

Unlike manually reinstalling Windows, you don't have to go through the Windows Welcome screens again and reconfigure all the initial settings, as

your user accounts and those settings are all preserved. You can sign in with the same account and password, and all of your documents and data are preserved in the same locations they were before. To accomplish this, we actually use the same imaging and migration technologies behind Windows Setup. In fact, the [underlying setup engine](#) is used to perform both Reset and Refresh, which also benefit from the performance and reliability improvements we added to setup for Windows 8.



Refreshing your PC

Misconfigured settings are sometimes the cause of problems that lead to customers needing to refresh their PCs. To ensure that Refresh is both effective in fixing problems and in making sure customers don't lose settings that they might have trouble reconfiguring, we've thought a great deal about which settings to preserve. In Windows 8 Beta, some of the settings we'll preserve include:

- Wireless network connections
- Mobile broadband connections
- BitLocker and BitLocker To Go settings
- Drive letter assignments
- Personalization settings such as lock screen background and desktop wallpaper

On the other hand, we deliberately chose not to preserve the following settings, as they can occasionally cause problems if misconfigured:

- File type associations
- Display settings
- Windows Firewall settings

We will continue to enhance and tune both lists over time based on how we see the feature being used in the Developer Preview and Beta.

Restoring your apps

We preserve only Metro style apps when customers refresh their PCs, and require desktop apps that do not come with the PC to be reinstalled manually. We do this for two reasons. First, in many cases there is a single desktop app that is causing the problems that lead to a need to perform this sort of maintenance, but identifying this root cause is not usually possible. And second, we do not want to inadvertently reinstall "bad" apps that were installed unintentionally or that hitched a ride on something good but left no trace of how they were installed.

It is also important to understand that we cannot deterministically replace desktop apps, as there are many installer technologies as well as custom setup and configuration logic, of which Windows has little direct knowledge. That is why we discourage the use of third-party uninstallers or scrubbers. One simple thing to consider is that many setup and installation programs conditionally implement functionality based on the state of the machine at the time of the install (for example default browser, default photo handler, etc.)

You can, however, cleanly install and uninstall all Metro style apps using the .appx package format. If you're interested in learning more about how Metro style apps work in this regard, check out the following sessions from //build:

- [Platform for Metro style apps](#)
- [Under the hood: installation and updates for Metro style apps](#)

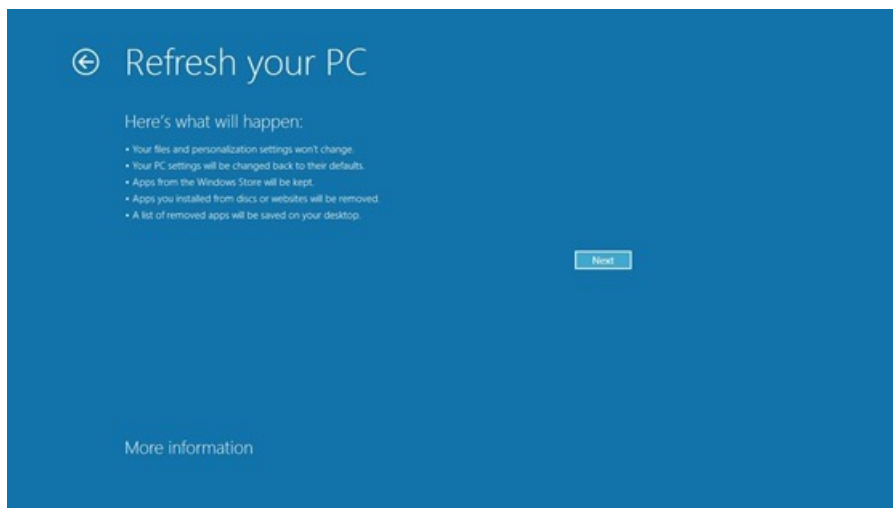
If you do need to reinstall some desktop apps after you refresh your PC, we save the list of apps that were not preserved in an HTML file, and put this list on the desktop, so you have a quick way to see what you might need to reinstall and where to find them.

One caution is that if any desktop apps you have require a license key, you will need to follow your manufacturer's instructions for how to reuse the key. This might involve uninstalling the app first, going to a web site, or going through some automated steps by phone, for example.

What if the PC can't boot?

When your PC is able to boot normally, you can get started with refreshing or resetting it from PC settings. (This is the Metro style app that we called “Control Panel” in the Windows Developer Preview. It is different from the standard Control Panel that you can still use for more complex tasks from the desktop.) The options are easily discoverable, and will be in the same place on every Windows 8 PC. Once launched, you can get through them with just a few clicks, which makes it easy to guide someone through the process over the phone.

However, in some situations, the PC might not boot successfully and you might want to refresh or reset it to get it back to a working condition. In a previous post, Billie Sue Chafins discussed how [the boot experience has been redesigned from the ground up](#), including troubleshooting using Windows RE. Naturally, we’ve made it possible for you to refresh or reset your PC from there as well.



Refreshing or resetting your PC from the new boot UI

In Windows 8 Beta, there will also be a tool that you can use to create a bootable USB flash drive, in case even the copy of Windows RE on the hard drive won't start. You'll be able to start your PC with the USB drive, and fix problems by refreshing your PC or performing advanced troubleshooting. And if your PC comes with a hidden recovery partition, you'll even have the option to remove it and reclaim disk space once you've created the USB drive.

Refreshing your PC to a state you define, including desktop apps

We know that many of you like to first configure your PC just the way you like it, by installing favorite desktop apps or removing apps that came with the PC, and then create an image of the hard drive before you start using the PC. This way, when you need to start over, you can just restore the image and you won't have to reinstall the apps from scratch.

With this in mind, we've made it possible for you to establish your own baseline image via a command-line tool (**recimg.exe**). So when you get a Windows 8 PC, you will be able to do the following:

1. Go through the Windows first-run experience to configure basic settings.
2. Install your favorite desktop apps (or uninstall things you don't want).
3. Configure the machine exactly as you would like it.
4. Use **recimg.exe** to capture and set your custom image of the system.

After you've created the custom image, whenever you refresh your PC, not only will you be able to keep your personal data, settings, and Metro

style apps, but you can restore all the desktop apps in your custom image as well. And if you buy a PC that already comes with a recovery image on a hidden partition, you'll be able to use the tool to switch from using the hidden partition to instead use the custom image you've created.

If you'd like to try this out now, a preview version of this tool is included in the Windows 8 Developer Preview. You can try it out by typing the following in a command prompt window running as administrator:

```
mkdir C:\RefreshImage  
  
recimg -CreateImage C:\RefreshImage
```

This creates the image under C:\RefreshImage and will register it to be used when you refresh your PC. Again, **this is a very early version of the tool**, so we know it's not perfect yet. Rest assured that we're working hard to get it ready for primetime.

Getting back to productivity quickly

When we started building these features, we knew that ease of use wasn't going to be enough – refresh and reset had to be fast as well. Many of the recovery tools preloaded with PCs today take an hour or more just to get the PC back to factory condition, and you often still have to spend hours copying back your data and reconfiguring everything. Even solutions that back up and restore the entire hard drive can take a long time, as the time required generally scales with how much data you have.

To give an example of the performance of our solution, we installed a clean copy of Windows on the [Developer Preview PC](#) that we gave out to attendees at the BUILD conference, filled most of the drive with data, and measured the time it took to go through various recovery operations:

Recovery operation	Time required
Refreshing the PC	8 minutes 22 seconds
Resetting the PC (quick)	6 minutes 12 seconds
Resetting the PC (thorough, with BitLocker enabled)	6 minutes 21 seconds
Resetting the PC (thorough, without BitLocker)	23 minutes 52 seconds

Compared to a baseline time of 24 minutes 29 seconds for restoring the same contents from a system image backup, most of these times show a considerable improvement.

The beauty of refreshing the PC is that performance isn't impacted by the amount of data you have. Using the migration technology behind Windows Setup, your data never leaves the drive, and they are not physically moved from one location on the disk to another either, hence minimizing disk reads/writes. Restoring a system image from an external drive using the Windows backup utility, on the other hand, took much longer due to the data in the backup, even with the relatively small 64GB drive on the prototype PC. Thoroughly erasing data did take a bit longer than the other operations, as every sector of the drive had to be overwritten. However, you may also notice that when BitLocker drive encryption was enabled on the drive, this process took much less time. This is due to an optimization we employ so that erasing an encrypted drive would require erasing only the encryption metadata, rendering all the data unrecoverable.

A consistent and easy way to get back to a known good state

Sometimes things can go wrong and you just want to get back to a good state quickly, while other times you might want to remove your data before giving a PC to another family member, employee, or co-worker. With Windows 8, we've streamlined these processes and made them more accessible to customers with the new refresh and reset features. Here's a video showing these features in action:

Your browser doesn't support HTML5 video.
Download this video to view it in your favorite media player:
[High quality MP4](#) | [Lower quality MP4](#)

We hope you'll find these features useful and time-saving when you're fixing your own PC or helping others with theirs.

-- Desmond Lee

Virtualizing storage for scale, resiliency, and efficiency

Steven Sinofsky | [2012-01-05T11:15:00+00:00](#)

*In this post, we are going to dive into a feature in the Windows 8 Developer Preview. **Storage Spaces** are going to dramatically improve how you manage large volumes of storage at home (and work). We've all tried the gamut of storage solutions—from JBOD arrays, to RAID boxes, or NAS boxes. Many of us have been using Windows Home Server Drive Extender and have been hoping for an approach architected more closely as part of NTFS and integrated with Windows more directly. In building the Windows 8 storage improvements, we set out to do just that and developed Storage Spaces. Of course, the existing solutions you already use will continue to work fine in Windows 8, but we think you will appreciate this new feature and the flexible architecture. As we talk all about consumer electronics next week, thinking about all the media we all have in photos (especially huge digital negatives) and videos, this feature is sure to come in handy. In this post, Rajeev Nagar, a group program manager on our Storage and File System team, details this new feature.*

In previous posts we've seen folks jump to try to identify edge cases or debug the designs. We're trying an FAQ approach at the end of this post to see if we can focus the dialog a bit ?? The FAQ also talks about the numerous opportunities to use PowerShell as a management tool for Storage Spaces.

--Steven

[By my own admission](#), I am a digital packrat. My data collection continues to expand and includes some of my most precious memories, including irreplaceable photos and home videos of my children since their birth. For quite some time now, I have sought a dependable, expandable, and easy to use solution that maximizes utilization of my ever-growing collection of USB drives. Further, I want guarantees that my data will always be protected despite the occasional hardware failure.

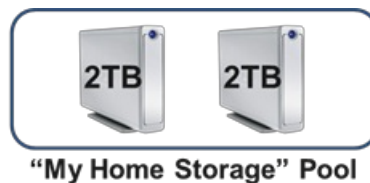
Windows 8 provides a new capability called **Storage Spaces** enabling just that. In a nutshell, Storage Spaces allow:

- Organization of physical disks into *storage pools*, which can be easily expanded by simply adding disks. These disks can be connected either through USB, SATA (Serial ATA), or SAS (Serial Attached SCSI). A storage pool can be composed of heterogeneous physical disks – different sized physical disks accessible via different storage interconnects.
- Usage of virtual disks (also known as *spaces*), which behave just like physical disks for all purposes. However, spaces also have powerful new capabilities associated with them such as thin provisioning (more about that later), as well as resiliency to failures of underlying physical media.

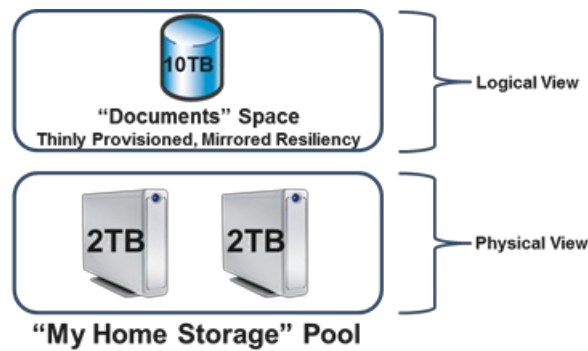
Before we start exploring Storage Spaces in more detail, I will digress briefly to give you a little more context: some of us have used (or are still using), the Windows Home Server Drive Extender technology which was [deprecated](#). Storage Spaces is not intended to be a feature-by-feature replacement for that specialized solution, but it does deliver on many of its core requirements. It is also a fundamental enhancement to the Windows storage platform, which starts with NTFS. Storage Spaces delivers on diverse requirements that can span deployments ranging from a single PC in the home, up to a very large-scale enterprise datacenter.

Pools and spaces

The figure below illustrates the concept of a storage pool. As you can see, we have taken a pair of 2TB (note we use byte measurements as you see in marketing) USB disks and “pooled them” (logically speaking) for subsequent usage.



From this storage pool, we are free to create one or multiple spaces. Note that once physical disks have been added to a pool, they are no longer directly usable by the rest of Windows – they have been *virtualized*, that is, dedicated to the pool in their entirety. And although we call this “virtualized,” the storage and reliability provided is very real. The available storage capacity can be utilized through creation of spaces from this pool. In the illustration below, we have carved out one such space from the “My Home Storage” pool.



This virtual disk is usable just like a regular physical disk – you can partition it, format it, and start copying data to it. You will notice, however, that the space has a couple of interesting properties:

- Its logical capacity is listed as 10TB although the underlying physical disks in the pool have only 4TB of total raw capacity. As a result, you no longer need to worry up-front about the size.
- Resiliency is built in by associating the *mirrored* attribute, which means that there are at least two copies of all data contained within the space on at least two different physical disks. Because the space is mirrored, it will continue to work even if one of the physical disks within the pool fails.

The magic that allows us to create a 10TB mirrored space on 4TB of total raw capacity is called **thin provisioning**. Thin provisioning ensures that actual capacity is reserved for the space only when you decide to use it, for example, when you copy some files to the volume on the space. Previously allocated physical capacity can be reclaimed safely whenever files are deleted, or whenever an application decides that such capacity is no longer needed. This reclaimed capacity is subsequently available for usage by either the same space, or by some other space that is carved out from the same pool. We achieve all of this through architected cooperation between the underlying file-system (NTFS) and Storage Spaces.

With thin provisioning, you can augment physical capacity within the pool on an as-needed basis. As you copy more files and approach the limit of available physical capacity within the pool, Storage Spaces will pop up a notification telling you that you need to add more capacity. You can do so very simply by purchasing additional disks and adding them to your existing pool.



As you see in the illustration above, we have expanded the raw capacity of the “My Home Storage” pool by purchasing and adding four 3TB disks – of course, you could just as well connect SATA and/or SAS storage in conjunction with USB-connected physical disks, and, grow your pool capacity that way. Once we have added this physical capacity, we don’t need to do anything more to consume it. We can simply keep copying files or other data to the space within the pool and this space will automatically grow to utilize all available capacity within the containing pool, subject to its maximum logical size of 10TB. If needed, you can certainly also increase the maximum logical size of a space.

You do not need to explicitly inform Storage Spaces which of your USB disks should be used for each of the spaces you have created. Behind the scenes, Storage Spaces optimally manages the capacity of each of the physical disks within the storage pool, for all the spaces carved out from the pool.

Another core (also optional) capability associated with a space is resiliency to failure of the physical disks comprising the storage pool. For example, the space we’ve illustrated above is a **mirrored** space (in other words, it has the *mirrored* resiliency attribute associated with it). This *mirrored* setting ensures that we always store at least two (and optionally three) complete copies of data on different physical disks within the pool. This way, despite partial or complete disk failure, you’ll never need to worry about loss of data. As a matter of fact, the physical disks comprising the pool are typically not even visible to other components within Windows or to applications running on your PC. By extension, the fact that some physical disks within the pool have failed, is completely shielded from other Windows components or applications. They continue to operate on the space, completely oblivious to the fact that Storage Spaces is working quietly in the background to maintain data availability. Additionally, upon disk failure, Storage Spaces automatically regenerates data copies for all affected spaces as long as sufficient alternate physical disks are available within the pool.

Resiliency through mirroring

It might be interesting to more closely examine how your data is mirrored on different disks. The illustration below shows how a (two-copy) mirrored space is constructed from a two-disk pool:

Space Offset	Length	Disk Number	Drive Offset	Column
127.00G	256.00M	0	127.50G	0
127.00G	256.00M	1	127.50G	1
127.25G	256.00M	0	127.75G	0
127.25G	256.00M	1	127.75G	1
127.50G	256.00M	0	128.00G	0
127.50G	256.00M	1	128.00G	1
127.75G	256.00M	0	128.25G	0
127.75G	256.00M	1	128.25G	1
128.00G	256.00M	0	128.50G	0
128.00G	256.00M	1	128.50G	1
128.25G	256.00M	0	128.75G	0
128.25G	256.00M	1	128.75G	1
128.50G	256.00M	0	129.00G	0
128.50G	256.00M	1	129.00G	1
128.75G	256.00M	0	129.25G	0
128.75G	256.00M	1	129.25G	1
129.00G	256.00M	0	129.50G	0
129.00G	256.00M	1	129.50G	1

In this case, Storage Spaces has allocated physical capacity for the mirrored space in what we call “*slabs*”, which are multiples of 256MB. Also, for this particular example, half of each slab is mirrored on 2 separate disks. Even if one of the two disks fails, Storage Spaces can continue to deliver your data because at least one copy exists on a non-failed physical disk. When multiple disks are available, Storage Spaces spreads slabs across suitable disks as shown in the six-disk pool below:

Space Offset	Length	Disk Number	Drive Offset	Column
127.25G	256.00M	6	42.75G	0
127.25G	256.00M	7	42.75G	1
127.50G	256.00M	2	43.00G	0
127.50G	256.00M	3	43.00G	1
127.75G	256.00M	4	43.00G	0
127.75G	256.00M	5	43.00G	1
128.00G	256.00M	6	43.00G	0
128.00G	256.00M	7	43.00G	1
128.25G	256.00M	2	43.25G	0
128.25G	256.00M	3	43.25G	1
128.50G	256.00M	4	43.25G	0
128.50G	256.00M	5	43.25G	1
128.75G	256.00M	6	43.25G	0
128.75G	256.00M	7	43.25G	1
129.00G	256.00M	2	43.50G	0
129.00G	256.00M	3	43.50G	1

When a pool disk fails, Storage Spaces identifies the impacted slabs for all spaces utilizing the failed disk, and reallocates them to any available hot-spare disk *or* to any other suitable disk within the pool (*hot-spares* are reserved disks within the pool, only to be used as automatic replacements for failed disks). This self-healing is done automatically and transparently so as to minimize the need for manual intervention. We’ve also optimized for speed to prevent data loss from multiple hardware failures at the same time.

Resiliency through parity

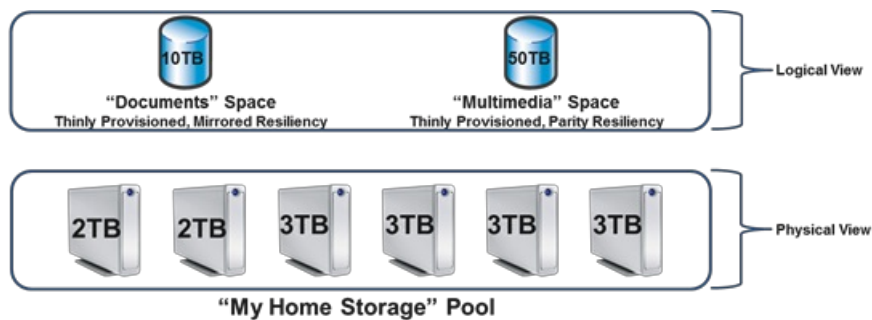
There’s another resiliency attribute, called *parity*, which directs Storage Spaces to store some redundancy information alongside user data contained within the space, thereby enabling automatic data reconstruction in the event of physical disk failure. While conceptually similar to mirroring, parity-based resiliency utilizes capacity more efficiently than mirrored spaces do, but with higher random I/O overhead. Parity spaces are well suited for storing data such as large home videos, which have large capacity requirements, large sequential (predominantly append) write requests, and an infrequent-to-minimal need to update existing content.

Akin to mirrored spaces, slabs for parity spaces are strewn across available disks (with capacity utilized for *parity* information) as shown below for a *parity* space contained within a six-disk pool:

Space	Offset	Length	Disk Number	Drive Offset	Column
170.00G		256.00M	0	34.75G	0
170.00G		256.00M	1	34.75G	1
170.00G		256.00M	2	34.75G	2
170.00G		256.00M	3	34.75G	3
170.00G		256.00M	4	34.75G	4
170.00G		256.00M	5	34.75G	5
171.25G		256.00M	0	35.00G	0
171.25G		256.00M	1	35.00G	1
171.25G		256.00M	2	35.00G	2
171.25G		256.00M	3	35.00G	3
171.25G		256.00M	4	35.00G	4
171.25G		256.00M	5	35.00G	5
172.50G		256.00M	0	35.25G	0
172.50G		256.00M	1	35.25G	1
172.50G		256.00M	2	35.25G	2
172.50G		256.00M	3	35.25G	3
172.50G		256.00M	4	35.25G	4
172.50G		256.00M	5	35.25G	5

When a disk fails, the parity space recovers equally transparently and automatically as does the mirrored space. For parity spaces, Storage Spaces utilizes the parity information to reconstruct affected slabs for all affected spaces, and then automatically reallocates the slab to utilize any available hot-spare disk *or* any other suitable disk within the pool (just as it does for mirrored spaces)

The illustration below shows two spaces – one with mirrored resiliency and the other with parity resiliency – carved out from the same pool:



Obviously, both spaces above are thinly provisioned and share the same backing pool (physical disks). Slabs for both spaces are intermingled, and optimally spread over all available physical disks, although each space uses different mechanisms to recover from physical disk failure.

You can access spaces contained within a pool, as long as a simple majority of physical disks comprising the pool are healthy and connected to your PC, a concept called *quorum*. For example, you will need four of the six disks comprising the **My Home Storage** pool to be healthy and physically connected to the PC in order to access either the **Documents** or the **Multimedia** space. Of course, as previously stated, the resiliency attribute associated with the space determines degree of data availability in the presence of physical disk failure – for example, if the **Documents** space is three-way mirrored and allowed to use all disks within the pool, you can continue accessing data despite the loss of any two disks.

I'll explain the virtualization capabilities of Storage Spaces by walking you through a common usage scenario. Imagine that you have just purchased a Windows 8 PC and wish to use this machine as a central repository for much of the digital content in your home or small business. A reasonable setup would involve creation of two resilient spaces – one is a **mirrored** space for your important documents and the like (these are typically modified more often), and the other is a **parity** space, for your large multi-media content like home videos and family pictures, which you typically update less often, but view more often. By using the appropriate resiliency scheme, you can optimize for both capacity utilization as well as for best performance.

Logically, your storage configuration would look exactly like the illustration provided above, wherein two spaces with different resiliency attributes are carved out from a single pool. Achieving this is quite simple:

1. Connect your physical disks to your PC via USB
2. Create your pool and the two spaces

You can invoke `powershell` to create the pool and spaces, as well as to complete more advanced tasks. In our example, we have purchased and connected six physical disks to our PC. Below are the simple PowerShell commands to set up our pool and two spaces:

(a) To create a storage pool:

```
>$pd = Get-PhysicalDisk
>New-StoragePool -PhysicalDisks $pd -StorageSubSystemFriendlyName *Spaces* -FriendlyName
"My Home Storage"
```

(b) To create the two spaces:

```
>New-VirtualDisk -StoragePoolFriendlyName "My Home Storage" -ResiliencySettingName Mirror -Size 10TB -Provisioningtype Thin -FriendlyName "Documents"  
>New-VirtualDisk -StoragePoolFriendlyName "My Home Storage" -ResiliencySettingName Parity -Size 50TB -Provisioningtype Thin -FriendlyName "Multimedia"
```

Note that the above commands will **only** work on the forthcoming Windows 8 Beta and subsequent releases. A preliminary version of Storage Spaces is available in the Windows 8 Developer Preview, but the above PowerShell commands will not work in that build. If you are curious to try out Storage Spaces on the Developer Preview build, you can use the below alternative commands:

(a) To create a storage pool:

```
>$pd = Get-PhysicalDisk  
>New-StoragePool -PhysicalDisks $pd -FriendlyName "My Home Storage" -  
StorageSubSystemFriendlyName *Spaces*
```

(b) To create the two spaces:

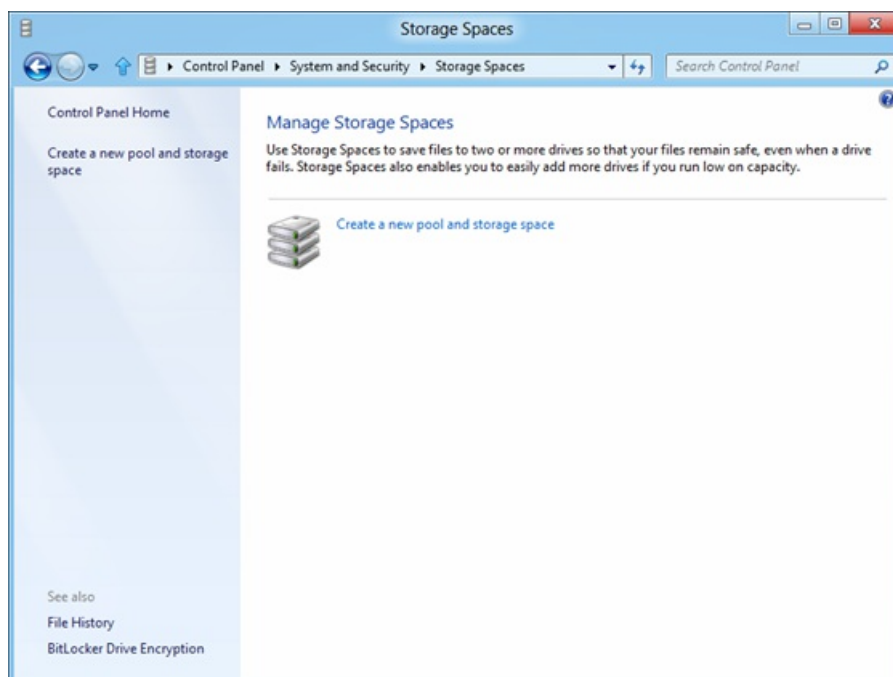
```
>New-VirtualDisk -StoragePoolFriendlyName "My Home Storage" -StorageAttributesName Mirror -Size 2TB -ProvisioningScheme Sparse -FriendlyName "Documents"  
>New-VirtualDisk -StoragePoolFriendlyName "My Home Storage" -StorageAttributesName Parity -Size 1TB -ProvisioningScheme Sparse -FriendlyName "Multimedia"
```

Also note that, in the Developer Preview, space sizes were limited to < 2TB. That limitation will be removed in the Beta release. Since the availability of the WDP release, we have also activated many additional features within Storage Spaces.

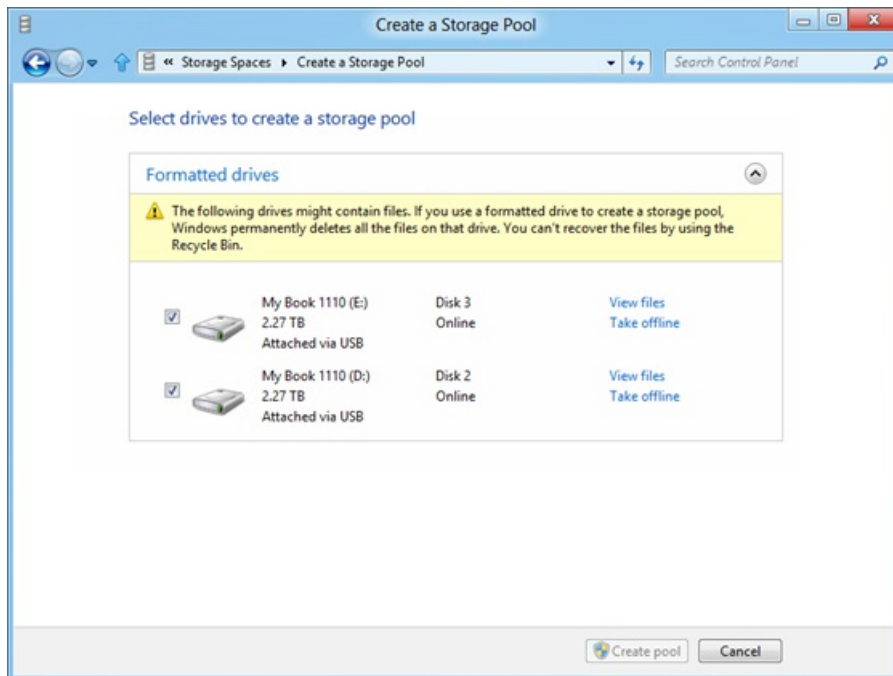
We now get to take a sneak peak at an alternative easy-to-use tool to configure pools and spaces. Beginning with the forthcoming Windows 8 Beta, you can simply go to Control Panel and walk through the sequence below:

(a) To create our pool and a mirrored space, go to Control Panel, click **System and Security**, and then **Storage Spaces**.

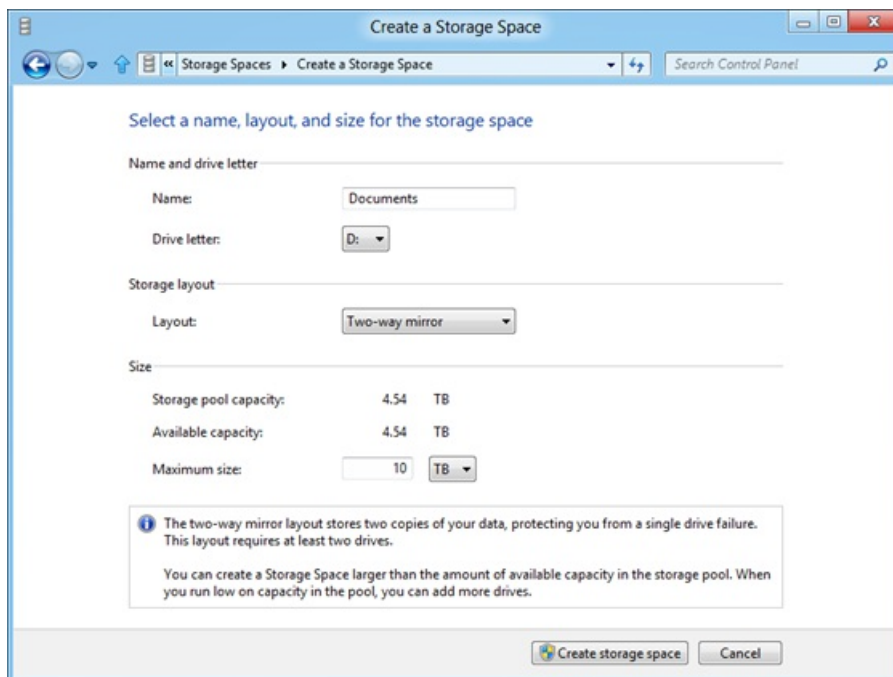
Click **Create a new pool and storage space**.



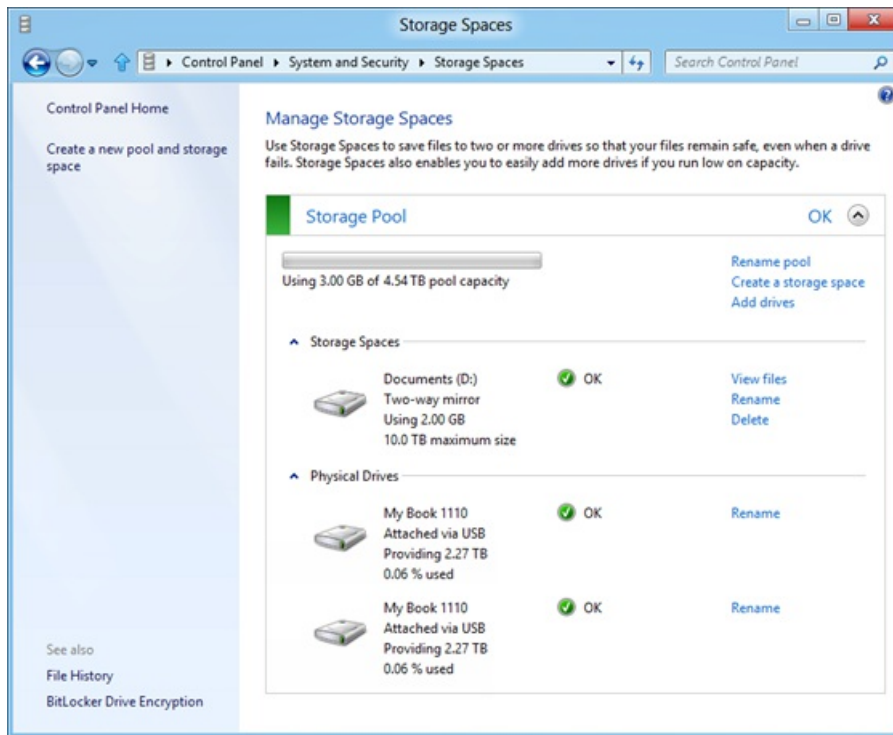
Select the drives you want to add to the new pool.



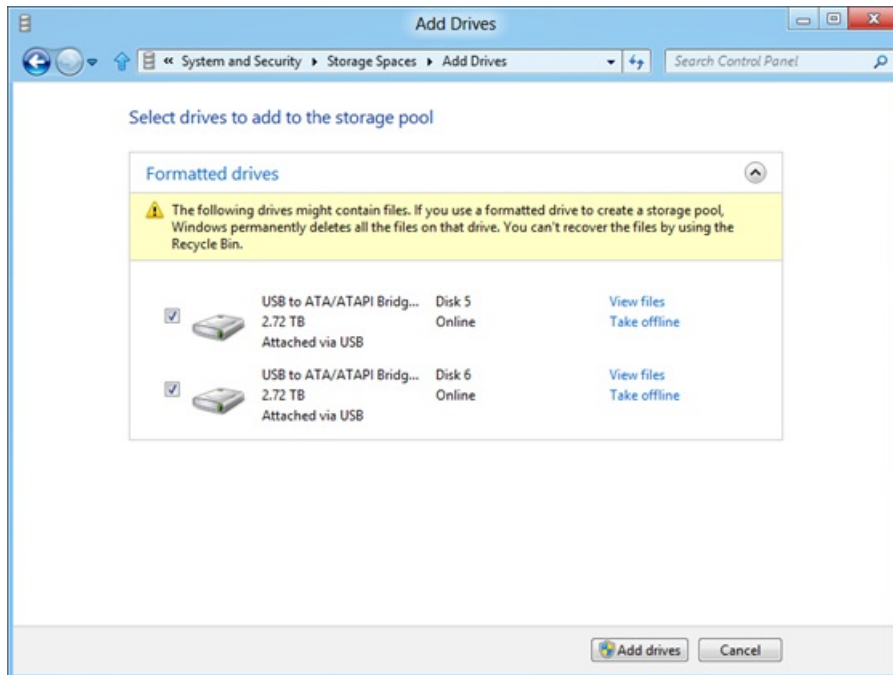
Select your resiliency mechanism and other options.

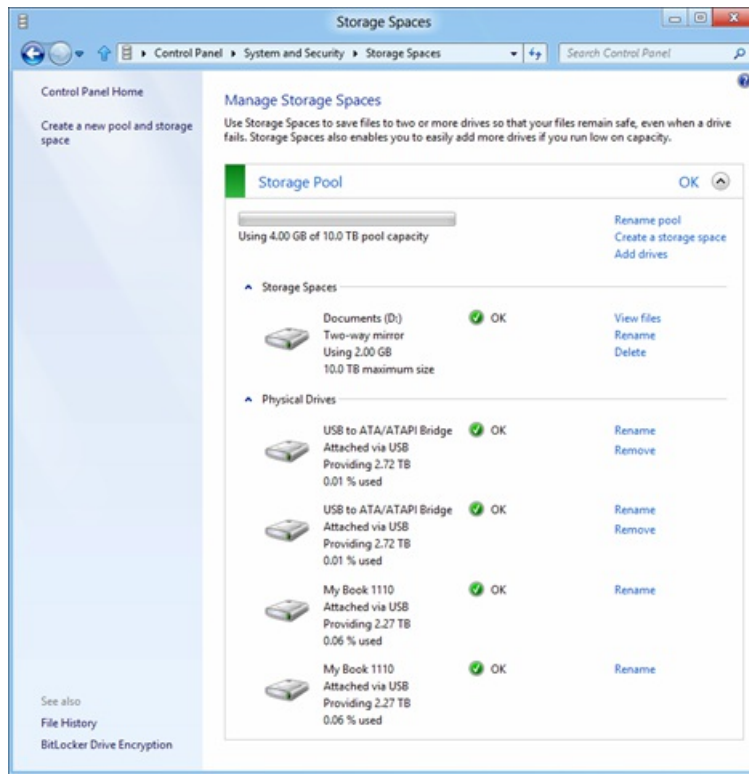


Note that you can assign a drive letter and format the resultant volume as part of creating the space.

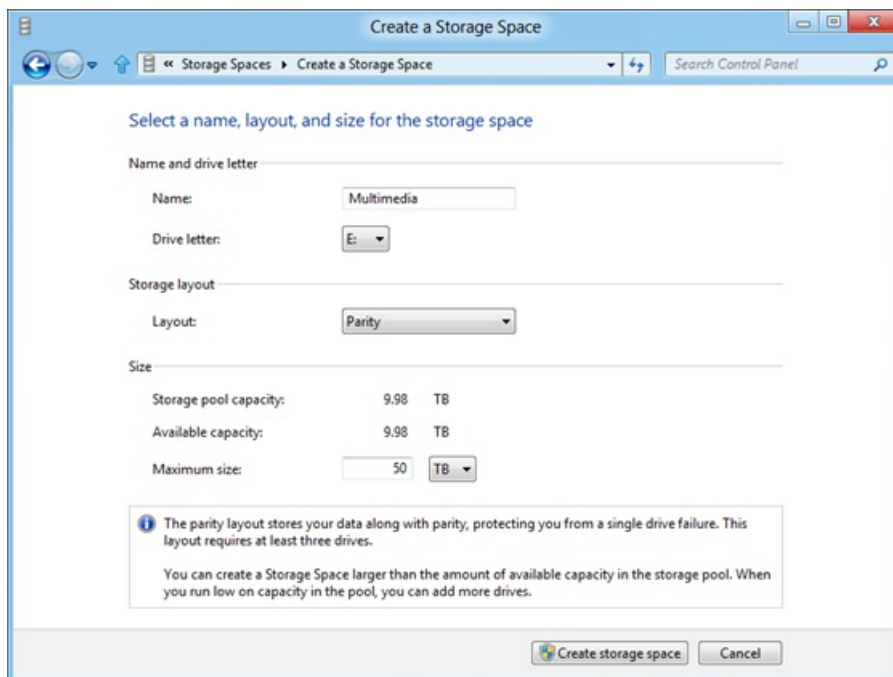


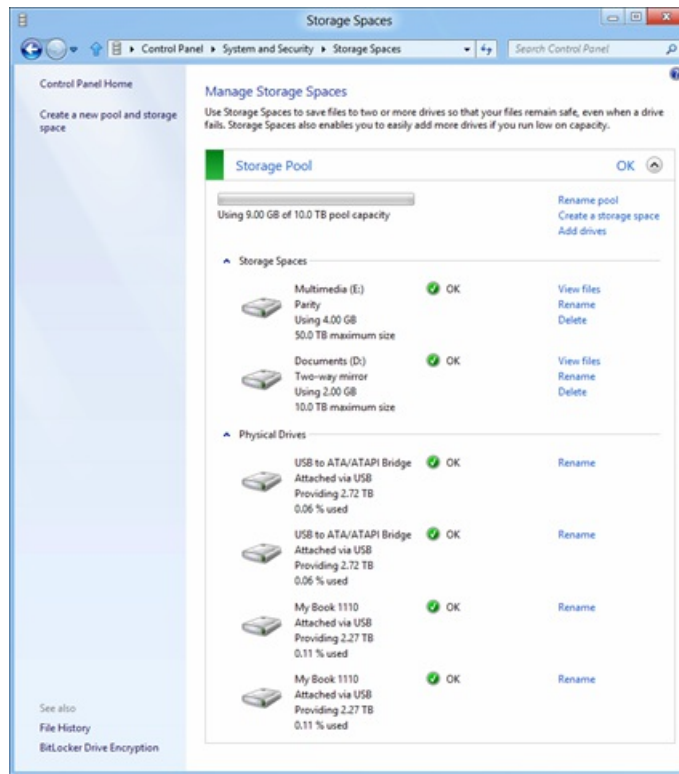
(b) To add a couple of disks to an existing pool, select the drives you want to add.



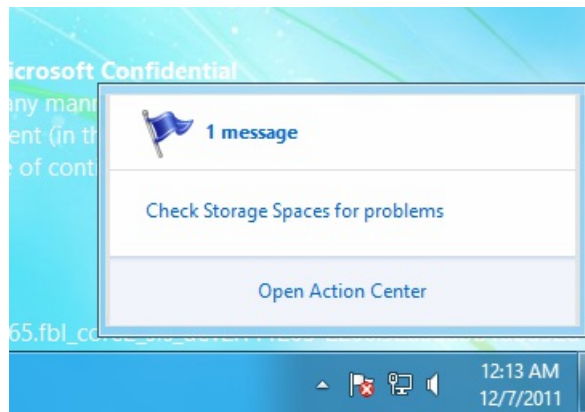


(c) To create an additional parity space, click **Create a storage space**, and then select **Parity** from the layout options.

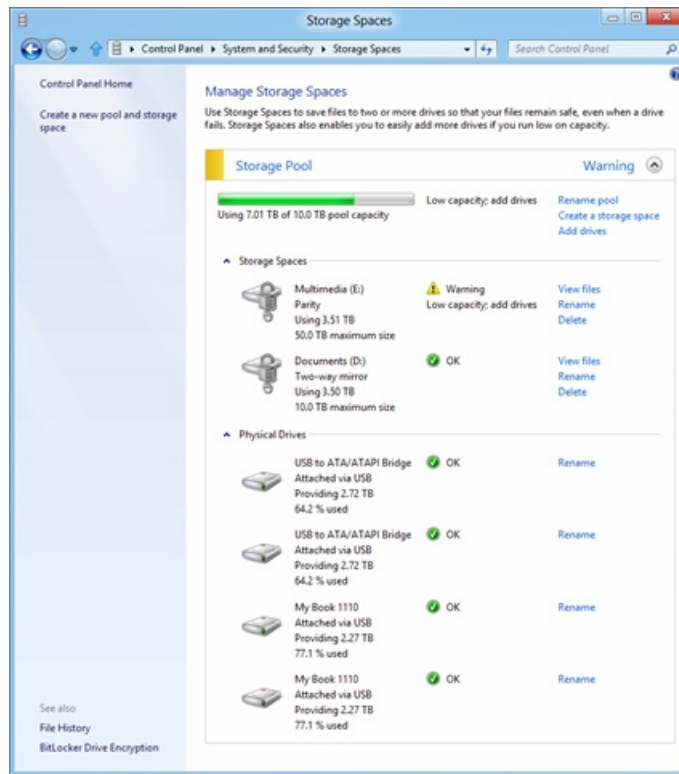




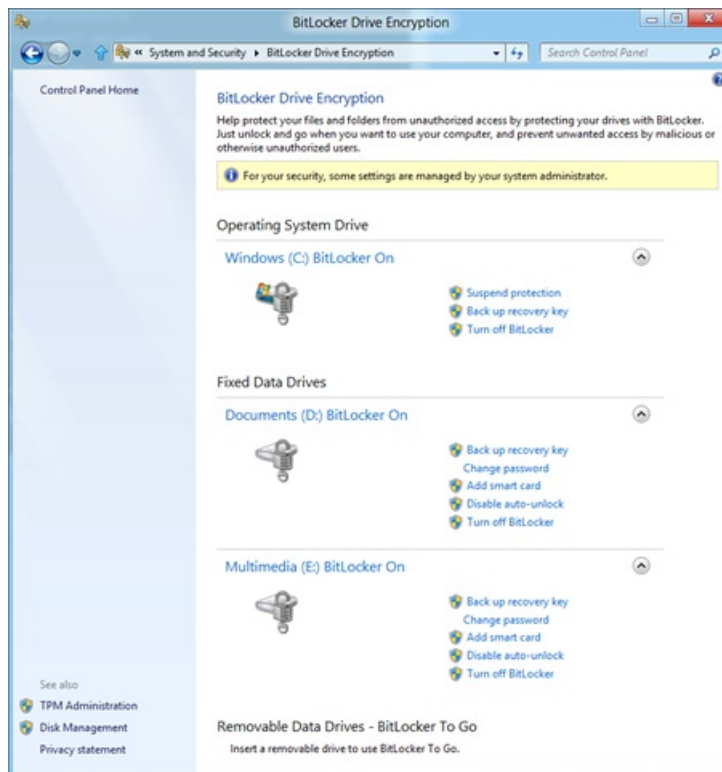
(d) In the event you start running out of capacity, expect a notification like this:

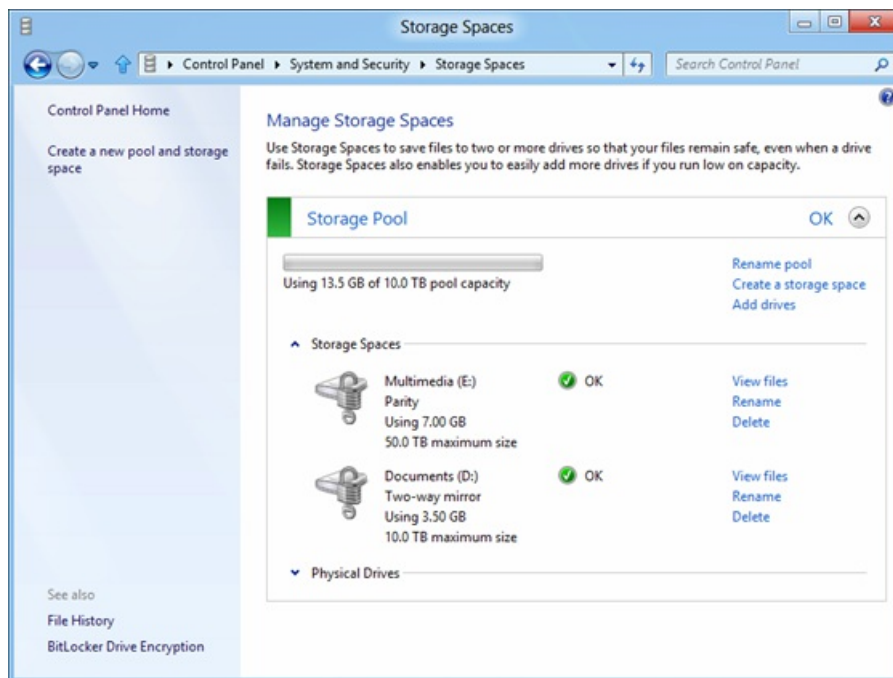


Click the notification to see information about the problem and how to fix it.



That's all you need to do to start using Storage Spaces. Once the spaces have been created, you can utilize them just like any other "disk." For example, you can turn on BitLocker for the spaces you have created, as shown below.





There is a lot more to say about the many capabilities of Storage Spaces and how other Windows technologies can also leverage these capabilities – we will continue with this discussion in subsequent write-ups.

I hope you find this new capability intriguing and encourage you to play with it. It will all be available to you as part of the Windows 8 Beta release in addition to the features available in the Developer Preview.

- Rajeev

Storage Spaces FAQ

We know that some of you will still have questions about Storage Spaces, so here is an FAQ that we hope will cover most of them. As we get more questions from you in the Comments, we will try to update this FAQ to be more complete.

Q) I use Windows Home Server with Drive Extender. Is there a tool to help me migrate data from the Drive Extender format to Storage Spaces?

No. You will need to create a pool on a Windows 8 PC with a fresh set of disks. Then, you can simply copy data over from your Drive Extender-based volumes to a space within your pool. The functionality delivered through Storage Spaces is more flexible and better integrated with NTFS, so it will generally be more reliable and useful.

Q) Are Storage Spaces some kind of RAID? If it is, what RAID versions do you implement?

Fundamentally, Storage Spaces virtualizes storage in order to be able to deliver a multitude of capabilities in a cost-effective and easy-to-use manner. Storage Spaces delivers resiliency to physical disk (and other similar) failures by maintaining multiple copies of data. To maximize performance, Storage Spaces always stripes data across multiple physical disks. While the RAID concepts of mirroring and striping are used within Storage Spaces, the implementation is optimized for minimized user complexity, maximized flexibility in physical disk utilization and allocation, and fast recovery from physical disk failures. Given these significant differences in objectives and implementation between Storage Spaces and traditional inflexible RAID implementations, the RAID nomenclature is not used by Storage Spaces.

Q) How does the read performance of a space compare to RAID 0 or RAID 10?

For both mirrored and striped spaces, read performance is very competitive with optimized RAID 0 or RAID 10 implementations.

Q) Can I use a RAID enclosure with Storage Spaces for additional reliability and/or performance? Is that a good idea?

We don't recommend it. Storage Spaces were designed to work with off-the-shelf commodity disks. This feature delivers easy-to-use resiliency to disk failures, and optimizes concurrent usage of all available disks within the pool. Using a RAID enclosure with Storage Spaces adds complexity and a performance penalty that does not provide any improvement in reliability.

Q) Can I boot from a space?

In Windows 8, you cannot boot from a space. As an alternative, you can continue to use dynamic volumes for booting. At release, we will offer guidance on how you can add appropriately partitioned system/boot disks (with dynamic volumes) to a pool.

Q) What is the minimum number of disks I can use to create a pool? What is the maximum?

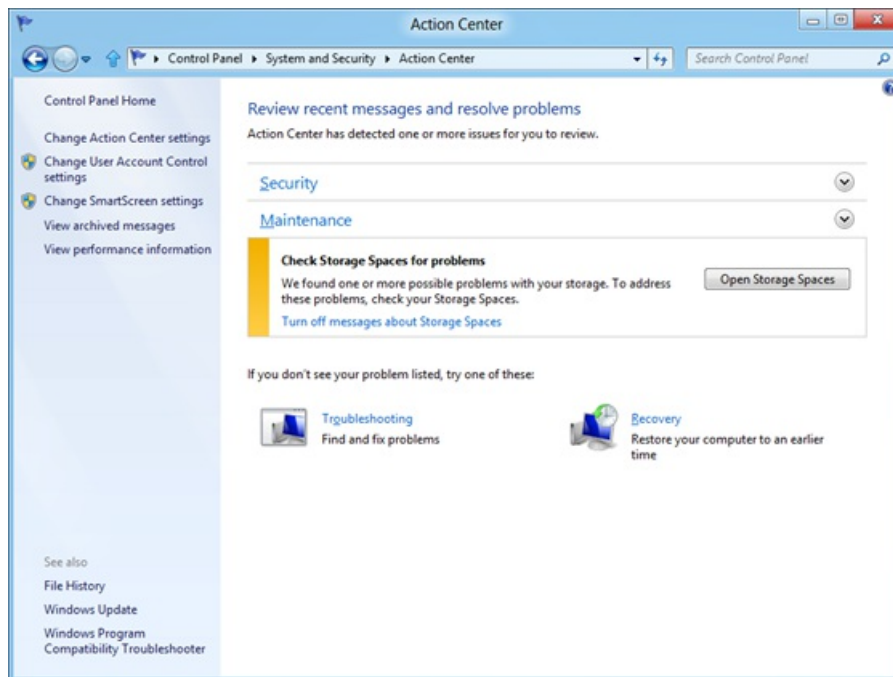
You can create a pool with only one disk. However, such a pool cannot contain any resilient spaces (i.e. mirrored or parity spaces). It can only contain a *simple* space which does not provide resiliency to failures. We do test pools comprising multiple hundreds of disks – such as you might see in a datacenter. There is no architectural limit to the number of disks comprising a pool.

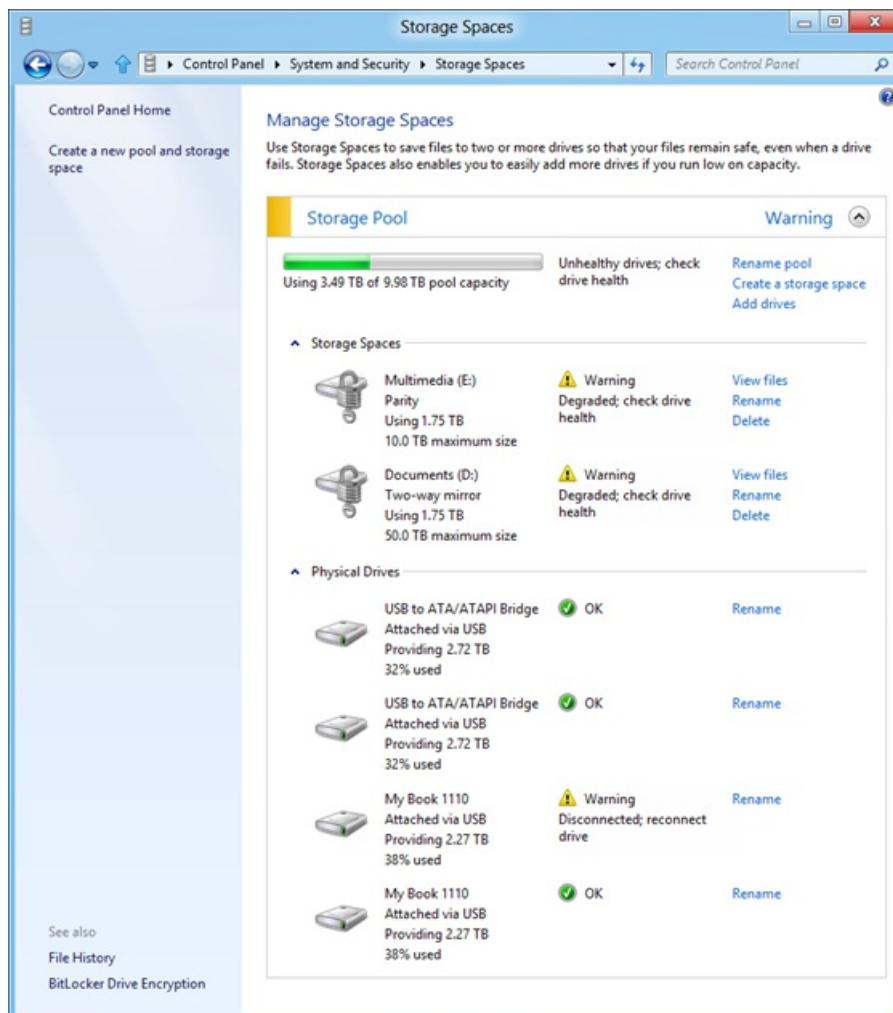
Q) How can I know which physical disk a space is on?

Through PowerShell, you can query the set of physical disks backing a particular space. Since all data is striped across all physical disks backing the space, you have this information.

Q) How will I know when a physical disk fails? How do I replace a failed disk?

If the physical disk is contained within an enclosure that supports the SCSI Enclosure Services protocol, we will activate a red LED (if present) next to the failed physical disk. A standard notification will pop up in the desktop. You can also see information about the failure in the Storage Spaces applet in Control Panel. Here is what that looks like:





Through PowerShell, you can also query disk health to determine if a disk has failed.

Once you've detected the failed disk, you can physically disconnect it at any time. Replacing a failed physical disk is easy – after removing the failed disk you simply connect the replacement disk to the PC, and then add the disk to the pool either via PowerShell or via Control Panel.

Q) How do I replace a working drive with a bigger one (or just cycle drives)? Does it require a “rebuild”?

As long as you have created mirrored or parity spaces, you can always simply remove a physical disk within the pool, and add a different (perhaps larger) one. Within a short period of time, the impacted spaces will automatically be resynchronized (the Storage Spaces design optimizes this operation to be faster than traditional RAID rebuilds). You can determine whether all spaces are healthy – i.e. data has been resynchronized so as to maintain the designated number of copies – either via Control Panel or via PowerShell commands.

Q) Can I trigger resynchronization myself?

Yes. If you don't want to wait for automatic resynchronization to start, you can choose the **Repair** command via PowerShell, which will initiate resynchronization so long as suitable replacement disks and/or spare capacity is available.

Q) What kind of disks will Storage Spaces work with? Are there any special requirements? What about custom enclosures housing these disks?

You can use Storage Spaces with any physical disk that otherwise works with Windows, connected via USB, SATA, or SAS. If the physical disks are connected via some custom enclosure (e.g. in JBOD configurations), Storage Spaces will utilize the SES protocol (if supported by the enclosure), to identify physical slots where the disks are located. When needed, Storage Spaces will also use SES to light up failure LEDs associated with physical disks (assuming that the enclosure has such LEDs). For Storage Spaces to use enclosure capabilities, the enclosure must conform to the Windows logo requirements. Enclosure vendors have been made aware of these requirements and we expect increasing conformance over time.

If your disks are housed within an enclosure, and if Storage Spaces either does not provide you with slot information associated with the physical disks or does not light up LEDs on the enclosure, you can assume that the enclosure does not conform to Windows logo requirements.

Q) Is there a defrag or CHKDSK equivalent for pools?

No. Storage Spaces optimally utilizes all physical disks. In the event that Storage Spaces metadata on a physical disk becomes corrupt (which will be obvious since the disk health will indicate a problem with the physical disk), you can treat the disk just as you would any other failed disk – simply remove it from the pool. If the physical disk is healthy, you can subsequently re-add it to the pool.

Q) How do I know how many mirrors a given file has?

If your file resides within a NTFS volume on a two-way mirrored space, two copies of all your file data will be maintained. If you configure a three-way mirrored space, there will be three copies.

Q) Can I pick which drive to use for mirrors? For example, if I know a particular disk is faster/better/newer?

Yes. In typical deployments, Storage Spaces will automatically select physical disks from the pool to back your spaces. However, if you so desire, you can manually specify a specific set of physical disks within your pool to back a particular space and thereby control allocation. You can do this via PowerShell options at the time you create the space.

Q) Can I change the maximum size of a space? Are there advantages or disadvantages to just making every space 50TB?

You can increase the logical size of a space at any time via Control Panel or PowerShell. Decreasing the logical size is not supported (or needed), given thin provisioning. It makes no difference whether you specify the initial logical size to be a smaller number (say 1TB) and grow it as needed, or set it to a very large number (say 50TB) right from the beginning. The latter may save you time and effort later.

Q) Can I change the slab size to something other than a multiple of 256MB?

No. The slab size is automatically determined by Storage Spaces based on a multitude of factors to deliver an optimal experience in terms of performance and availability.

Q) Does the pre-defined slab size result in sub-optimal utilization of capacity? For example, what if most of my files are very small? What if they're all large video files?

The slab size is an internal unit of capacity that we use for provisioning across multiple spaces within the same pool. Its value has no bearing on optimal storage of files, regardless of file size.

Q) Can I move a storage pool from one PC to another, once created? For example, if I have a cage with 6 removable drives?

Yes. Just connect the physical disks comprising the pool to the new PC.

Q) Say I have 3 external enclosures and I remove them one at a time. I then plug them into another Windows 8 PC in reverse order. Will the new PC think I have a broken pool or will it eventually catch up? What if I never plug in one of the enclosures?

You can plug enclosures back in in any order. When Storage Spaces detects a sufficient number of disks for quorum, it activates the pool and contained spaces. You can plug in more enclosures later. If the data on any disks becomes out of sync, Storage Spaces will automatically sync them. Even if you never plug in some enclosures, as long as Storage Spaces detects the minimum number of disks needed, you can continue working with your data. Both via PowerShell and via Control Panel, Storage Spaces informs you that a few physical disks are missing, thereby encouraging you to plug them back in.

Q) You mentioned that quorum for the pool requires a simple majority of healthy and connected physical disks. Does that mean I always need to have an even number of physical disks in the pool? Or do I need an odd number of physical disks? What about two-disk pools?

There is no requirement for an even or odd number of physical disks. Storage Spaces correctly handles two-disk pools and continues delivering resiliency to failures for a two-way mirrored space contained within such a pool, even if one physical disk fails or is disconnected.

Q) What happens when I plug physical disks comprising a pool into a Windows 7 machine?

Windows 7 does not support Storage Spaces and will treat the physical disks just as it would any disk with an unfamiliar partitioning scheme.

Building the next generation file system for Windows: ReFS

Steven Sinofsky | [2012-01-16T14:00:00+00:00](#)

We wanted to continue our dialog about data storage by talking about the next generation file system being introduced in Windows 8. Today, NTFS is the most widely used, advanced, and feature rich file system in broad use. But when you're reimagining Windows, as we are for Windows 8, we don't rest on past successes, and so with Windows 8 we are also introducing a newly engineered file system. ReFS, (which stands for Resilient File System), is built on the foundations of NTFS, so it maintains crucial compatibility while at the same time it has been architected and engineered for a new generation of storage technologies and scenarios. In Windows 8, ReFS will be introduced only as part of Windows Server 8, which is the same approach we have used for each and every file system introduction. Of course at the application level, ReFS stored data will be accessible from clients just as NTFS data would be. As you read this, let's not forget that NTFS is by far the industry's leading technology for file systems on PCs.

*This detailed architectural post was authored by **Surendra Verma**, a development manager on our Storage and File System team, though, as with every feature, a lot of folks contributed. We have also used the FAQ approach again in this post.*
--Steven

PS: Don't forget to track us on @buildwindows8 where we were providing some updates from CES.

In this blog post I'd like to talk about a new file system for Windows. This file system, which we call ReFS, has been designed from the ground up to meet a broad set of customer requirements, both today's and tomorrow's, for all the different ways that Windows is deployed.

The key goals of ReFS are:

- Maintain a high degree of compatibility with a subset of NTFS features that are widely adopted while deprecating others that provide limited value at the cost of system complexity and footprint.
- Verify and auto-correct data. Data can get corrupted due to a number of reasons and therefore must be verified and, when possible, corrected automatically. Metadata must not be written in place to avoid the possibility of "torn writes," which we will talk about in more detail below.
- Optimize for extreme scale. Use scalable structures for everything. Don't assume that disk-checking algorithms, in particular, can scale to the size of the entire file system
- Never take the file system offline. Assume that in the event of corruptions, it is advantageous to isolate the fault while allowing access to the rest of the volume. This is done while salvaging the maximum amount of data possible, all done live.
- Provide a full end-to-end resiliency architecture when used in conjunction with the Storage Spaces feature, which was co-designed and built in conjunction with ReFS.

The key features of ReFS are as follows (note that some of these features are provided in conjunction with Storage Spaces).

- Metadata integrity with checksums
- Integrity streams providing optional user data integrity
- Allocate on write transactional model for robust disk updates (also known as copy on write)
- Large volume, file and directory sizes
- Storage pooling and virtualization makes file system creation and management easy
- Data striping for performance (bandwidth can be managed) and redundancy for fault tolerance
- Disk scrubbing for protection against latent disk errors
- Resiliency to corruptions with "salvage" for maximum volume availability in all cases
- Shared storage pools across machines for additional failure tolerance and load balancing

In addition, ReFS inherits the features and semantics from NTFS including BitLocker encryption, access-control lists for security, USN journal, change notifications, symbolic links, junction points, mount points, reparse points, volume snapshots, file IDs, and oplocks.

And of course, data stored on ReFS is accessible through the same file access APIs on clients that are used on any operating system that can access today's NTFS volumes.

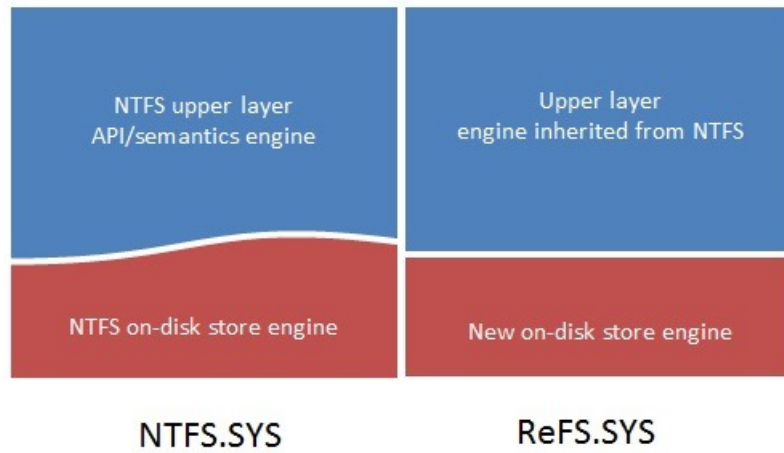
Key design attributes and features

Our design attributes are closely related to our goals. As we go through these attributes, keep in mind the history of producing file systems used by hundreds of millions of devices scaling from the smallest footprint machines to the largest data centers, from the smallest storage format to the largest multi-spindle format, from solid state storage to the largest drives and storage systems available. Yet at the same time, Windows file systems are accessed by the widest array of application and system software anywhere. ReFS takes that learning and builds on it. We didn't start from scratch, but reimagined it where it made sense and built on the right parts of NTFS where that made sense. Above all, we are delivering this in a pragmatic manner consistent with the delivery of a major file system—something only Microsoft has done at this scale.

Code reuse and compatibility

When we look at the file system API, this is the area where compatibility is the most critical and technically, the most challenging. Rewriting the code that implements file system semantics would not lead to the right level of compatibility and the issues introduced would be highly dependent on application code, call timing, and hardware. Therefore in building ReFS, we reused the code responsible for implementing the Windows file system semantics. This code implements the file system interface (read, write, open, close, change notification, etc.), maintains in-memory file and volume state, enforces security, and maintains memory caching and synchronization for file data. This reuse ensures a high degree of compatibility with the features of NTFS that we're carrying forward.

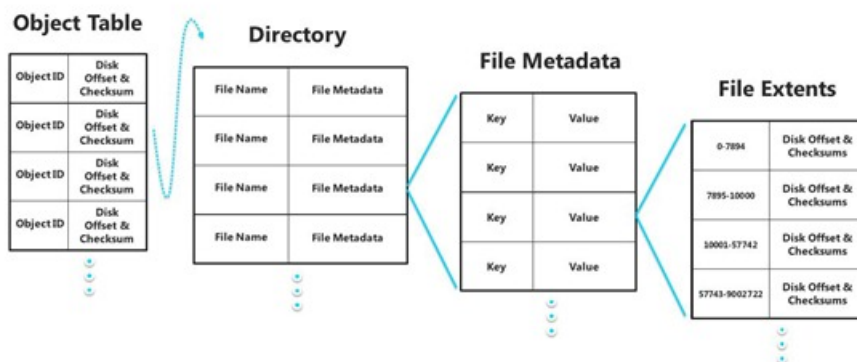
Underneath this reused portion, the NTFS version of the code-base uses a newly architected engine that implements on-disk structures such as the Master File Table (MFT) to represent files and directories. ReFS combines this reused code with a brand-new engine, where a significant portion of the innovation behind ReFS lies. Graphically, it looks like this:



Reliable and scalable on-disk structures

On-disk structures and their manipulation are handled by the on-disk storage engine. This exposes a generic key-value interface, which the layer above leverages to implement files, directories, etc. For its own implementation, the storage engine uses [B+ trees](#) exclusively. In fact, we utilize B+ trees as the single common on-disk structure to represent all information on the disk. Trees can be embedded within other trees (a child tree's root is stored within the row of a parent tree). On the disk, trees can be very large and multi-level or really compact with just a few keys and embedded in another structure. This ensures extreme scalability up and down for all aspects of the file system. Having a single structure significantly simplifies the system and reduces code. The new engine interface includes the notion of "tables" that are enumerable sets of key-value pairs. Most tables have a unique ID (called the object ID) by which they can be referenced. A special object table indexes all such tables in the system.

Now, let's look at how the common file system abstractions are constructed using tables.



File structures

As shown in the diagram above, directories are represented as tables. Because we implement tables using B+ trees, directories can scale efficiently, becoming very large. Files are implemented as tables embedded within a row of the parent directory, itself a table (represented as File Metadata in the diagram above). The rows within the File Metadata table represent the various file attributes. The file data extent locations are represented by an embedded stream table, which is a table of offset mappings (and, optionally, checksums). This means that the files and directories can be very large without a performance impact, eclipsing the limitations found in NTFS.

As expected, other global structures within the file system such as ACLs (Access Control Lists) are represented as tables rooted within the object table.

All disk space allocation is managed by a hierarchical allocator, which represents free space by tables of free space ranges. For scalability, there are three such tables – the large, medium and small allocators. These differ in the granularity of space they manage: for example, a medium allocator manages medium-sized chunks allocated from the large allocator. This makes disk allocation algorithms scale very well, and allows us the benefit of naturally collocating related metadata for better performance. The roots of these allocators as well as that of the object table are reachable from a well-known location on the disk. Some tables have allocators that are private to them, reducing contention and encouraging better allocation locality.

Apart from global system metadata tables, the entries in the object table refer to directories, since files are embedded within directories.

Robust disk update strategy

Updating the disk reliably and efficiently is one of the most important and challenging aspects of a file system design. We spent a lot of time evaluating various approaches. One of the approaches we considered and rejected was to implement a log structured file system. This approach is unsuitable for the type of general-purpose file system required by Windows. NTFS relies on a journal of transactions to ensure consistency on the disk. That approach updates metadata in-place on the disk and uses a journal on the side to keep track of changes that can be rolled back on errors and during recovery from a power loss. One of the benefits of this approach is that it maintains the metadata layout in place, which can be advantageous for read performance. The main disadvantages of a journaling system are that writes can get randomized and, more importantly, the act of updating the disk can corrupt previously written metadata if power is lost at the time of the write, a problem commonly known as *torn write*.

To maximize reliability and eliminate torn writes, we chose an allocate-on-write approach that never updates metadata in-place, but rather writes it to a different location in an atomic fashion. In some ways this borrows from a very old notion of “[shadow paging](#)” that is used to reliably update structures on the disk. Transactions are built on top of this allocate-on-write approach. Since the upper layer of ReFS is derived from NTFS, the new transaction model seamlessly leverages failure recovery logic already present, which has been tested and stabilized over many releases.

ReFS allocates metadata in a way that allows writes to be combined for related parts (for example, stream allocation, file attributes, file names, and directory pages) in fewer, larger I/Os, which is great for both spinning media and flash. At the same time a measure of read contiguity is maintained. The hierarchical allocation scheme is leveraged heavily here.

We perform significant testing where power is withdrawn from the system while the system is under extreme stress, and once the system is back up, all structures are examined for correctness. This testing is the ultimate measure of our success. We have achieved an unprecedented level of robustness in this test for Microsoft file systems. We believe this is industry-leading and fulfills our key design goals.

Resiliency to disk corruptions

As mentioned previously, one of our design goals was to detect and correct corruption. This not only ensures data integrity, but also improves system availability and online operation. Thus, all ReFS metadata is check-summed at the level of a B+ tree page, and the checksum is stored independently from the page itself. This allows us to detect all forms of disk corruption, including lost and misdirected writes and *bit rot* (degradation of data on the media). In addition, we have added an option where the contents of a file are check-summed as well. When this option, known as “integrity streams,” is enabled, ReFS always writes the file changes to a location different from the original one. This allocate-on-write technique ensures that pre-existing data is not lost due to the new write. The checksum update is done atomically with the data write, so that if power is lost during the write, we always have a consistently verifiable version of the file available whereby corruptions can be detected authoritatively.

We blogged about [Storage Spaces](#) a couple of weeks ago. We designed ReFS and Storage Spaces to complement each other, as two components of a complete storage system. We are making Storage Spaces available for NTFS (and client PCs) because there is great utility in that; the architectural layering supports this client-side approach while we adapt ReFS for usage on clients so that ultimately you’ll be able to use ReFS across both clients and servers.

In addition to improved performance, Storage Spaces protects data from partial and complete disk failures by maintaining copies on multiple disks. On read failures, Storage Spaces is able to read alternate copies, and on write failures (as well as complete media loss on read/write) it is able to reallocate data transparently. Many failures don’t involve media failure, but happen due to data corruptions, or lost and misdirected writes.

These are exactly the failures that ReFS can detect using checksums. Once ReFS detects such a failure, it interfaces with Storage Spaces to read all available copies of data and chooses the correct one based on checksum validation. It then tells Storage Spaces to fix the bad copies based on the good copies. All of this happens transparently from the point of view of the application. If ReFS is not running on top of a mirrored Storage Space, then it has no means to automatically repair the corruption. In that case it will simply log an event indicating that corruption was detected and fail the read if it is for file data. I’ll talk more about the impact of this on metadata later.

Checksums (64-bit) are always turned on for ReFS metadata, and assuming that the volume is hosted on a mirrored Storage Space, automatic correction is also always turned on. All integrity streams (see below) are protected in the same way. This creates an end-to-end high integrity solution for the customer, where relatively unreliable storage can be made highly reliable.

Integrity streams

Integrity streams protect file content against all forms of data corruption. Although this feature is valuable for many scenarios, it is not appropriate for some. For example, some applications prefer to manage their file storage carefully and rely on a particular file layout on the disk. Since integrity

streams reallocate blocks every time file content is changed, the file layout is too unpredictable for these applications. Database systems are excellent examples of this. Such applications also typically maintain their own checksums of file content and are able to verify and correct data by direct interaction with Storage Spaces APIs.

For those cases where a particular file layout is required, we provide mechanisms and APIs to control this setting at various levels of granularity.

At the most basic level, integrity is an attribute of a file (`FILE_ATTRIBUTE_INTEGRITY_STREAM`). It is also an attribute of a directory. When present in a directory, it is inherited by all files and directories created inside the directory. For convenience, you can use the “format” command to specify this for the root directory of a volume at format time. Setting it on the root ensures that it propagates by default to every file and directory on the volume. For example:

```
D:\>format /fs:refs /q /i:enable <volume>
```

```
D:\>format /fs:refs /q /i:disable <volume>
```

By default, when the `/i` switch is not specified, the behavior that the system chooses depends on whether the volume resides on a mirrored space. On a mirrored space, integrity is enabled because we expect the benefits to significantly outweigh the costs. Applications can always override this programmatically for individual files.

Battling “bit rot”

As we described earlier, the combination of ReFS and Storage Spaces provides a high degree of data resiliency in the presence of disk corruptions and storage failures. A form of data loss that is harder to detect and deal with happens due to “bit rot,” where parts of the disk develop corruptions over time that go largely undetected since those parts are not read frequently. By the time they are read and detected, the alternate copies may have also been corrupted or lost due to other failures.

In order to deal with bit rot, we have added a system task that periodically scrubs all metadata and Integrity Stream data on a ReFS volume residing on a mirrored Storage Space. Scrubbing involves reading all the redundant copies and validating their correctness using the ReFS checksums. If checksums mismatch, bad copies are fixed using good ones.

The file attribute `FILE_ATTRIBUTE_NO_SCRUB_DATA` indicates that the scrubber should skip the file. This attribute is useful for those applications that maintain their own integrity information, when the application developer wants tighter control over when and how those files are scrubbed.

The `Integrity.exe` command line tool is a powerful way to manage the integrity and scrubbing policies.

When all else fails...continued volume availability

We expect many customers to use ReFS in conjunction with mirrored Storage Spaces, in which case corruptions will be automatically and transparently fixed. But there are cases, admittedly rare, when even a volume on a mirrored space can get corrupted – for example faulty system memory can corrupt data, which can then find its way to the disk and corrupt all redundant copies. In addition, some customers may not choose to use a mirrored storage space underneath ReFS.

For these cases where the volume gets corrupted, ReFS implements “salvage,” a feature that removes the corrupt data from the namespace on a live volume. The intention behind this feature is to ensure that non-repairable corruption does not adversely affect the availability of good data. If, for example, a single file in a directory were to become corrupt and could not be automatically repaired, ReFS will remove that file from the file system namespace while salvaging the rest of the volume. This operation can typically be completed in under a second.

Normally, the file system cannot open or delete a corrupt file, making it impossible for an administrator to respond. But because ReFS can still salvage the corrupt data, the administrator is able to recover that file from a backup or have the application re-create it without taking the file system offline. This key innovation ensures that we do not need to run an expensive offline disk checking and correcting tool, and allows for very large data volumes to be deployed without risking large offline periods due to corruption.

A clean fit into the Windows storage stack

We knew we had to design for maximum flexibility and compatibility. We designed ReFS to plug into the storage stack just like another file system, to maximize compatibility with the other layers around it. For example, it can seamlessly leverage BitLocker encryption, Access Control Lists for security, USN journal, change notifications, symbolic links, junction points, mount points, reparse points, volume snapshots, file IDs, and oplocks. We expect most file system filters to work seamlessly with ReFS with little or no modification. Our testing bore this out; for example, we were able to validate the functionality of the existing Forefront antivirus solution.

Some filters that depend on the NTFS physical format will need greater modification. We run an extensive compatibility program where we test our file systems with third-party antivirus, backup, and other such software. We are doing the same with ReFS and will work with our key partners to address any incompatibilities that we discover. This is something we have done before and is not unique to ReFS.

An aspect of flexibility worth noting is that although ReFS and Storage Spaces work well together, they are designed to run independently of each other. This provides maximum deployment flexibility for both components without unnecessarily limiting each other. Or said another way, there are reliability and performance tradeoffs that can be made in choosing a complete storage solution, including deploying ReFS with underlying storage

from our partners.

With Storage Spaces, a storage pool can be shared by multiple machines and the virtual disks can seamlessly transition between them, providing additional resiliency to failures. Because of the way we have architected the system, ReFS can seamlessly take advantage of this.

Usage

We have tested ReFS using a sophisticated and vast set of tens of thousands of tests that have been developed over two decades for NTFS. These tests simulate and exceed the requirements of the deployments we expect in terms of stress on the system, failures such as power loss, scalability, and performance. Therefore, ReFS is ready to be deployment-tested in a managed environment. Being the first version of a major file system, we do suggest just a bit of caution. We do not characterize ReFS in Windows 8 as a “beta” feature. It will be a production-ready release when Windows 8 comes out of beta, with the caveat that nothing is more important than the reliability of data. So, unlike any other aspect of a system, this is one where a conservative approach to initial deployment and testing is mandatory.

With this in mind, we will implement ReFS in a staged evolution of the feature: first as a storage system for Windows Server, then as storage for clients, and then ultimately as a boot volume. This is the same approach we have used with new file systems in the past.

Initially, our primary test focus will be running ReFS as a file server. We expect customers to benefit from using it as a file server, especially on a mirrored Storage Space. We also plan to work with our storage partners to integrate it with their storage solutions.

Conclusion

Along with Storage Spaces, ReFS forms the foundation of storage on Windows for the next decade or more. We believe this significantly advances our state of the art for storage. Together, Storage Spaces and ReFS have been architected with headroom to innovate further, and we expect that we will see ReFS as the next massively deployed file system.

-- Surendra

FAQ:

Q) Why is it named ReFS?

ReFS stands for Resilient File System. Although it is designed to be better in many dimensions, resiliency stands out as one of its most prominent features.

Q) What are the capacity limits of ReFS?

The table below shows the capacity limits of the on-disk format. Other concerns may determine some practical limits, such as the system configuration (for example, the amount of memory), limits set by various system components, as well as time taken to populate data sets, backup times, etc.

Attribute	Limit based on the on-disk format
Maximum size of a single file	2 ⁶⁴ -1 bytes
Maximum size of a single volume	Format supports 2 ⁷⁸ bytes with 16KB cluster size (2 ⁶⁴ * 16 * 2 ¹⁰). Windows stack addressing allows 2 ⁶⁴ bytes
Maximum number of files in a directory	2 ⁶⁴
Maximum number of directories in a volume	2 ⁶⁴
Maximum file name length	32K-255 unicode characters (for compatibility this was made consistent with NTFS for the RTM product)
Maximum path length	32K

Maximum size of any storage pool 4 PB

Maximum number of storage pools in a system No limit

Maximum number of spaces in a storage pool No limit

Q) Can I convert data between NTFS and ReFS?

In Windows 8 there is no way to convert data in place. Data can be copied. This was an intentional design decision given the size of data sets that we see today and how impractical it would be to do this conversion in place, in addition to the likely change in architected approach before and after conversion.

Q) Can I boot from ReFS in Windows Server 8?

No, this is not implemented or supported.

Q) Can ReFS be used on removable media or drives?

No, this is not implemented or supported.

Q) What semantics or features of NTFS are no longer supported on ReFS?

The NTFS features we have chosen to not support in ReFS are: named streams, object IDs, short names, compression, file level encryption (EFS), user data transactions, sparse, hard-links, extended attributes, and quotas.

Q) What about parity spaces and ReFS?

ReFS is supported on the fault resiliency options provided by Storage Spaces. In Windows Server 8, automatic data correction is implemented for mirrored spaces only.

Q) Is clustering supported?

Failover clustering is supported, whereby individual volumes can failover across machines. In addition, shared storage pools in a cluster are supported.

Q) What about RAID? How do I use ReFS capabilities of striping, mirroring, or other forms of RAID? Does ReFS deliver the read performance needed for video, for example?

ReFS leverages the data redundancy capabilities of Storage Spaces, which include striped mirrors and parity. The read performance of ReFS is expected to be similar to that of NTFS, with which it shares a lot of the relevant code. It will be great at streaming data.

Q) How come ReFS does not have deduplication, second level caching between DRAM & storage, and writable snapshots?

ReFS does not itself offer deduplication. One side effect of its familiar, pluggable, file system architecture is that other deduplication products will be able to plug into ReFS the same way they do with NTFS.

ReFS does not explicitly implement a second-level cache, but customers can use third-party solutions for this.

ReFS and VSS work together to provide snapshots in a manner consistent with NTFS in Windows environments. For now, they don't support writable snapshots or snapshots larger than 64TB.

Engineering Windows 8 for mobile networks

Steven Sinofsky | [2012-01-20T10:30:00+00:00](#)

In this post, we dig into the details of how we have re-engineered the wireless networking stack to optimize it for both mobile broadband and Wi-Fi networks. We've done a ton of work to enable mobile broadband providers to make it easy for you to use 3G and 4G connectivity along with Wi-Fi in Windows 8. In addition to this architectural work, we've worked on keeping Windows connected to a network even when in a low-power state (when the screen is off, for example) when running on supporting architectures/PCs. You can learn more about this in the //build/ sessions on [connected standby](#). Billy Anders, a group program manager on our devices and networking team, authored this post.

--Steven

People want similar mobility on their PCs as they get on their smartphones.

It is unlikely that your end goal is just to get connected to the Internet. Instead, connecting to the Internet is a step (or a hurdle) towards what you really want to do, like surf, socialize, or explore, and you would prefer that your PC is connected and ready for you to use whenever you want and wherever you are.

We looked at the fundamentals of wireless connectivity and re-engineered Windows 8 for a mobile and wireless future, going beyond incremental improvements. This is a good example of work that requires new hardware to work in concert with new software in order to realize its full potential.

Simplifying your mobile broadband experience

We knew that if we were to give you true mobility, that Wi-Fi alone would not be enough. Therefore, for Windows 8, we fully developed and integrated *mobile broadband* (MB) as a first-class connectivity experience within Windows – right alongside Wi-Fi.

We first included mobile broadband in Windows 7, but if you were a mobile broadband user, you likely had a number of hurdles to overcome before connecting with mobile broadband. Yes, you needed the requisite mobile broadband hardware (e.g., mobile broadband dongle or embedded module and SIM) and data plan, but you also needed to locate and install third-party device drivers, and in some cases software, before ever getting your first connection. If the drivers for your device and software from your mobile operator were not available locally, you had to find another connection type (perhaps Wi-Fi) to the Internet to search for software on the websites of the PC maker or mobile operator. This placed a sizable hurdle in front of users trying to connect with mobile broadband, right when they most needed that connection.

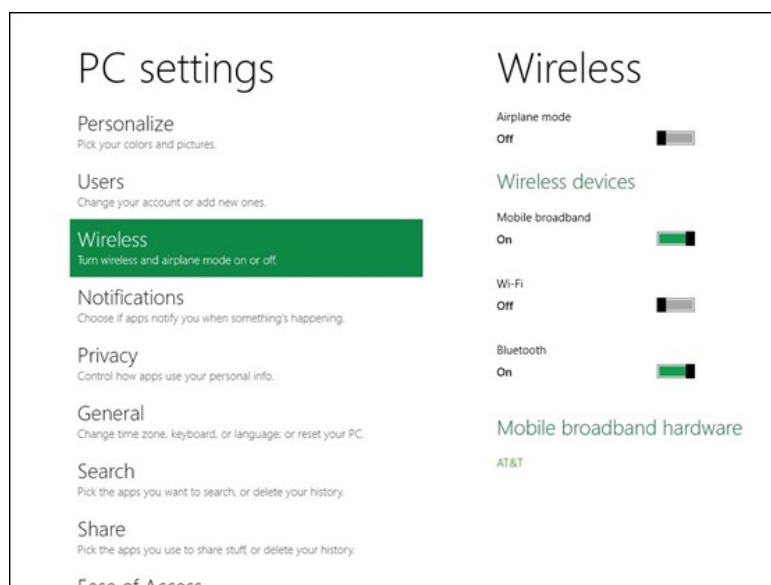
We wanted to eliminate the guesswork in locating and installing device drivers for mobile broadband. We did this by working with our mobile operator and mobile broadband hardware partners across the industry, designing a hardware specification that device makers can incorporate into their device hardware. In Windows 8, we developed an in-box mobile broadband class driver that works with all of these devices and eliminates your need for additional device driver software. You just plug in the device and connect. The driver stays up to date via Windows Update, ensuring you have a reliable mobile broadband experience.

The USB Implementers Forum (USB-IF) recently approved the Mobile Broadband Interface Model (MBIM) specification as a standard, and major device makers have already begun adopting this standard into their device designs, including some designed for other operating systems. For more information on the specification, see the [USB-IF press release](#).

Helping you manage your connections and radios

Typically, mobile broadband devices come with radio and connection management software. Device manufacturers, PC manufacturers, and mobile operators all develop, distribute, and support these applications for you to connect to their networks, turn radios on and off, configure connection settings, and get contact information for help and support. Prior to Windows 8, you needed these applications to compensate for functionality not provided natively in Windows. This additional software confused and frustrated users by conflicting with the Windows connection manager, showing different networks, network status, and a separate user interface. Windows 8 eliminates this confusion by providing simple, intuitive, and fully integrated radio and connection management.

The new Windows 8 network settings allow you to turn individual radios on and off (Wi-Fi, mobile broadband, or Bluetooth), as well as disable all radios at once with the new "airplane mode." Windows 8 provides native radio management to eliminate the conflicts and confusion, and to provide a consistent experience for controlling your radios without the need to install additional software. This is new for PCs even though it has obviously long been available on today's mobile phones (or Windows Mobile phones, going way back).



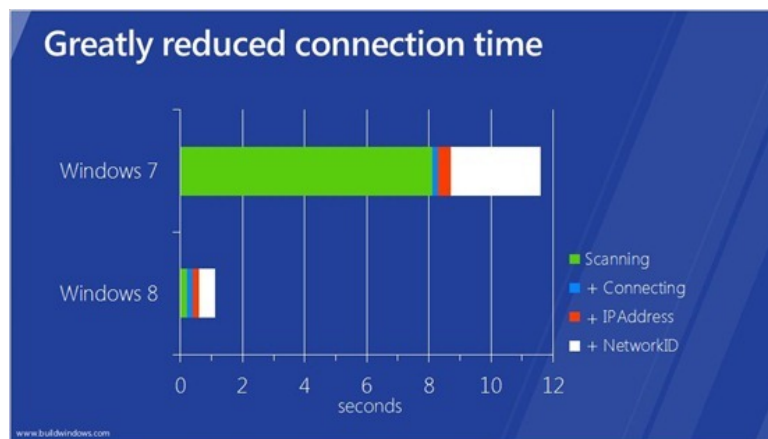
You can turn airplane mode on or off in one click

The new wireless network settings in Windows 8 allow you to see and connect to all available MB and Wi-Fi networks from one convenient user interface. We made sure that this interface is consistent and allows you to think less about which network you want to connect. Windows does this by starting with the right default behaviors, and then it gets smarter by learning your network preferences over time.

One of those default behaviors is to prioritize Wi-Fi networks over broadband whenever one of your preferred Wi-Fi networks is available. Wi-Fi networks are typically faster, with lower latency, and have higher data caps (if they are not free). When you connect to a Wi-Fi network, we automatically disconnect you from your mobile broadband network and, when appropriate, power down the mobile broadband device, which also increases battery life. If no preferred Wi-Fi network is available, we automatically reconnect you to your preferred mobile broadband network.

To make sure we connect to the right network when multiple networks are available, Windows maintains an ordered list of your preferred networks based on your explicit connect and disconnect actions, as well as the network type. For example, if you manually disconnect from a network, Windows will no longer automatically connect to that network. If, while connected to one network, you decide to connect to a different network, Windows will move the new network higher in your preferred networks list. Windows automatically learns your preferences in order to manage this list for you.

When you resume from standby, Windows can also reconnect you faster to your preferred Wi-Fi networks by optimizing operations in the networking stack, and providing your network list, connection information, and hints to your Wi-Fi adapter. Now when your PC resumes from standby, your Wi-Fi adapter already has all the information it needs to connect to your preferred Wi-Fi networks. This means **you can reconnect your PC to a Wi-Fi network from standby in about a second** – oftentimes before your display is even ready. You do not have to do anything special for this – Windows just learns which networks you prefer and manages everything for you. This work was a major part of the architectural work we did in the networking stack and with our hardware partners.



Getting connected to mobile broadband

Even with its broad availability, Wi-Fi by itself does not enable the ubiquitous Internet access that users increasingly want. True mobility requires mobile broadband, which provides connectivity over cellular networks (the same networks as your smartphone). However, just including mobile broadband in Windows 8 was not enough. We also wanted to remove any hurdles to getting you connected to mobile broadband, making it simpler, more intuitive, and more like Wi-Fi.

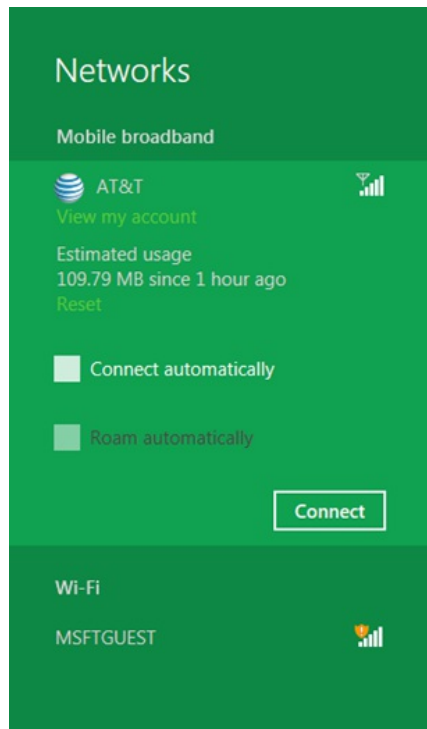
We made things simpler and more intuitive by fully integrating mobile broadband into Windows 8. When you're ready to connect to a mobile broadband network, you simply insert your mobile broadband device or SIM card into your Windows 8 PC and we take care of the setup.

If you have a carrier-unlocked mobile broadband device that supports carrier switching (this includes most mobile broadband users outside the US), Windows 8 has native support that allows you to select and connect to any supported carrier from within the Windows UI.

Selecting from available carriers (with supported hardware)

We've already talked about how we removed the need to install a driver, or a radio and connection manager. We also automatically identify which mobile operator is associated with your device (or SIM card), brand it in the Windows connection manager with the mobile operator's logo, configure the PC for connecting to the mobile operator's network, and download the operator's mobile broadband app (if they have one) from the Windows Store.

If you purchased and activated a data plan along with your SIM or mobile broadband device, all you need to do is connect to the network and we get out of the way, allowing you to do what you want to do.



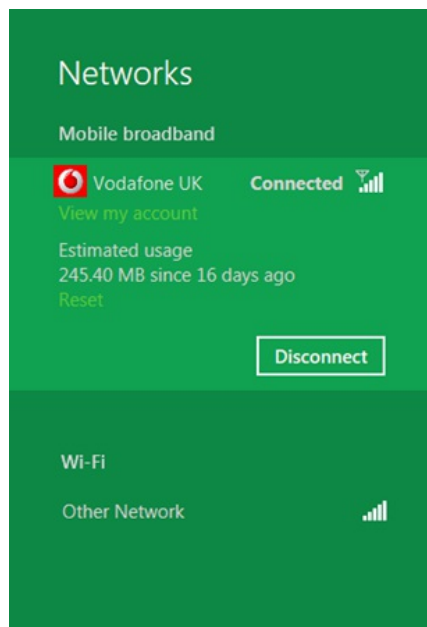
Getting connected via mobile broadband using an AT&T SIM card

If you don't already have a data plan and would like to purchase one, then simply click the "Connect" button for the mobile operator you want, and we automatically direct you to their mobile broadband app or website, where you can select a data plan (for example, a time-based, limit-based, or subscription-based plan).



AT&T's new mobile broadband app walks you through purchasing a data plan

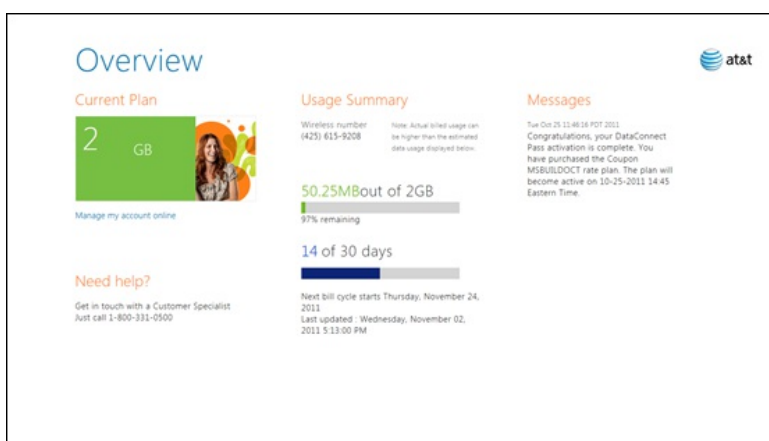
After you've purchased your plan, your mobile operator provisions your PC over the air for their network, including information about your data plan details and Wi-Fi hotspots.



Usage details are shown with the connected account

Behind the scenes, Windows identifies the mobile broadband subscriber information, looks up the mobile operator in the new Access Point Name (APN) database, and pre-provisions the system to connect to the operator's network. Meanwhile, your core connection experience stays the same.

The operator's mobile broadband app is available via the "View my account" link, or from the app's tile on the Start screen. Here, you can see how much data you've used, pay your bill, manage your account, and get customer support.



AT&T mobile broadband app, account overview

Avoiding "bill shock"

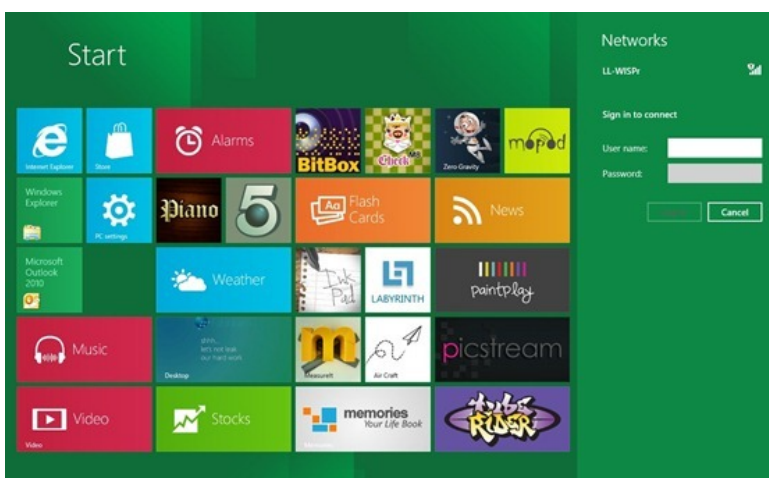
Many of us have read headlines about people receiving surprisingly expensive bills from their mobile operators. The industry has termed this [bill shock](#), and the problem has received enough attention that some governments have begun taking regulatory steps that ask mobile operators to alert their customers when their data usage reaches a certain threshold. Today, mobile operators all have different ways of responding when subscribers exceed their data usage allotment. An operator may block your Internet access, throttle (slow down) your data speed, or simply begin charging you per kilobyte or megabyte. If you are unaware that you are over your data usage limit, you will likely continue using your data plan and rack up additional charges, resulting in shock when you receive your bill.

Prior to Windows 8, we maintained consistent behavior on all types of networks relative to bandwidth usage. With Windows 8, we now take the cost of the network into consideration: we assume that mobile broadband networks have restrictive data caps with higher overage costs (vs. Wi-Fi), and adjust networking behavior with these *metered* networks accordingly.

As mentioned earlier, we automatically disconnect from mobile broadband and connect you to your preferred Wi-Fi networks whenever they're available. This reduces your data usage on mobile broadband when possible.

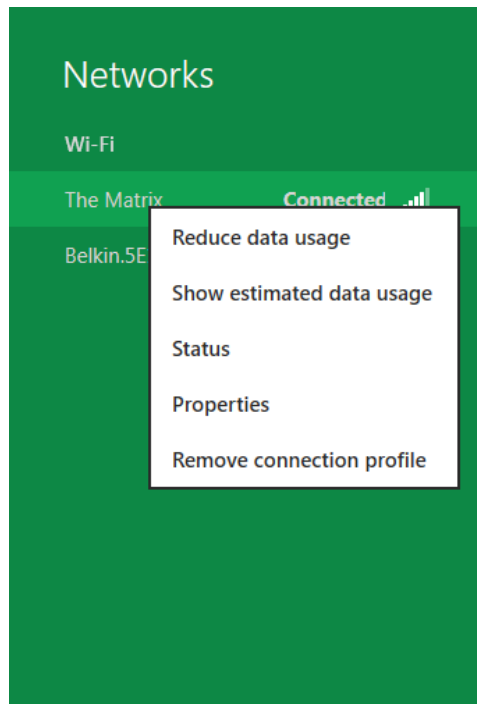
Because many of us use public Wi-Fi, Windows 8 includes support for popular Wi-Fi hotspot authentication types, including [WISPr](#) (Wireless Internet Services Provider roaming), [EAP-SIM/AKA/AKA Prime](#) (SIM-based authentication), and [EAP-TTLS](#) (popular on university campuses). Windows manages the authentication for you when you come within range of a Wi-Fi network that uses one of these methods, so you won't have to re-authenticate each time (for instance, by going to a web page). This means you get the same automatic behavior at a public Wi-Fi hotspot as you would at home or the office.

On a PC that has both mobile broadband and Wi-Fi, we'll move you from MB to the less costly Wi-Fi network automatically whenever Wi-Fi is available, again reducing your mobile broadband usage and your potential for bill shock.



Another way we optimize your bandwidth usage is by changing the Windows Update download behavior. For a majority of users, who have turned on automatic updating, Windows Update will defer the background download of all updates until you connect to a *non-metered* network, such as your home broadband connection. There is one exception, as noted in our earlier [post on Windows Update](#), and that is in the case of a critical security update to fix a worm-like vulnerability (e.g., a Blaster worm). In that case, Windows Update will download the update regardless of the network type. You can always override the deferred download by launching Windows Update and manually initiating the download of updates at a time more convenient to you. Again, you are in full control of your device.

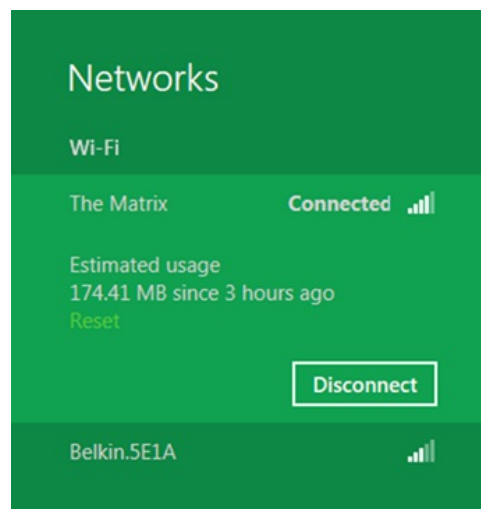
We recognize that most fixed-line broadband plans also have data caps and overage fees. Those data caps are typically much higher than mobile broadband, and therefore we do not change the behavior for these connections. You are always in control and can always mark any wireless network as metered or unmetered by selecting "reduce data usage" in the right-click (or tap and hold) menu for that network.



Marking the Wi-Fi connection as “metered”

We also want Windows applications to behave well on metered networks, so we’ve provided a new set of developer APIs within the `ConnectionCost` class of the `Windows.Networking.Connectivity` namespace. If you are an application developer, we encourage you to leverage these APIs and adapt the behavior of your app, such as allowing a low-definition vs. high-definition video stream, or a header-only vs. full-sync of email, depending on the network type. We believe that this adaptive behavior is critical, as it results in actual cost savings for end users. All Metro style apps in the Windows Store must implement these APIs if they use the network.

Even with Windows and other applications behaving smartly on the network, you still may want to know how much data you have consumed. Windows 8 provides local data usage counters right within the network settings. These counters provide real time local data usage estimates for Wi-Fi and mobile broadband network connections.



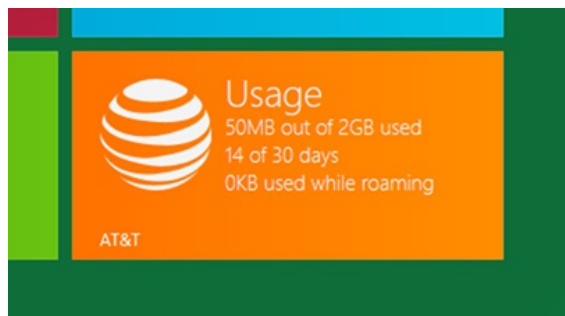
Local data usage estimates

The local counters keep track of the amount of data used on each individual network type so you don’t have to. You can reset the counter whenever you want, which may be useful if you want to monitor your usage month-to-month or even within a session. Although you should think of the local data counters as a quick way to determine your usage, they are not a substitute for what mobile operators report as their usage, which may vary slightly, and should be available in the operator’s app.

Another way we help you manage your mobile broadband data usage is by allowing mobile operators to alert you as you approach your bandwidth cap. Some countries have already begun to mandate that operators send messages to subscribers as they approach their bandwidth cap, or once they begin roaming to a different network. The mobile operator sends you an SMS or USSD alert as you approach your bandwidth cap (e.g., 70% used, 85% used, etc.), and the MB operator’s app notifies you and updates its Start screen tile. The following screen shots show what is already available in the Windows Developer Preview (and on the Samsung Preview PC that had an AT&T SIM and plan).



Data usage notification, bottom right.



Data usage information on the mobile operator's app tile

The [Windows 8 task manager](#) provides more granular information if you want to know how much data a particular app has consumed on the network. Here, you can see the approximate active and historical data consumption of any process over metered and non-metered networks. With this information, you can take control by identifying which apps are consuming the most bandwidth and taking action if needed.

Application	CPU (Time)	Network (MB)	Metered network (MB)	Tiles (MB)
Picstream	0:00:05	6.6	6.4	0.2
NearMe	0:00:54	2.1	2.1	0
AT&T Mobile Broadband	0:10:53	1.9	0.1	0
Stocks	0:00:15	0.2	0.2	0.1
Tweet@rama	0:00:00	0.1	0.1	0.1
News	0:00:00	0.1	0	0.1
5 in a row	0:00:00	0	0	0
Air Craft	0:00:00	0	0	0
Alarms	0:00:06	0	0	0
BitBox	0:00:00	0	0	0
BUILD	0:00:00	0	0	0
CheckM8	0:00:00	0	0	0
Control Panel	0:00:03	0	0	0

Data consumption information in the Windows Task Manager

Here's a short video that demonstrates some of the new wireless networking features and enhancements in Windows 8.

Your browser doesn't support HTML5 video.
 Download this video to view it in your favorite media player:
[High quality MP4](#) | [Lower quality MP4](#)

We designed Windows 8 with you—and mobility—in mind. We set out to simplify your experience with getting and staying connected across mobile broadband and Wi-Fi networks, removing hurdles and whenever possible, doing the right things automatically for you.

-- Billy Anders

Supporting sensors in Windows 8

Steven Sinofsky | [2012-01-24T10:00:00+00:00](#)

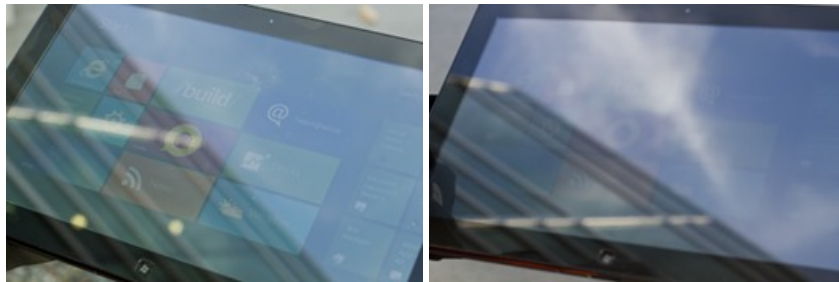
Recent advances in sensor technology are catalysts for the acceleration and evolution of user experiences on PCs. The ability to react to changes in ambient light, motion, human proximity, and location are becoming common and essential elements of the computing experience. Even something simple—like an ambient light sensor to adjust display brightness in a room with changing light—is potentially a basic scenario for desktop PCs. Of course, we also want to make sure you have full control over the use of these peripherals, since we know that different sensors leave open opportunities for risk or abuse that some folks might not be comfortable with. This post looks at the details of supporting sensors in Windows 8 and was authored by Gavin Gear, a PM on the Device Connectivity team.

—Steven

The first thing we explored about sensors was how Windows 8 should use them at the system level, to adapt the PC to the environment while preserving battery life.

Adaptive brightness

The first system feature was automatic display brightness control, or what we call “adaptive brightness.” This was a feature that we first introduced in Windows 7 using ambient light sensors (ALS), and is targeted at mobile form factors like slates, convertibles, and laptops. With today’s display panels supporting brightness levels at approximately twice the intensity of what was common just a few years ago, this feature is more important than ever. By dynamically controlling screen brightness based on changing ambient light conditions, we can optimize the level of reading comfort, and save battery life when the screen is dimmed in darker environments.

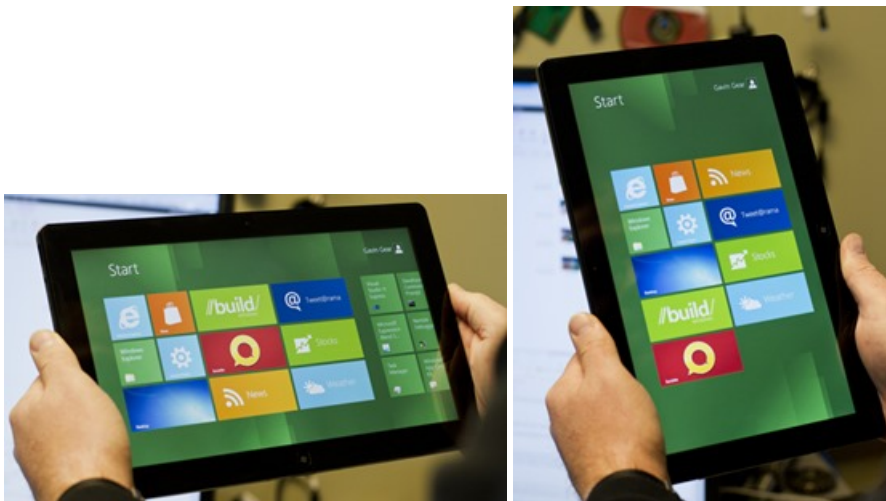


A tablet PC in harsh outdoor lighting with adaptive brightness (left), and without (right)

You can see here that adaptive brightness helps you see content on the screen more clearly, since the screen automatically gets brighter when the tablet enters a bright environment. And for those of you who use your desktop PCs in a sunny room, you know this same thing can happen at different times of the day in different seasons.

Automatic screen rotation

Many smartphones and other mobile devices have established the expectation that when you rotate the device, the graphic display will also rotate and adapt to the new orientation (including adapting to aspect ratio changes). Data from an accelerometer allows the device to determine its basic orientation. By automatically rotating the screen, people can use their devices (primarily slates and convertibles) in a more natural and intuitive way, without needing to manually rotate the screen with software controls or hardware buttons.



Developer support for sensors

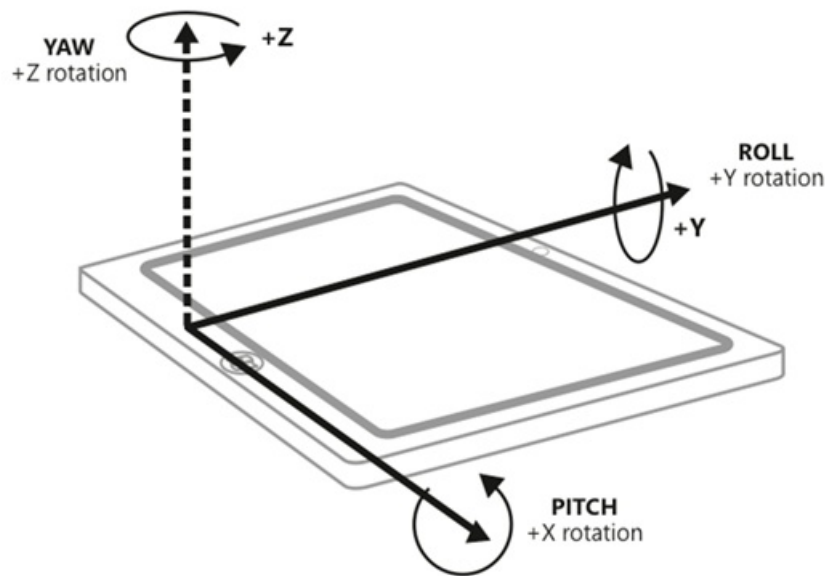
Beyond figuring out the basics for how a Windows 8 system might use sensors, we also needed to think about how apps might use sensors. We looked at a variety of examples of sensor-enabled apps including games, commercial applications, tools, and utilities, to help us determine which scenarios to support.

First on the list was the ability for apps to understand motion and screen rotation. This requires an accelerometer – a device that can be used to measure the force due to gravity, and the motion of the device itself. But most scenarios require more than just an understanding of motion and gravity. Orientation is also an important requirement for many applications. To enable a PC to understand orientation we needed to integrate the functionality of a compass.

Supporting a compass would at minimum require a 3D accelerometer (which measures acceleration on three axes) and a 3D magnetometer (which measures magnetic field strengths on 3 axes). This combination of sensors is called a *6-axis motion and orientation sensing system*, and can support a basic tilt-compensated compass, screen rotation, and certain casual game apps like a labyrinth style game. However, in our testing and prototyping, we found the 6-axis motion sensing system has two key drawbacks: sporadic compass inaccuracy, and a lack of the responsiveness required by 3D interactive games.

Recently, a new type of sensor has started to emerge on phone platforms – the gyro sensor. Gyro sensors measure angular speed, typically along 3 axes. You can also use the data from gyro sensors to increase the responsiveness and accuracy of 3D motion-sensing systems. A gyro sensor is very sensitive, but it lacks any form of orientation reference (such as gravity or north heading).

This diagram shows how gyro data is represented as a set of three rotations along the three primary axes for the device:

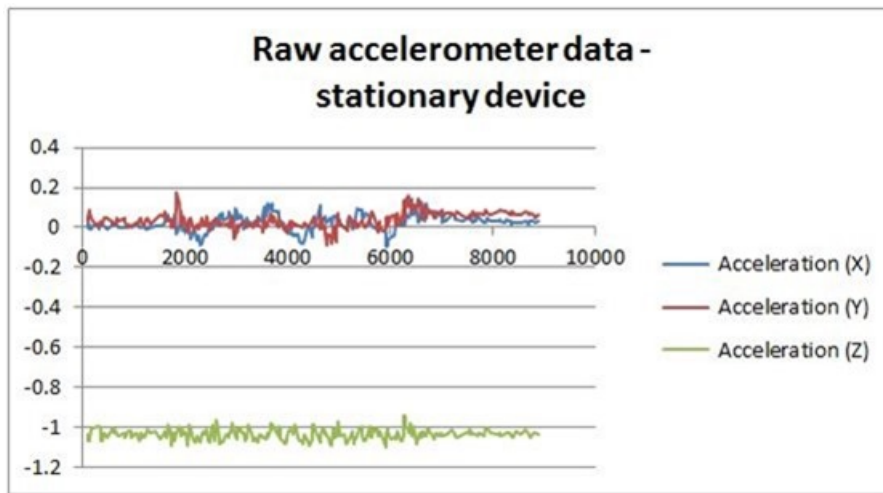


Initially, some thought that the need for such sensors was scoped to very few apps, such as specialized games. But the more we examined the 3D motion and orientation sensing problem, the more we realized that applications are much more immersive and attractive if they react to the kind of motion humans can easily understand, such as shakes, twists, and rotations in multiple dimensions. With these kinds of sensors it would certainly be possible to build very immersive 3D games, but it would also enable lots of other apps to more naturally respond to input from a variety of motions, including mapping and navigation applications, measuring utilities, interactive (between two machines) applications, and simple apps like casual games.

Engineering challenges

We started our exploration into motion apps by prototyping some 3D experiences. The first challenge was to map the physical orientation of the device directly to a virtual 3D environment in the app. We decided to model a simple augmented reality experience by emulating a tablet as a window into a virtual world. The concept was fairly simple: when you move the device while looking at the screen, the virtual environment (the inside of a room) would appear to stay stationary.

Initially, we tried an experiment using the accelerometer to map up and down movement of the device to up and down movement of the 3D environment in response. When you hold the device still, the scene should remain stable. When you tilt the device, the view should tilt up or down. Right away we encountered an issue: “noise” in the data from the accelerometer sensor was causing jittery movement of the 3D environment even when the device was held stationary. We were able to see this noise clearly by capturing accelerometer data and charting it.



Without noise, the lines on the chart would be straight, with no vertical deviation. The conventional way to remove such noise is to apply a low-pass filter to the raw data stream. When we implemented this mitigation in our prototype, the resultant motion was smooth and stable (jitter-free). But the low-pass filter introduced another problem: the app lost responsiveness and felt sluggish when responding to motion. We needed a way to compensate for this jitter without reducing responsiveness.

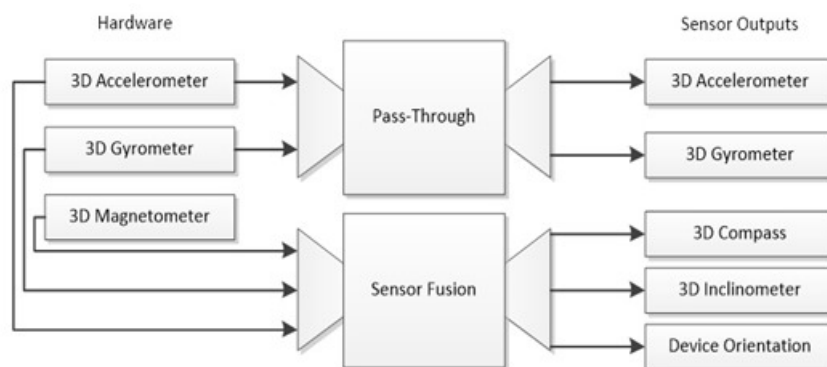
The next experiment was to provide the ability to “look left” and “look right” in our virtual 3D environment app. We used a 6-axis compass solution (3D accelerometer + 3D magnetometer) to support this type of movement. Although this *kind of* worked, the movement was not consistent due to the general instability of the 6-axis compass. It was also challenging to blend the up-and-down movement with the left-and-right movement.

From these experiments it was clear that this combination of sensors could not provide the fluid and responsive experience we wanted. The accelerometer sensor was not providing clean data, and could not be used alone to determine device orientation. The magnetometer was slow to update and was susceptible to electromagnetic interference (think of a compass needle that sticks in one position occasionally). We had yet to experiment with the gyro sensors, but because gyros could only determine rotational speed, it wasn’t clear how they could help.

Creating “sensor fusion”

But further experimentation demonstrated that using all three sensors together *could* solve the problem. It turns out that an accelerometer, magnetometer, and a gyro can complement each-other’s weaknesses, effectively filling in gaps in data and data responsiveness. Using a combination of these sensors it is possible to create a better, more responsive, and more fluid experience than the sensors can provide individually. Combining the input of multiple sensors to produce better overall results is a process we call *sensor fusion*.

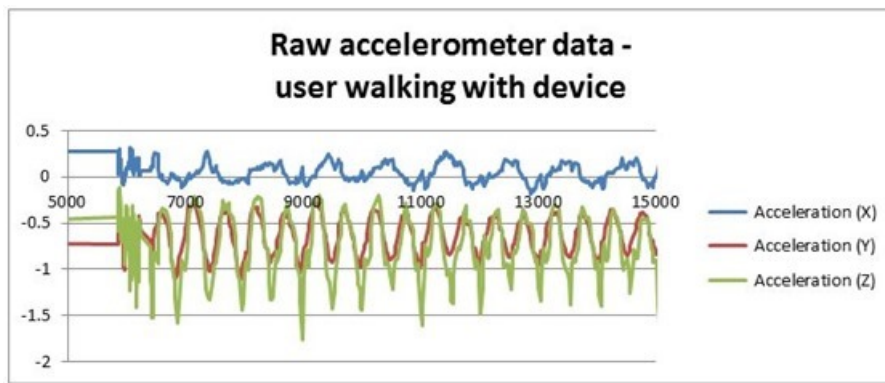
Essentially, sensor fusion is a case where the whole is greater than the sum of the parts. A typical sensor fusion system uses a 3D accelerometer, a 3D magnetometer, and a 3D gyro to create a combined “9-axis sensor fusion” system. To understand how this system works, let’s take a look at the inputs and outputs.



9-axis sensor fusion system

This diagram shows two types of outputs: pass-through outputs in which the sensor data is passed directly to an application, and sensor fusion outputs in which the sensor data is synthesized into more powerful data types.

Some applications can use pass-through sensor data directly. This data can be used at “face value” for a variety of scenarios. One such scenario is an app that implements a pedometer to count your steps as you walk. The graph below shows the output of the accelerometer for a person walking with a tablet PC. This graph clearly shows it is possible to detect every step the person took.



But, as our experiments revealed, many applications can't effectively use the raw sensor data. Some of these applications include:

- Compass apps
- Enhanced navigation and augmented reality apps
- Casual games
- 3D gaming apps

Here's a screenshot from a 3D game sample:



3D first-person shooter game (shown at //Build/)

These applications need to use sensor fusion data in order to support the features they implement. The “magic” of sensor fusion is to mathematically combine the data from all three sensors to produce more sophisticated outputs, including a tilt-compensated compass, an inclinometer (exposing yaw, pitch, and roll), and more advanced representations of device orientation. With this kind of data, more sophisticated apps can produce fast, fluid, and responsive reactions to natural motions.

By integrating a sensor fusion solution, Windows 8 provides a complete solution for the full range of applications. Sensor fusion in Windows solves the problems of jittery movement and jerky transitions, reduces data integrity issues, and provides data that allows a seamless representation of full device motion in 3D space (without any awkward transitions).

Working with hardware partners

While designing a sensor fusion solution for Windows, we also needed to help hardware designers to take advantage of this solution by partnering with them early. Designing a sensor fusion system is relatively easy if you're designing a single device. But Windows runs on many kinds of PCs in many form factors, using hardware components from many different manufacturers. We needed to provide a solution that enabled the entire ecosystem of Windows hardware partners to participate.

The first step was to provide a baseline of performance for sensor packages that would work with Windows' sensor fusion solution. Using Windows certification guidelines, we provided specifications for sensor performance. To help hardware companies verify that their solutions were compatible with Windows, we built a number of tests, which we provide with the Windows Certification kit.

Reducing the cost of developing and supporting drivers was another challenge. In order to make it simpler for sensor hardware manufacturers and PC makers, we wrote a single Microsoft-supplied driver that would work with all Windows-compatible sensor packages connected over USB and even lower power busses like I2C. This sensor class driver enables hardware companies to innovate with sensor hardware while ensuring that their hardware can be supported easily with drivers that ship with the Windows operating system.

To help speed adoption of the class driver, Microsoft worked with industry partners to introduce the specification into public standards. In July 2011 the standard for sensors was introduced in the HID (Human Interface Device) specification of the USB-IF (HID spec version 1.12, introduced with [review request #39](#)). This standardization enables any sensor company to build a sensor package that is compatible with Windows 8 by following the public standard USB-IF specifications for compliant device firmware. This reduces the time and cost required to integrate sensor hardware with Windows 8 PCs. Other benefits include a lower support cost and more consistent hardware capabilities for Windows 8 PCs that are equipped with sensors.

But beyond standardizing the class driver, we also wanted to optimize the performance of the sensor fusion solution, and minimize its impact on battery life. Each active sensor on a system draws power, and sending data up the stack consumes both memory and CPU time. We helped minimize the power and performance impact for sensor fusion systems running on Windows 8 in two major ways:

1. We architected the sensor fusion interfaces in Windows 8 to enable much of the processing of sensor fusion data to happen at the hardware level. This hardware-level sensor fusion capability means that computationally expensive algorithms don't have to run on the main CPU, saving power and CPU cycles.
2. We implemented powerful filtering mechanisms that we tied directly to the needs of sensor apps running at any given point in time. This pay-for-play data and event model means that sensor data is only sent up the stack at the rate that apps need that data, and no faster. This results in greatly reduced CPU utilization for sensor data throughput.

Sensors and Metro style apps

To pull all of this together, our final challenge was to make the power and promise of sensor fusion available to those writing Metro style apps. To enable this, we designed a sensor API as part of the new WinRT. Through these APIs, developers can access the power of sensor fusion from any Metro style app. These APIs are clean and simple, and at the same time give developers access to the data needed to support everything from casual games to virtual reality applications. Of course these capabilities are all available as Win32 APIs for game developers or other uses in desktop applications.

The following JavaScript code snippet shows how easy it is to get access to an accelerometer and subscribe to events using the Windows Runtime:

```
var accelerometer;
accelerometer = Windows.Devices.Sensors.Accelerometer.getDefault();
accelerometer.addEventListener("readingchanged", onAccReadingChanged);

function onAccReadingChanged(e) {
    var accelX = e.reading.accelerationX;
    var accelY = e.reading.accelerationY;
    var accelZ = e.reading.accelerationZ;
}
```

For more information about support for sensors in the Windows Runtime, please see this [//build/](#) session on [using location & sensors in your app](#).

You may be wondering at this point how you can try out sensor fusion on Windows 8, or even write some apps that use these new capabilities. Developers who attended the [//build/](#) conference in 2011 received the Samsung Windows 8 Developer Preview slate PC, which included a full package of sensors. There were only about 4,000 of those given out, so of course, not everyone had the opportunity to get one. The good news is that the same 9-axis sensor fusion system that was built into the Windows Developer Preview device is now available online for purchase from ST Microelectronics. The "ST Microelectronics eMotion Development Board for Windows 8" (model # STEVAL-MKI119V1) attaches via USB, and works with the HID sensor class driver that's included in Windows 8. If you've downloaded the Developer Preview version of Windows 8 and are itching to try out the sensor experience you should consider getting one of these devices.



ST Microelectronics eMotion Development Board for Windows 8

Now let's take a look at sensor fusion in action!

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

-- Gavin

Acting on file management feedback

Steven Sinofsky | [2012-01-30T12:00:00+00:00](#)

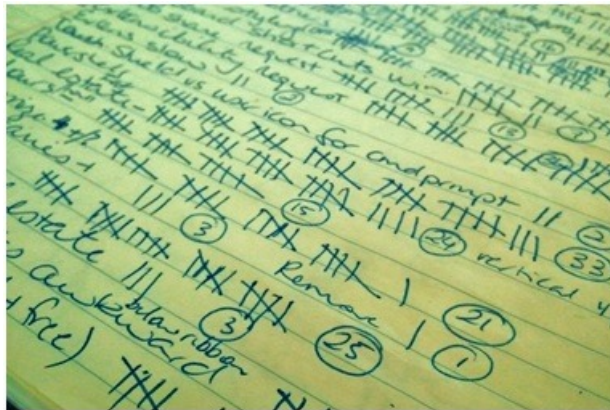
As we approach our next public milestone, we will begin to circle back to topics we covered in the blog and talk about the changes we've made to the product since the Developer Preview. As we've said often, we read the comments, newsgroup discussions, and reviews that have been written about Windows 8 and track the feedback carefully. We listen to this feedback by taking into account the source of the feedback and factoring in the intended audience for features as well as trying to reconcile conflicting feedback (no matter how many thumbs up votes there might be, we can promise that, for any design worth discussing, there are conflicting and equally valid points of view). Of course, we always consider the engineering feasibility of any changes we make—compatibility, security, performance, and so on.

Ilana Smith, a lead program manager on the Engineering System team, authored this post.

--Steven

We previously published three blog posts that discussed the new file management experience in Windows 8: one about the [new copy experience](#), one that detailed the design process we went through for [the new conflict experience](#) and one about the [changes to Windows Explorer](#), including the introduction of the ribbon.

Those posts prompted great discussion and we read the approximately 2200 comments you left. This was wonderful feedback for us, and, along with information from our other feedback channels, we incorporated it into our design process.



Summarizing blog post comments

As we prepare for the beta, we thought we would update you on some of the key issues, and the changes you should expect to see.

Conflict: identifying duplicate files during conflict resolution

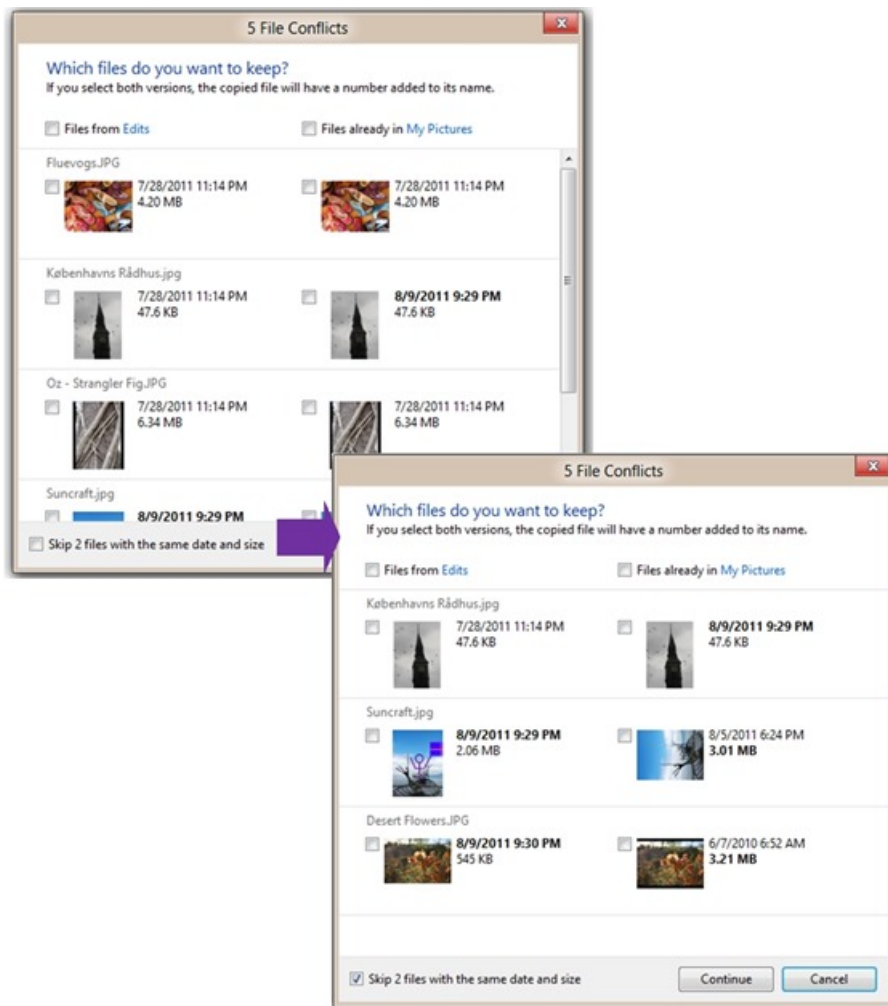
In Windows 8, we have a new experience for selecting the right file when file name collisions are encountered during a copy or move.

L. Brown said:

A compare button to show if files are equal in the "Choose" dialog would be really great!

Frequently, the reason two files have the same name is because they're copies. Making a choice between two identical files is usually pointless – it's unnecessary for a copy operation, and often unnecessary for a move operation. We looked at several methods of identifying duplicate files and decided that checking the file name, file size and date modified attributes was the most effective approach. They can be used to identify the vast majority of duplicate files quickly, efficiently, and with good backward compatibility compared to other methods like file hashes.

In the beta, we've added a new option to the detailed conflict resolution dialog. By checking the box in the bottom left of the dialog, you can filter out all files that match on name, size (down to the byte) and time (down to the [granularity of the file system timestamp](#): 2 seconds for FAT, 100 nanoseconds for NTFS). The system will skip copying or moving these files. This functionality adds no additional time to the operation, works both locally and across networks, and on all types of systems and storage.



We'll skip copying files with the same name, date, and size

This check box is deselected by default (to ensure users opt into the changed behavior), but it persists once you select it.

Copy: system changes

JL asked:

You know when you start a big copy job and realize that you are doing it over the wireless so you grab a network cable and plug it in? Does the file copy know to utilize the faster connection now?

If both sides of the copy operation are on Windows 8 machines, yes, it will be able to take advantage of the increased network throughput on the fly, thanks to advancements in the [Server Message Block \(SMB\) protocol](#) to support multiple channels.

Tobi asked:

Will it be possible to pause the copy operation and resume it after reboot/sleep/hibernate?

In the beta, when a system sleeps or hibernates, the copy operation will automatically pause, and when the machine wakes, you can choose to resume the copy by clicking the depressed pause button. (We decided not to have copies automatically resume on wake, as the system environment may have changed significantly in the interim and we do not want to cause an error.)

Copy: handling confirmations and interrupts

gawicks asked:

Please please display all the copy 'error dialogs' after the copying has completed so I don't have to sit in front of the machine all the time.

We have two types of user interaction that can occur during a copy job - we break these into two groups, "confirmations" and "interrupts." Confirmations like "Are you sure you want to permanently delete this file?" need to be completed before the copy operation can start. Interrupts are issues that the system encounters while copying, things like "File not found," "File in use," and file name conflicts.

The system presents all confirmations before it starts to move or copy files. While copying, any interrupt issues are queued and presented once the

system has completed all the work it can. In the beta, we've made improvements in how confirmations are presented, making sure they don't get lost amongst existing running copies.

Explorer: navigation pane scrolling issue

xpclient said

Please fix the infamous Windows 7 navigation pane scrolling bug.

(See [this Microsoft Answers thread](#) for more information.)

We fixed it! As of the beta, it's gone.

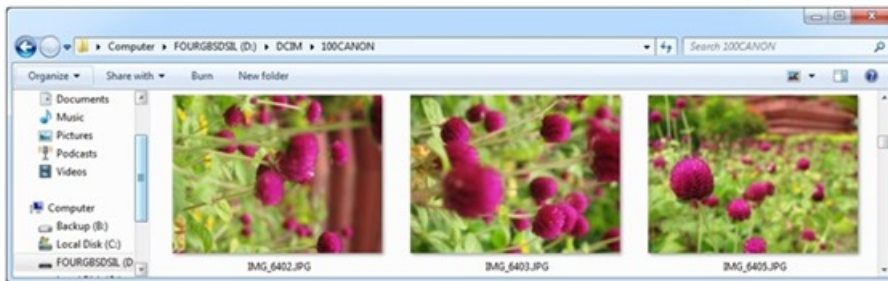
Explorer: respect picture orientation metadata

Raf asked

*Will you support *lossless* picture rotation?*

In Windows 7 and 8, JPEG rotation is lossless when both image dimensions are divisible by 16 (standard image sizes).

Additionally, Explorer now respects EXIF orientation information for JPEG images. If your camera sets this value accurately, you will rarely need to correct orientation. Look for a future blog post where we will discuss this in more detail.



Images in Windows 7 Explorer



Images in Windows 8 Explorer

Explorer: Overlay changes to improve performance

In Windows 8, we continue to prioritize great performance. We pay close attention to milliseconds of lag and look for reductions. In Explorer, we found an opportunity for improvement in delays caused by icon overlays.

In Windows 7, we have a padlock icon overlay to indicate a private file. (You might recall that, due to the increase in shared files, it had superseded the "palm up" overlay for shared files.) We recently found that checking for these overlays was adding about 120 milliseconds to our Explorer library launch tests. This might not seem like much, but we consider this a big delay.

Name	Date modified	Type	Size	Sharing status
Bar	1/25/2012 8:33 AM	File folder		Not shared
Foo	1/25/2012 8:33 AM	File folder		Private

Overlays have limitations – they can only show a single state, add a lot of visual noise, and can be confusing. The padlock overlay has been removed; this information is conveyed better by the "Sharing status" column.

This column has these advantages:

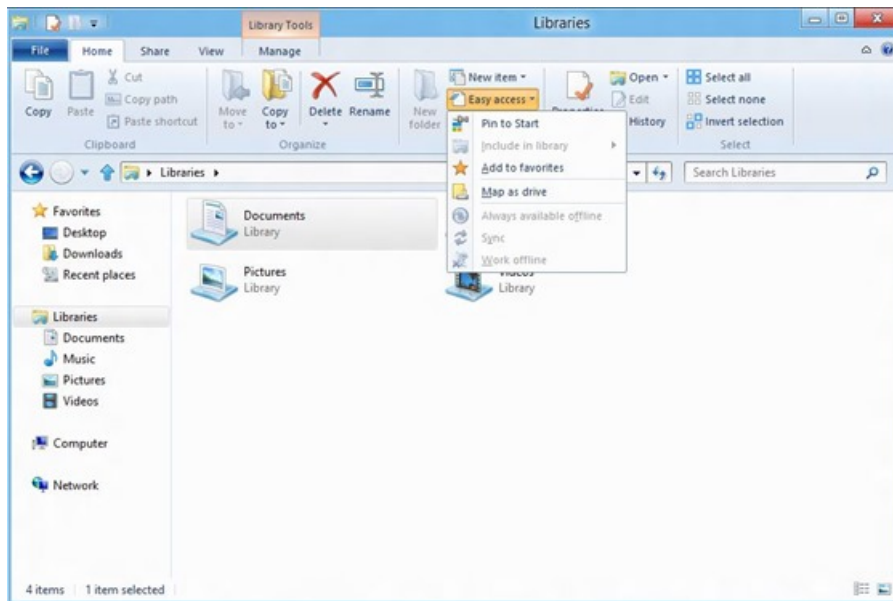
- **Performance:** The column is hidden by default, so the delay is incurred only when you opt into showing this information.
- **Tri-state:** This column has three values: Shared, Not shared, and Private, so you get more detail than you would from the icon overlay.
- **Sorting/filtering:** You can sort and filter the sharing status property, providing more powerful file management capabilities.

Explorer: pin to Start

On Marina's [post about the Start screen](#), Boots112233 said:

Half of the items in my Windows 7 Start Menu are shortcuts to folders and one is to a file [...] How can I do this in Windows 8 if the start screen won't allow shortcuts for folders?

In the beta, you can now easily pin your favorite folders to Start, and take advantage of the rich customization functionality that we built into it to arrange the folders into groups and into any order you want.



"Pin to Start" from the Windows Explorer ribbon

Additionally, just as in Windows 7, you can pin shortcuts to executables to Start directly from Windows Explorer, which can be very useful for applications that don't add themselves to the Start screen by default.

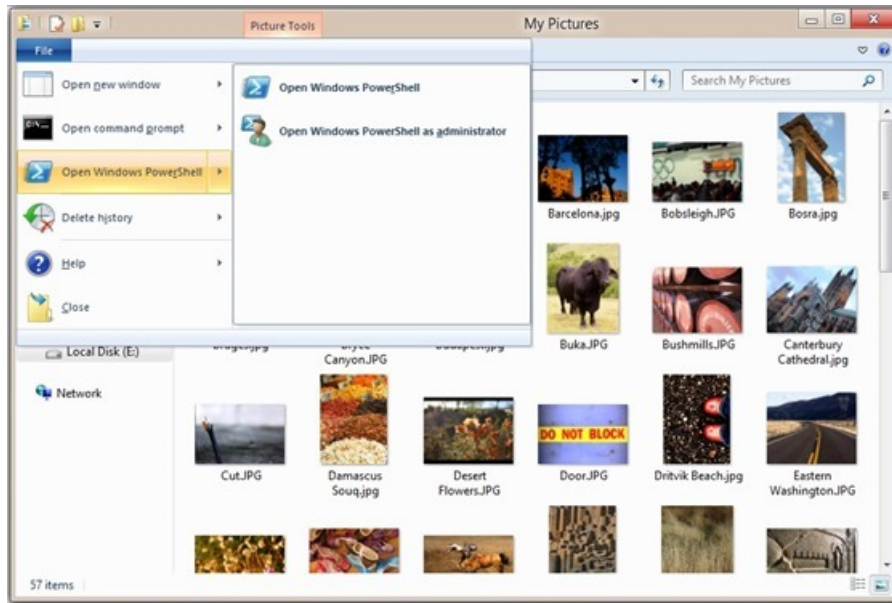


Explorer: PowerShell

Jamie Thomson said:

Really liking the "Open Command prompt" option in the File menu however I prefer to use PowerShell so would like an "Open PowerShell prompt" option too.

We agree, and so we added this as well. It is worth noting that there are sometimes conflicting points of view on whether advanced things should be in the GUI or in PowerShell, and how front and center they should be. We are always balancing the complexity of too many options and too many ways to do things. As you can see, there is no right answer, so we'll continue to balance these complex choices.



Windows PowerShell buttons in Windows Explorer

These menu items launch the PowerShell console. The PowerShell ISE continues to be available from the Edit command on a PowerShell file.

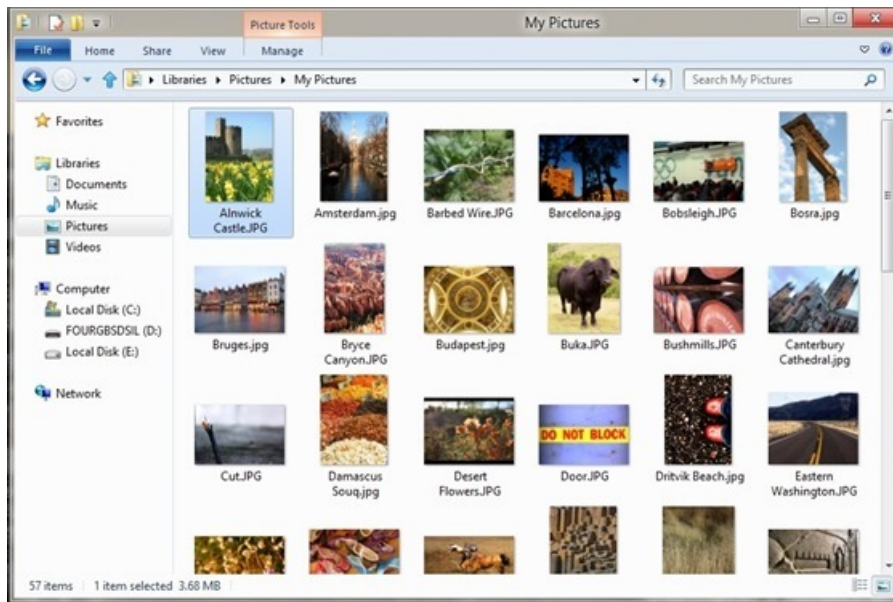
Explorer: ribbon changes

We had expected the introduction of the ribbon to Explorer to spur conversation, and it is fair to say the voluminous response was in line with our expectations. It's exciting to work on something that brings so many different perspectives.

There were many reactions, and as we expected, there is a set of people who have an entirely negative reaction to the affordance and have been telling us about it in no uncertain terms. Our view is that we do need to move the user interface forward and accept that a vocal set of customers are just not happy with the direction we're going. When looked at broadly, that is balanced out by a majority of people who are happy and more productive with the changes. We remind folks that there are third-party tools available (likely the tools being used by this set of people), that provide a number of different interface paradigms. We do embrace the notion that third-party tools play an important part in the Windows experience.

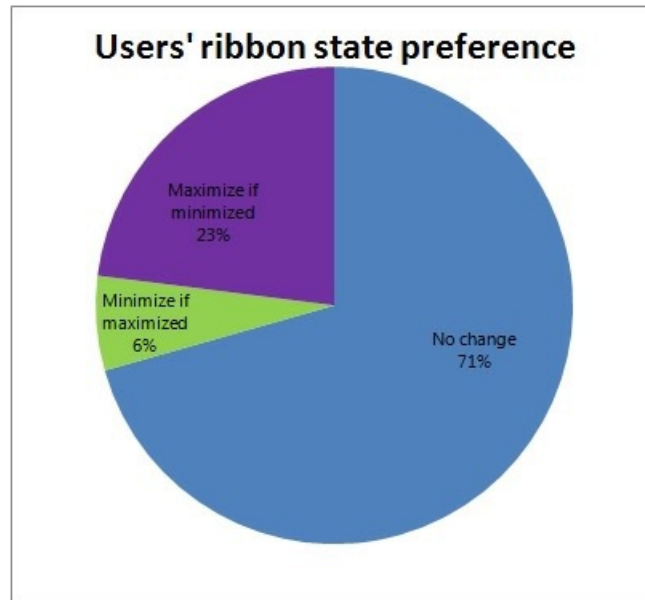
That said, we've internalized your feedback, experimented with and tested various approaches, and used our co-workers as test subjects, in addition to the formal testing as you would expect. You'll see three major changes in the ribbon in the beta.

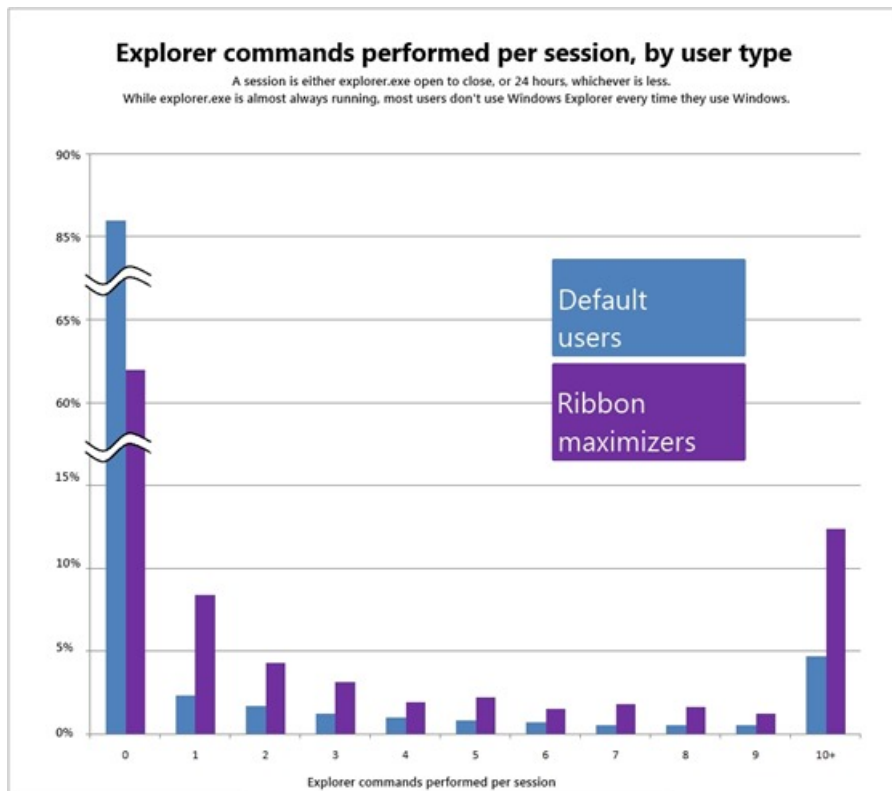
Ribbon minimized by default: With the ribbon maximized in the Developer Preview, we've been able to learn a lot about how people interact with it, which has enabled us to tweak and fine-tune it. With the beta, we will be making a major change that brings Explorer in line with our design principles for Windows 8. As in our copy dialogs, Task Manager, and Metro style experiences, we will be reducing distractions and trusting users to discover functionality on their own, by minimizing the ribbon by default.



Windows Explorer ribbon minimized by default

We've tested this change for a while now, and the results have been heartening. This is data from internal usage at Microsoft, which we know not to be representative of broad audiences, but is generally representative of the folks like you that engage in the dialog on the blog.

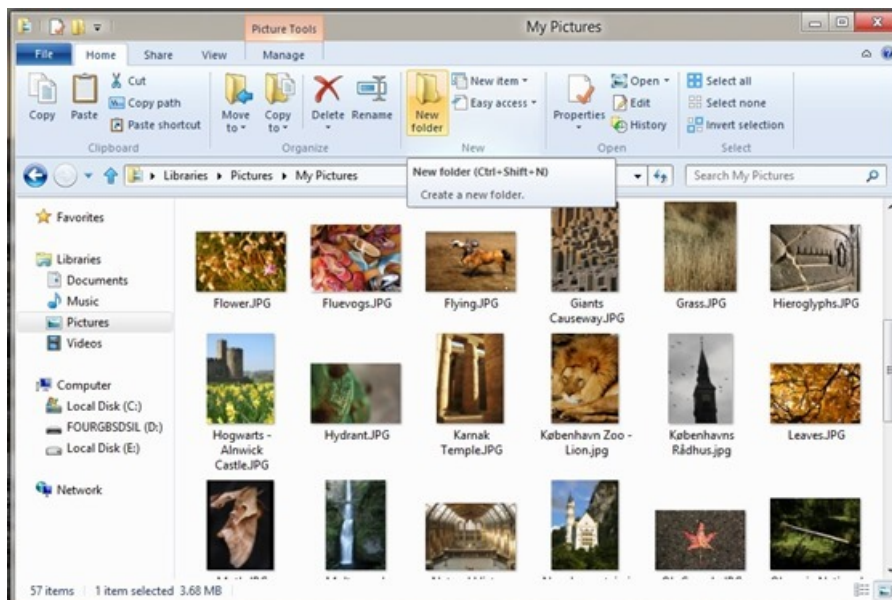




This data shows that our very tech-savvy users are generally fine with either setting, but that our heavier Explorer users are our ribbon maximizers. For lighter file browsing scenarios, we can provide a UI with reduced distractions, and still trust that users who want to really exercise Explorer functionality will maximize and leverage the ribbon.

Visible hotkeys: Our telemetry data has shown us that for users who actively choose to minimize the ribbon, their strong preference is to use hotkeys. The ribbon provides new ways to access functionality via the keyboard with keytips (those floating cues that pop up when you hit Alt), but traditional shortcut keys like Ctrl+V remain the most efficient method. We love shortcut keys (internally, their usage gets up over 85% of all Explorer commands issued), so we want to help more people discover them.

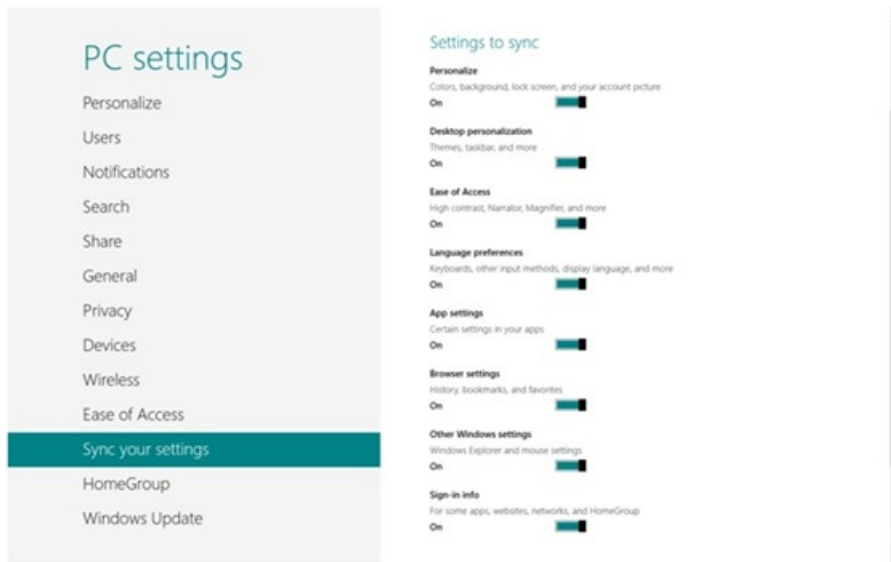
For the beta release, we've added hotkey information to the tooltips of relevant buttons.



“New folder” tooltip shows the keyboard shortcut

User setting roaming: We want to make sure you only need configure your Explorer options once. If you maximize your ribbon, and add Undo and Map Network Drive in your Quick Access Toolbar, we want your Explorer to look like that every time.

For the beta release, we've added Explorer settings to the attributes that are roamed to your other Windows 8 PCs. In the “Sync your settings” UI, this shows up under “Other Windows settings.” (For more information about roaming user settings, take a look at [Katie's post.](#))



Syncing Explorer settings across PCs

We really appreciate all your feedback on our previous posts. We believe it has contributed directly to an improved file management experience for Windows 8.

--Ilana Smith

Improving power efficiency for applications

Steven Sinofsky | [2012-02-07T10:00:00+00:00](#)

Minimizing the power consumption of your PC while maximizing the responsiveness and utility (making it “fast and fluid”), is a significant engineering challenge. While it starts with the work we do in Windows to provide support for the right level resource usage, this work requires developers to take resource utilization into account as they develop their apps. Power efficiency applies to all form factors and all usage scenarios—using less power is the right thing to do for everyone. This is an area of significant innovation for Windows 8 PCs, and builds on the foundation of the new runtime model in WinRT—it is not the sort of thing you can retrofit onto existing desktop applications while still maintaining functionality and compatibility. Much like state migration and setup that we talked about earlier, power consumption is an area of Windows that has been reimagined for new scenarios. With your existing x86-based PC, all of the existing support is still there, and all of the work you do on the desktop continues exactly as it has before (and of course, has been improved, as we have seen). As we see new hardware across all supported SoC hardware (including Intel) this level of power efficiency will be more broadly available. Though we will discuss some of the work we’ve done to improve the power consumption of desktop apps, to enable the all-day, always-connected scenarios we’re going to see new apps written to WinRT that run on a new generation of hardware that supports new power management capabilities.

Sharif Farag and Ben Srour, lead program managers on the Fundamentals and User Experience teams respectively, authored this post.

--Steven

We’ve featured several posts about how we’re working to improve battery life for Windows 8 PCs. In Pat Stemen’s [Building a power-smart general-purpose Windows](#) post, we discussed some of the investments that we are making in Windows 8 to enable a new smartphone-like power mode on system-on-a-chip (SOC) hardware, a mode we call Connected Standby. In [Updating live tiles without draining your battery](#), we talked about how we’re enabling live tiles to give you fresh and current information without creating a lot of underlying activity that erodes battery life. In this post, we’ll expand on a few additional innovations that we didn’t cover previously, about how we’ve minimized the power usage of running apps on Windows 8 PCs while still getting the most out of them.

As Pat mentioned in his post, applications can influence power consumption by consuming resources—CPU, disk, memory, and other resources—as each of those resources has a power cost associated. So the trick is to let applications utilize the resources they need while you’re actively using them, but reduce resource utilization to the bare minimum when you’re doing something else. This is true of course for the OS itself as well. Pat covered some of the work we did to improve this, but in fact, there were hundreds of small improvements made on this front—what we call “power hygiene” improvements to limit OS resource usage and activity. We were careful not to take this too far though, and undermine functionality that customers expect – like completing activities that they’ve started and then switched away from.

For example, in response to the live tiles post, @ItsMe asked:

“What about background copy jobs. If I put explorer application to background to look [at or] write a word document, do you mean the copy-job is paused until I “fullscreen” the copy-job again? Seriously?”

The answer to that question is no, file copy will definitely continue to work exactly as it does today and will not get suspended if you start a copy job and then go do something else while it completes in the background. This works the same whether you are in front of the PC using it or leave the PC long enough for the screen saver or lock screen to kick in. Suspension of inactive apps only applies to Metro style apps, not to basic OS functions like copying files.

Focus on the foreground

For Windows 8, we started off with a rule that would apply to the large majority of Metro style apps: if an app is not on screen, and the screen is not on, it should not impact your battery life. That doesn’t mean WinRT and the user model preclude multi-tasking. There’s a new way of thinking about how and when code takes into account modern hardware capabilities, networking demands, form factors, and reliability/security/privacy. There are going to be some exceptions (e.g. background email syncing, desktop tools), but for the majority of cases, we expect the app to do most of its work while you are actively interacting with it. When an app is not in the foreground, we wanted to ensure that it would either suspend completely, or use limited resources based on a set of common background capabilities (like copying files), which the app can access.

So basically, this means that an app can be in one of three possible states:

1. Actively running in the foreground
2. Suspended in the background
3. Performing some defined background activity

Actively running in the foreground

The foreground app actively running is pretty easy, as we just let it run and utilize CPU, disk, memory, and other resources as needed. In this way,

Metro style apps are essentially the same as Windows applications have always been. This includes both the case where a single app is full screen, as well as when two apps are on screen with one of them “snapped” to the side. This applies only when the screen is on, since it means the user is interacting with the PC.

There are a lot of new factors to consider in developing fast and responsive apps. Much like the transition from character to GUI programming, where concepts were substantially different, building apps that are respectful of power and resource consumption requires some new approaches. As an example, early Windows programmers were sometimes frustrated by the notion of a WNDPROC and felt that the best way to handle typing was by trapping the interrupt and translating the key press—the message-based approach inverted this, and Windows handled the translation and let your app know when to worry about the key press, which was quite different from the earlier way of doing things. In a world where 75% or more of the PCs sold are battery powered, programmers are, by definition, being asked to rethink how to get work done again.

Given this, it is important to think about app development in a forward-looking manner that keeps pace with and plans for the evolution of hardware and customer needs. The existing application model needed to evolve in order to yield the power savings and battery life that customers want. Of course, as we keep saying, desktop applications that you currently have will work exactly like they do on Windows 7 today (and were even improved in many dimensions). But over time, we’re equipping Windows to get more done and use less power, with new applications that help you get that work done—from entertainment to professional tools and everything in between. The resources available to compute, the resources required, and the types of computation done are changing, and Windows 8 is providing new facilities to tap into this opportunity.

In a foreground-based approach, concurrency becomes a big part of how to develop fast, fluid, and responsive apps. [At the //build/ conference](#), we showed how to use the new tools and APIs to develop highly concurrent applications. This enables developers to think differently about how to code scenarios. So for example, rather than keeping a separate background app always running to do something even when it isn’t needed (which wastes battery life), programmers can take advantage of the new OS [background tasks](#) infrastructure to complete the necessary activity in the background in a power-efficient manner. Background Tasks can be invoked in a number of ways, such as from a push notification, from a timed event, or even from incoming network data. The system is also smart enough to allow apps to run more often in the background when your PC is plugged into the wall. Overall, this is a big win for battery life because code will run only when it is needed instead of all the time. For example, your newsreader app can automatically download content for you in the middle of the night while your PC is plugged in, which allows you to have the freshest content available to you when you launch the app. And this in no way limits the ability to get the work done—it is a new way that gets the work done, and does so in a way that minimizes impact on critical system resources.

On top of improvements to how app code can execute in the background, we’ve made many improvements in the tools infrastructure and WinRT API to make asynchronous programming easier and more powerful. Fast and responsive apps are built on a solid foundation of asynchronous programming. In a main session at the //build/ conference, [Anders Hejlsberg showed off a WinRT approach](#) to building an asynchronous user experience centered on viewing a huge catalog of items. Leveraging techniques like these will help deliver great scenarios and foreground performance for apps, and extend battery life.

Suspending apps in the background

In the second state, after you launch an app and then switch away from it, the operating system suspends it. This means that the Windows scheduler (the component that schedules CPU access for processes and threads) does not include it in the CPU scheduling. Since the operating system is not scheduling the app, the app is not using the CPU, and it is possible for the CPU to drop into lower power states. Getting the CPU into low power states can be critical to achieving better battery life. Developers may be familiar with this approach, as it is similar to what happens to a process when you are debugging an app and you “pause” it. Essentially, all of the threads running for the app are halted. Suspended apps are in a similar kind of cached state. Since the app is already initialized, you get the benefit of instantaneous app switching. It’s simply a matter of the operating system scheduler allowing the app execution to progress again when it is switched back into the foreground.

The great thing about this new suspended state for your apps is that they are instantly ready for you to get back to them. When you switch to a suspended app, it resumes instantaneously and takes you back to exactly where you left off. With this, you will be able to switch between more apps faster than you ever have before on Windows. You’ll no longer need to care how many apps are running on your PC—with the help of live tiles keeping you informed about what is going in your apps, and the ability for apps to save and restore state, great apps always look like they are running.

For example, let’s say that you have an app that keeps track of what flights you have coming up. This app can show you the status of your next flight through a live tile notification, even though the app may be suspended or not even running in the background. When you switch to this app, it can open to the last place you were in the app (such as flight search), as if you never left. Since the notion of what is running is abstracted, we have made launching an app and switching to an app essentially synonymous. Whether you switch to an app using the back stack, or Alt+Tab, or the Start screen, you can get back to a suspended app instantly.

In this way, the list of “running” programs is for all practical purposes the same as the list of programs you see on the Start screen. (For keyboard users, do keep in mind that Alt+Tab works across all running programs the same as it does in Windows 7, and the Taskbar also works exactly the same (and is even improved for multi-monitor scenarios) for desktop apps.

The benefit of being able to suspend apps is that you get really fast switching between them without negatively impacting the battery life or performance of your system. This is altogether different than traditional desktop apps, where we are all used to optimizing our workflow for those apps that take a long time to launch.

There are two cases, in general, where we won’t suspend an app if it is not doing background activity. First, if you have not yet launched the app in your current logged-in session, then you’ll have to tap the app’s tile to launch it. The second case is more interesting. The system may remove an

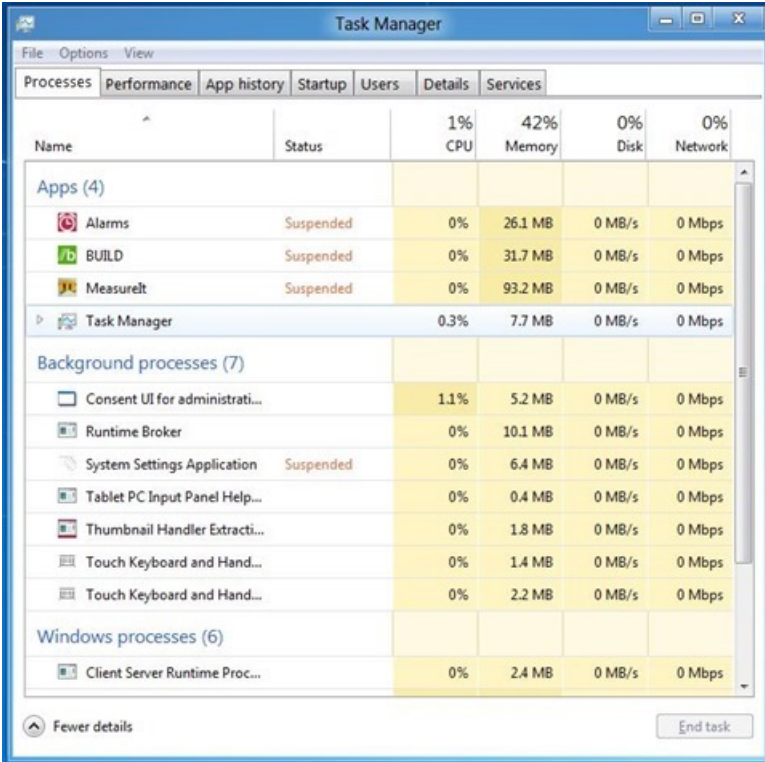
app from the suspended state and terminate the app if the system starts to run low on memory. Memory is a finite resource and we want the apps you are using most frequently to be ready for you instantly. If you have not used an app in a while and the operating system needs more memory, it terminates one of your suspended apps. This should happen relatively infrequently because the memory manager will take your suspended apps and save them to disk (which generally has more capacity than physical memory). When you switch back to these apps, they will be ready instantly. However, there will be cases where the system does have to terminate a suspended app. This typically occurs when there are multiple users logged into one PC, or when you are using a bunch of memory-intensive apps.

The operating system takes several things into account when it decides which apps to terminate, such as when the app was last used, and how much memory it is taking up. Ideally the operating system is terminating as few suspended apps as possible, so you can switch back to suspended apps as often as possible. Even though an app may be terminated from the suspended state, there is really very little impact to your experience, because the app model has evolved to easily allow developers to save state incrementally while the app is being used, and restore it when the apps is re-launched. For example, the flight tracking app could remember that you were on the flight search page when you decided to move on to another app, and then use this information to bring you back there when you switch back to the app, even if it happens to have been terminated.

The important thing to underscore is that even though there may be several suspended apps in the background taking up memory, there is no negative performance or battery life impact to your PC. In fact, you do not need to manage or close apps directly at all. This is a common approach being used across computing devices now and represents a modern view of operating system design (see for example, this [blog post about iOS multitasking by Frasier Spears](#)).

Even though you do not need to close apps for improved performance or battery life, you will be able to close apps in the upcoming beta of Windows 8. Sometimes an app may get into a bad state or you are just done and don't want to see it again. You'll be able to use the mouse, a touch action, or a keyboard shortcut, to close an app. We'll have a follow up post that goes into some of the changes we are making here when the beta comes out, so stay tuned!

In the Developer Preview build, you can see the suspended state in action by opening up the [new Task Manager](#) after launching a couple of the apps that came with that build, such as the Stocks, News, or Weather app. Notice that the CPU utilization is listed as 0% because, even though those apps are in memory, they are essentially asleep, and cannot negatively impact battery life or performance.



The screenshot shows the Windows Task Manager window with the 'Processes' tab selected. The window is divided into three sections: 'Apps (4)', 'Background processes (7)', and 'Windows processes (6)'. The 'Apps' section shows four Metro style applications: Alarms, BUILD, MeasureIt, and Task Manager. The 'Background processes' section shows seven system processes, including Consent UI for administrative..., Runtime Broker, System Settings Application, Tablet PC Input Panel Help..., Thumbnail Handler Extracti..., Touch Keyboard and Hand..., and another Touch Keyboard and Hand... process. The 'Windows processes' section shows one process: Client Server Runtime Proc... The table below summarizes the data shown in the screenshot.

Name	Status	1% CPU	42% Memory	0% Disk	0% Network
Apps (4)					
Alarms	Suspended	0%	26.1 MB	0 MB/s	0 Mbps
BUILD	Suspended	0%	31.7 MB	0 MB/s	0 Mbps
MeasureIt	Suspended	0%	93.2 MB	0 MB/s	0 Mbps
Task Manager		0.3%	7.7 MB	0 MB/s	0 Mbps
Background processes (7)					
Consent UI for administrati...		1.1%	5.2 MB	0 MB/s	0 Mbps
Runtime Broker		0%	10.1 MB	0 MB/s	0 Mbps
System Settings Application	Suspended	0%	6.4 MB	0 MB/s	0 Mbps
Tablet PC Input Panel Help...		0%	0.4 MB	0 MB/s	0 Mbps
Thumbnail Handler Extracti...		0%	1.8 MB	0 MB/s	0 Mbps
Touch Keyboard and Hand...		0%	1.4 MB	0 MB/s	0 Mbps
Touch Keyboard and Hand...		0%	2.2 MB	0 MB/s	0 Mbps
Windows processes (6)					
Client Server Runtime Proc...		0%	2.4 MB	0 MB/s	0 Mbps

Metro style apps get suspended in the background

Performing background activities

As we've already discussed, developers need to think of how work that was previously done in the background can still be accomplished without impacting battery life. It is easy to ask for multitasking and just enable it, but the downside of this is that if all your apps are always running background tasks then you will never achieve long (or even improved) battery life. In a mobile and constantly connected world of laptops, this is incredibly important. So with Windows 8 and WinRT, we created new APIs to cover [background processing for Metro style apps](#).

Again, your desktop apps will continue to run just as before, but they will also impact battery life just as they currently do (albeit with some improvements we will talk about below).

We set out to achieve a balance between enabling the kinds of rich app capability and multitasking that people expect in Windows, while also being conservative about resource utilization. To do this, we listed out a set of key scenarios we wanted to enable, and set out to achieve each of

them in the most resource-efficient way possible. The result is a set of background multitasking APIs, which allow apps to complete an action in the background in a way that is resource- and power-efficient, and allow app developers to focus on what they want the app to do without having to do a bunch of extra work.

We took a scenario-focused approach to enable the most common tasks that apps would need to do in the background. Here is what we enable in the background for Metro style apps in WinRT:

- Playing music
- Downloading a file from or uploading it to a website
- Keeping live tiles alive with fresh content
- Printing
- Receiving a VoIP call
- Receiving an instant message
- Receiving an email
- Sharing content (like uploading photos to Facebook)
- Synchronizing content with a tethered device (like syncing photos)

This set of scenarios is based on common patterns used by developers and common patterns we expect to see. Some of these scenarios end up using the same platform affordance, so let's walk through each of them, so you can understand the landscape and power of Windows 8:

Scenario	Description
Background download or upload	Accessing and storing content on the Internet is a pretty common scenario for apps. We want you to always have the freshest content already loaded as soon as you switch back to your app. This will be particularly helpful with magazine or news-based apps. Apps can use the new background transfer API to perform uploads and downloads in the background. This API is what we call "fully brokered," which means that the OS itself performs the upload/download. This takes app code out of the picture, and helps maximize battery life.
Background audio	We still want you to be able to do more than one thing at a time, especially if one of those things is just listening to music. Any media or communications app can play audio in the background . To maximize efficiency, we suspend the app when you pause the audio.
Sharing	If your app is in the middle of sending content to a cloud service using the Share charm , it can complete that operation in the background.
Lock screen apps	Lock screen apps typically need to notify you with the latest information, and this could happen at any time, even when you are not using the app. The most common examples are your email, VoIP, and IM apps. Lock screen apps can deliver notifications and sync your data, even in the background while on battery, and even when the screen is locked.
Printing	You can print documents even though the app doing the printing has been moved to the background.
Device sync	You can synchronize content between a connected device (like a camera) and your PC even though the app is not visible on screen.
Live tiles with Windows Notification Service	Apps can give the impression they are running all of the time (even if they are suspended) by sending push notifications to your Windows 8 PC to provide the freshest content for the app's live tile.

Scheduled notifications

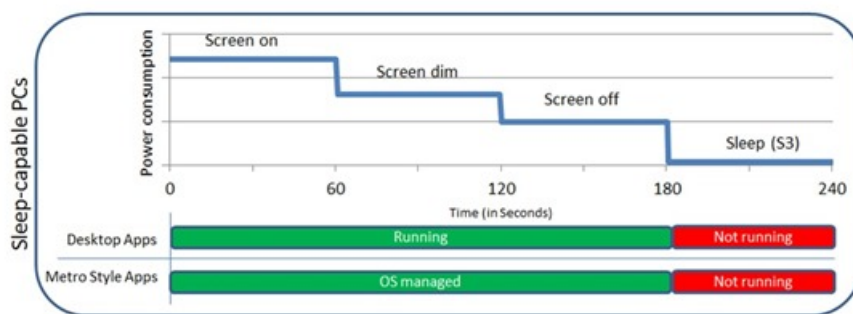
Apps can notify you of an event at a particular time by either updating a tile at particular time (think: calendar appointments) or by popping a notification up on the desktop reminding you to do something before you leave the office. These events are scheduled by the app, but Windows is responsible for delivering the notification, which helps minimize battery impact.

Background tasks

[Apps can run code when certain events occur](#), such as on a periodic interval, or when you sign in to Windows or an IM service, for example. Lock screen apps can run code every 15 minutes, but non-lock screen apps can register to run code every 15 minutes as long as the device is plugged into A/C power.

Connected standby and sleep-capable machines

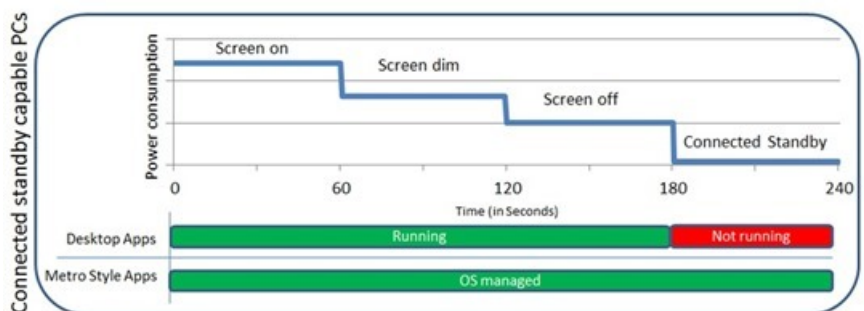
By the time Windows 8 is released, there will be a broader range of PCs available than ever before. Many of these will have similar power options to those running Windows 7 today. Besides turning off completely, they will be able to go into a “sleep” state, either on demand, or after a period of inactivity. During sleep, all system activity is completely suspended.



Application execution on PCs that are sleep-capable (default settings)

The chart above shows how, as the PC idles just prior to sleep, desktop apps continue to run in the same way as they have in prior versions of Windows, while Metro style apps run in the managed way I described earlier. When the PC goes to sleep, both desktop apps and Metro style apps are fully suspended. This is great for battery life—when the machine is asleep, it consumes very little power. It’s not as great for a data-freshness though, since when the machine is asleep, it isn’t getting live tile updates, downloading new mail, or getting ready to alert you with alarms or other notifications.

As Pat covered in his post, we’ve enabled a new smartphone-like power state for a new class of PCs that rarely get turned off completely. Typically based on “System on Chip” (SoC) architectures, these PCs are interesting because instead of turning off during periods of inactivity they go into a very low power state while still running. This new state is referred to as “connected standby.” This enables some great connected scenarios, such as always having email up-to-date, and being able to receive instant messages or phone calls, while still delivering amazing battery life. The chart below shows behavior for both desktop and Metro style apps during connected standby. For this to really work effectively though, we had to consider both Metro style apps (which, as you saw earlier, we can very effectively ensure are conservative with system resources), as well as desktop applications, which presented a tougher challenge because they have been designed over the years to expect either full access to system resources (when running in the fore or background) or no access (when the PC is asleep.)



Application execution on PCs that have connected standby

To this end, we have added a new component to Windows 8 called the “Desktop Activity Moderator,” which only runs on these new connected standby-capable platforms. This component is designed to help reduce the resource utilization of desktop apps when the device goes into connected standby. If we allowed apps to continue running unchecked in this low-power mode, the PC would run down the battery more quickly.

Instead, we suspend desktop applications, stopping their resource use and maximizing battery life. From the applications' perspective, it will appear as if the PC has simply been put to sleep. When the PC is woken from connected standby, the app will resume as if the PC had been woken from a sleep state.

However, there are actually several components on the system that are required for connected standby, which we cannot suspend. These include drivers, some inbox and 3rd party services, and of course, the Metro style apps that use the background features mentioned earlier. Many of these provide functionality such as responding to user input when you return to your device, or providing network functionality. We enable these to run in connected standby after careful evaluation to ensure they do not have a significant impact on battery life. In addition, there are a set of processes that need to run in response to activity on the system. These processes are throttled to only run for short periods of time until a background activity is initiated, at which point they are allowed to run unimpeded. A great example of this is an antivirus product, which is often scanning in response to activity on the system. When there is background activity occurring such as receiving an incoming email via the background affordances, antivirus can run unimpeded during this time. But during the majority of the time when incoming network activity is not occurring, there is very minimal activity and therefore these components will be throttled to minimize their impact on battery life.

Summary

As you can see, we have made significant investments in engineering Windows 8 to be great for battery life. We engineered the new application model to deliver consistently long battery life while enabling connected experiences. Applications that were designed for Windows 7 will continue to work as they have before with no change in behavior, and new Metro style apps can be developed to enable new connected experiences that work in a more power-efficient manner, by taking advantage of the background infrastructure that the operating system provides.

-- Sharif Farag and Ben Srour

Building Windows for the ARM processor architecture

Steven Sinofsky | [2012-02-09T10:00:00+00:00](#)

One of the notable aspects of Microsoft Windows has been the flexibility the architecture has shown through shifts in technology and expansion of customer usage over time. What started out as an operating system for one person working solo with productivity software is now the foundation of a wide array of hardware and software technologies, a spectrum of connected Windows products, and an incredibly flexible approach to computing. With *Windows 8*, we have reimagined Windows from the chipset to the experience—and bringing this reimagined Windows to the ARM® processor architecture is a significant part of this innovation. Expanding the view of the PC to cover a much wider range of form factors and designs than some think of today is an important part of these efforts. Windows on ARM enables creativity in PC design that, in combination with newly architected features of the Windows OS, will bring to customers new, no-compromise PCs.

This post is about the technical foundation of what we call, for the purposes of this post, *Windows on ARM*, or *WOA*. WOA is a new member of the Windows family, much like Windows Server, Windows Embedded, or Windows Phone. As with those products, WOA builds on the foundation of Windows, has a very high degree of commonality and very significant shared code with Windows 8, and will be developed for, sold, and supported as part of the largest computing ecosystem in the world. Today we'll focus on the development of WOA and introduce some of the features, along with how customers will experience it. As with x86/64 Windows 8, there are still announcements to be made relative to the business and marketing aspects of the product(s). Today's blog post is about making WOA, not marketing or selling it.

At the same time, while this post is exclusively on our work on WOA, we have had a deeper level of collaboration with Intel and AMD on the full breadth of PC offerings than in any past release. Windows 8 innovations on powerful and richly capable x86/64 processors, and work on new low-power processors such as those that Intel demonstrated at CES, require an equally strong commitment, even larger engineering investment, robust new designs, and improved architecture for Windows across these platforms. While discussing our engineering for ARM processors, it is important to keep in mind that in addition to all of the new work for the ARM platform we have done, much of the work discussed in this post applies directly to the x86/64 platform and Windows 8 as well. We could not be more excited or supportive of the new products from Intel and AMD that will be part of Windows 8—across a full spectrum of PC form factors including tablet, notebook, Ultrabook™, all-in-one, desktop, and more that all take advantage of the new capabilities of Windows 8 while Windows 8 takes advantage of new features in hardware.

Using WOA “out of the box” will feel just like using Windows 8 on x86/64. You will sign in the same way. You will start and launch apps the same way. You will use the new Windows Store the same way. You will have access to the intrinsic capabilities of Windows, from the new Start screen and Metro style apps and Internet Explorer, to peripherals, and if you wish, the Windows desktop with tools like Windows File Explorer and desktop Internet Explorer. It will have the same fast and fluid experience. In other words, we've designed WOA to look and feel just like you would expect. WOA enables creativity in PC design that, in combination with newly architected features of the OS, will bring to customers new *no-compromise* experiences.

As an in-depth engineering dialog, we tend to favor the long form for *Building Windows 8* posts, and this post is no exception. It does seem like a good idea to first provide a summary of the important items we are going to cover in detail in this post:

- **Windows on ARM, or WOA, is a new member of the Windows family that builds on the foundation of Windows**, has a very high degree of commonality and very significant shared code with Windows 8, and will be developed for, sold, and supported as a part of the largest computing ecosystem in the world. We created WOA to enable a new class of PC with unique capabilities and form factors, supported by a new set of partners that expand the ecosystem of which Windows is part.
- **WOA PCs are still under development and our collective goal is for PC makers to ship them the same time as PCs designed for Windows 8 on x86/64**. These PCs will be built on unique and innovative hardware platforms provided by NVIDIA, Qualcomm, and Texas Instruments, with a common Windows on ARM OS foundation—all running the same Windows OS binaries, a unique approach for the industry. PC manufacturers are hard at work on PCs designed from the ground up to be great and exclusively for WOA.
- **Metro style apps in the Windows Store can support both WOA and Windows 8 on x86/64**. Developers wishing to target WOA do so by writing applications for the WinRT (Windows APIs for building Metro style apps) using the new Visual Studio 11 tools in a variety of languages, including C#/VB/XAML and Jscript/HTML5. Native code targeting WinRT is also supported using C and C++, which can be targeted across architectures and distributed through the Windows Store. WOA does not support running, emulating, or porting existing x86/64 desktop apps. Code that uses only system or OS services from WinRT can be used within an app and distributed through the Windows Store for both WOA and x86/64. Consumers obtain all software, including device drivers, through the Windows Store and Microsoft Update or Windows Update.
- **WOA can support all new Metro style apps, including apps from Microsoft for mail, calendaring, contacts, photos, and storage. WOA also includes industry-leading support for hardware-accelerated HTML5 with Internet Explorer 10**. WOA will provide support for other industry-standard media formats, including those with hardware acceleration and offloading computation, and industry-standard document formats. In all cases, Microsoft seeks to lead in end-user choice and control of what apps to use and what formats to support.
- **WOA includes desktop versions of the new Microsoft Word, Excel, PowerPoint, and OneNote**. These new Office applications, codenamed “Office 15”, have been significantly architected for both touch and minimized power/resource consumption, while also being fully-featured for consumers and providing complete document compatibility. **WOA supports the Windows desktop experience including File Explorer, Internet Explorer 10 for the desktop, and most other intrinsic Windows desktop features**—which have been significantly architected for both touch and minimized power/resource consumption.
- **With WOA you can look forward to integrated, end-to-end products—hardware, firmware and WOA software, all built from the ground up**. Building WOA has been an ongoing engineering effort involving Microsoft, ARM licensees, PC makers, and developers of components and peripherals. These efforts spanned a wide array of subsystems that have been newly created or substantially re-architected for WOA. Partners will provide WOA PCs as integrated, end-to-end products that include hardware, firmware, and Windows on ARM software. Windows on ARM software will not be sold or distributed independent of a new WOA PC, just as you would expect from a consumer electronics device that relies on unique and integrated pairings of hardware and software. Over the useful lifetime of the PC, the provided software will be serviced and improved.
- **Around the next milestone release of Windows 8 on x86/64, a limited number of test PCs will be made available to developers and hardware partners in a closed, invitation-only program**. These devices will be running the same branch of Windows 8 on x86/64 as we release broadly at that time. These are not samples or hints of forthcoming PCs, but tools for hardware and software engineers running WOA-specific hardware.
- **The Windows Consumer Preview, the beta of Windows 8 on x86/64, will be available for download by the end of February**. This next milestone of Windows 8 will be available in several languages and is open for anyone to download.

This post is organized with the following sections: Working with partners, Providing apps, Engineering for ARM (which will go through the various subsystems), Developing for ARM, Delivering WOA PCs, and finally, Next steps.

We've also prepared a short video demonstrating WOA as described in the post.

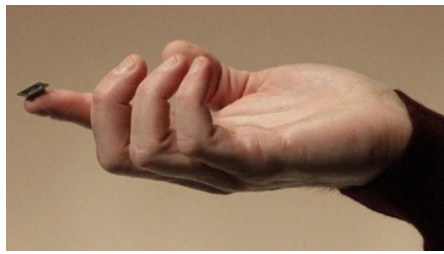
Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

Working with partners

Developing WOA begins as a partnership with companies that make ARM processors and package them together with the subsystems required to deliver the equivalent of a *motherboard*. Unlike the boards many are familiar with, you can think of a WOA board as a silicon package—a series of silicon layers bound together in an incredibly small form factor, called a *System on Chip* or *SoC*.



CES 2011 demonstration showing a System on Chip (SoC). (Julie Jacobson, Associated Press)

Each ARM licensee building these packages takes a different approach to selecting features, making product trade-offs, and designing the complete silicon package. These choices are what bring the diversity of different products built on ARM to the market. There is no single ARM experience, and as we have seen with other operating systems, even the same ARM CPU combined with different components, drivers, and software can yield different types or qualities of experiences. That is why from the start of the WOA project, we have been working with three ARM licensees: NVIDIA, Qualcomm, and Texas Instruments. Each brings different expertise and different approaches, and all will make a unique contribution to WOA. All of them have extremely successful ARM-based products in the market today, from tablets to smart phones to e-readers to embedded devices. We are fortunate to have the support of such amazing partners, and WOA is unique in working with such diversity.

A SoC package by itself is just the beginning. Delivering WOA PCs is a partnership with PC manufacturers who bring their expertise in manufacturing, system engineering, and industrial design and combine that with the engineering work of ARM partners to develop a complete PC. PC makers also bring expertise at selling PCs to consumers and businesses through a variety of channels and supporting those purchases over time.

Microsoft's role in this partnership is to deliver a Windows operating system that is tuned to this new type of hardware, new scenarios, and new engineering challenges. Our goal is to make sure that a reimagined Windows delivers a seamless experience from the chipset through firmware, through hardware, through the OS, through applications, and ultimately to the person interacting with the PC. This is a new level of involvement that brings with it a new level of engineering work across all of the parties involved. This new approach is about delivering a unique combination of choice, experiences, and a reliable end-to-end experience over the life of the PC.

Providing apps

Before we dive into the details of the changes to the underlying implementation of Windows to support ARM hardware, let's start at the top and talk about what apps Microsoft will offer (we're all happy to see a renewed use of the term "apps"—my first business card at Microsoft in 1989 read *Software Design Engineer in Apps Tools, Apps Division*.)

We have not yet announced the editions or SKUs we will have for any new Windows products, and during the pre-release phase we include almost all features in the software as a way of testing and exercising the full surface area of the product. You can expect that we will adjust the features included with the single WOA offering such that it is competitive in the marketplace and offers a compelling value proposition to customers of all types.

As we announced and demonstrated at //build/ and other forums, WOA has all the WinRT capabilities present in the Windows Developer Preview, and all the tools and techniques that you can use to build new Metro style apps for x86/64 are available to developers to also target WOA. Developers can use our tools to create native C/C++ code for maximal performance and flexibility, in addition to the C#, XAML, VB, and HTML5 based tools, to target apps for WOA, so long as their code targets the WinRT API set. Additionally, developers with existing code, whether in C, C++, C#, Visual Basic, or JavaScript, are free to incorporate that code into their apps, so long as it targets the WinRT API set for Windows services. The Windows Store can carry, distribute, and service both the ARM and x86/64 implementations of apps (should there be native code in the app requiring two distributions).

We have also previously demonstrated Microsoft's new Metro style apps for connecting to cloud-based services like Hotmail, SkyDrive, Messenger, and—through those services—a wide variety of third parties. For example, our mail app connects to industry-standard [EAS](#), which covers an array of enterprise and consumer-based mail, calendaring, contacts. With existing [Live Connect capabilities](#), you can chat with your Facebook friends, or keep up-to-date on your LinkedIn or Twitter feeds all in a Metro style app—these are just a couple of examples of over 100 different services globally that you can connect to your Microsoft account. These apps are provided with WOA, but of course, people can remove these, set different defaults, or use the Windows Store to get similar apps from third parties. In addition, any Metro style app in the Windows Store can work with any service it chooses, with or without using any Microsoft services—and this spans the range from sign-in, communications, in-app payments, to advertising services.

In the next pre-release of Windows 8 you will also see Metro style apps available from Microsoft that support a wide variety of industry-standard media and document formats, along with Internet Explorer 10 which supports the standard HTML5 web platform. We believe that the level of standards support provided in WOA is among the best in class, and comparable in scope to competitive products. And of course, our intent is to lead in the industry in providing end-user choice and control over the apps on your system and what you choose to run.

The availability of the Windows desktop is an important part of WOA. The desktop offers you a familiar place to interact with PCs, particularly files, storage, and networking, as well as a range of peripherals. You can use Windows Explorer, for example, to connect to external storage devices, transfer and manage files from a network share, or use multiple displays, and do all of this with or without an attached keyboard and mouse—your choice. This is all familiar, fast, efficient, and useful. You'll have access to a deep array of control panel settings to customize and access a finer-grained level of control over your system, should you want to. And if you've used the Developer Preview with a touch-capable PC, you know that the desktop user-interface has been refined for touch interaction with improved user-interface affordances.

At the same time, WOA (as with Windows 8) is designed so that customers focused on Metro style apps don't need to spend time in the desktop. Availability of the desktop incurs no runtime overhead. It is just there should you want or need it. Below, we will describe the technology behind the scenes that goes into making sure that the availability of the desktop does not compromise system security, reliability over time, performance, or power consumption of a WOA PC. To those of you who've tried out the Developer Preview, you'll notice that the user experience has continued to evolve and you will see a broad set of improvements in the upcoming Consumer Preview.

Some have suggested we might remove the desktop from WOA in an effort to be pure, to break from the past, or to be more simplistic or expeditious in our approach. To us, giving up something useful that has little cost to customers was a compromise that we didn't want to see in the evolution of PCs. The presence of different models is part of every platform. Whether it is to support a transition to a future programming model (such as including a virtualization or emulation solution if feasible), to support different programming models on one platform (native and web-based applications when both are popular), or to support different ways of working (command shell or GUI for different scenarios), the presence of multiple models represents a flexible solution that provides a true no-compromise experience on any platform.

Within the Windows desktop, **WOA includes desktop versions of the new Microsoft Word, Excel, PowerPoint, and OneNote, codenamed "Office 15"**. WOA will be a no-compromise product for people who want to have the full benefits of familiar Office productivity software and compatibility, an industry-leading hardware-accelerated web browser, apps from Microsoft, and access to apps in the Windows Store.

This creates a WOA PC with the full power of apps, media consumption, entertainment, mobility, and productivity in one place—a true no-compromise experience. The new Office applications for WOA have been significantly architected for both touch and minimized power/resource consumption. This engineering work is an important part of being able to provide Office software with WOA, as these are not simply recompilations or ports, but significant reworking of the products with a complete and consistent user experience and fidelity with their new x86/64 counterparts.

You can learn more about the next version of Microsoft Office, codenamed "Office 15," [on the Office Exec blog](#).

Engineering for ARM

Enabling Windows to run super well on the ARM architecture is a significant engineering task. We undertook this work because when you look to the future you can see that so many of the capabilities that have been added to Windows over the years are things that customers will inevitably desire or require in the types of devices supported by today's ARM-based products—changes in form factors and the desire for mobility only add to the scenarios and capabilities we all desire in our search for no-compromise PCs. While it is tempting to make bold statements about "starting over," we believe in the evolution of technology assets when the foundation is strong. The foundation of Windows, the core, is the most solid, scalable, and secure one around. Our desire

to deliver a no-compromise experience motivates our efforts.

We also know that there are elements of Windows that require re-engineering in order to meet customer expectations for reliability over time, power consumption, resource utilization, and *instant* connectivity and availability. Obviously, all of this work is relevant to our Windows 8 on x86/64 product too, and much of what we have done for ARM will be applicable to the exciting new products coming from Intel and AMD (which are not the subject of this post). ARM affords us a chance to look at assumptions in OS behavior and programming model in order to deliver significant improvements.

One of the new aspects of WOA you will notice is that you don't turn off a WOA PC. WOA PCs will not have the traditional hibernate and sleep options with which we are familiar. Instead, WOA PCs always operate in the newly designed *Connected Standby* power mode, similar to the way you use a mobile phone today. When the screen is on, you have access to the full power and capabilities of the WOA PC. When the screen goes dark (by pressing the power button or timer), the PC enters a new, very low-power mode that enables the battery to last for weeks. All along, however, the system dynamically adjusts power consumption and is always on the lookout for opportunities to reduce power to unused parts of the system. For end-users, a unique capability of WOA is that *you* are in control of what programs have access to background execution so that those apps are always connected, and information like new mail is always up to date. Connected Standby permeates the engineering for WOA PCs from the hardware through the firmware, OS, WinRT platform, and apps. Connected Standby won't be limited to the ARM architecture and we are actively working on these capabilities for x86/64 SoC products as well.

Today, we are familiar with a PC experience where hardware that runs Windows built on x86/64 adheres to a set of technical specifications that allow one distribution of Windows code to install and run on a wide variety of PCs. This has enormous benefits of scale. This openness is also the hallmark of the PC revolution and represents the collective work of the industry since about 1980. When new hardware comes along that is broadly supported, these baseline specifications evolve, and the PC architecture moves forward. **Absolutely nothing about this approach will change for Windows 8**—as millions have experienced with our Windows 8 Developer Preview, **Windows 8 will run on every Windows 7 logo PC**, and will run all of the existing software and peripherals designed for and supported on Windows 7 (when supported on Windows 8 by the manufacturer, of course).

The approach taken by ARM Holdings, the licensor of ARM products is, by design, not standardized in this manner—each device from each manufacturer is unique and the software that runs on that device is unique. There is of course a standard instruction set and CPU architecture, one that is always improving (for example, adding 64-bit support and multiple cores), but many of the connections between the CPU and other components are part of the innovation each licensee brings to the ARM platform. Commonality across devices can occur under the hood, but is not applicable or significant to consumers. End-users are technically restricted from installing a different OS (or OS version) on a device or extending the OS, so this is generally not possible, and rarely supported by the device maker. Device makers work with ARM partners to create a device that is strictly paired with a specific set of software (and sometimes vice versa), and consumers purchase this complete package, which is then serviced and updated through a single pipeline. The cross-partner, integrated engineering of these embedded devices is significant. In these ways, this is all quite different than the Windows on x86/64 world.

With WOA, we set out to define a new way of developing a computing platform. We architected our approach to ensure that software and peripherals can all benefit from the diversity enabled by the ARM architecture, along with the choice of form factors and manufacturers, and the openness of the platform. At the same time we are making a commitment to customers that WOA will be consistent in capabilities, experience, and baseline performance across this spectrum. To those familiar with the Windows Phone 7 approach, the *chassis specification*, WOA shares some of those elements. The specifications being implemented for WOA allow for more diversity across many dimensions, combined with the same commitment to engineering and product excellence—all while running the same OS binaries across WOA PCs.

Engineering for ARM starts with the work we did to architect the Windows kernel so it could boot and run on ARM. As you might imagine, this was a significant effort. Some might believe that this is work along the lines of *porting* or merely *re-compiling* the code for a new instruction set. There's much more to the work than that when it comes to the kernel and the parts of Windows that connect with hardware. Along with the kernel work, we also had the work to develop the ARM compilers and tools (including Visual Studio), for building Windows.

At the higher levels—the application layers—the code is significantly portable because of our long history of running on multiple architectures (x86, x64, PowerPC, Alpha, MIPS, IA64, and so on). Even the kernel itself has a significant amount of code that can be ported. At the hardware/software seam and all the places that make assumptions about how an OS interacts with hardware, Windows has been reimagined for this new platform. To put some acronyms around this, the ARM definition does not require support for some common subsystems such as the PCI bus or SATA. There are analogous concepts implemented by each ARM implementation, but those are not always common. All of this was done over the course of iterating on three major revisions of ARM hardware since the start of the project.

Let's look at some of the types of work undertaken as part of this effort, which we referred to internally as “porting” despite the fact that it is so much more than that. Keep in mind that all of this work has been going on in parallel with the development of the user experience, Windows Store, WinRT, and new features across Windows 8.

Getting ready to port

Before the porting work could even begin, we needed an ARM compiler and tool chain for building Windows. Since other products at Microsoft (such as Windows Phone and Embedded) use ARM processors, we had these pieces, but further improved them to build Windows. These tools are going to be available to developers, and if you're using C#/VB/XAML/HTML5 in the Windows 8 Developer Preview, then you're already on board. C/C++ requires ARM native hardware for testing, which we'll talk about below.

Booting the core of Windows

Once we had the tools, we could start porting the Windows boot environment and developing system firmware specifications. We even prototyped the firmware ourselves. There are several pieces to this:

- **UEFI firmware** is the lowest layer of a WOA system and provides consistent services for loading the OS. For WOA, we created firmware to bootstrap the system that we handed off to our partners. WOA systems also include a firmware-based **TPM** for trusted boot and storage encryption. Using the TPM, for example, we've implemented trusted boot which verifies that the system hasn't been tampered with by malware.
- **ACPI firmware** is used for plug and play enumeration of devices in the platform during boot, and is also responsible for power management of devices outside the SoC (such as sensors, touch controller, etc.). Over the years, the PC has standardized with plug-and-play busses and ACPI, so that operating system software and drivers can “walk the tree” to find everything in a PC. With SoC embedded designs, there is no “tree” or ability to discover what is connected to a SoC, or even how the SoC is connected. During Windows 8, we worked to define a new standard to describe the configuration of the system with tables, so software can simply read the table and configure the system.

From the firmware, the system can then load the boot manager, boot loader, and in turn the kernel, HAL, and boot device drivers.

- The **Windows Hardware Abstraction Layer (HAL)** supports variations in core system resources (timers, DMA, interrupt controllers). Windows was designed from the beginning to support multiple instruction set architectures (ISA), and the HAL is key to adapting to different system architectures that often come with a new ISA. By abstracting the hardware layers, the OS itself doesn't have to be modified to accommodate a new SoC for core system resources. The variation across ARM platforms is significant enough that we architected the HAL to support a new level of capabilities of abstraction. New to the Windows 8 HAL is the ability for each of the core system resources to be plugged in via an extension to the HAL, kind of like a driver for the interrupt controller.

Devices and busses

In order to load device drivers and continue Windows boot, we had to build several new drivers for new types of low-power busses, plus device drivers that support connections to those busses.

Our device strategy uses standardized protocols and class drivers extensively. Our first example below is the HID over I²C driver which we use for touch controllers and many sensors, another is the class driver for USB connected mobile broadband radios. Of course, Windows has many class drivers inside, which you experience when you plug in a wide variety of USB devices, such as storage, mice, or keyboards.

- Low power serial busses such as I²C / UART will be normal on ARM PCs and less common on x86 PCs. These busses generally have a lower data transfer rate, but also use very little power, in some cases 10x less. Support for these busses is key to reducing the overall power use of WOA and extending battery life. Collectively, we call these busses **Simple Peripheral Busses (SPBs)** and we've developed new interfaces in WOA for them. Once we had the interfaces, we had to address a gap. In Windows, we have many device classes that are natively supported over USB via class drivers. These classes are undefined over I²C, and hence they lack class driver support. One popular class of devices is Human Interaction Device (HID) protocol based devices. HID is the protocol of choice for devices like keyboards, mice, touchpads, speakerphones, buttons, touchscreens, etc. By defining a standardized protocol and

implementing driver support for HID over I²C, we can work with partners to adapt the firmware of their I²C-based devices to work with a single class driver. For example, by supporting HID over I²C, touch controllers can use that interface and leverage the input support that Windows already has.

- **SD I/O** allows you to connect low power Wi-Fi radios. Radios in current PCs are connected via USB or PCI-E. We added SD I/O support to preserve high data rates (100 MB/s) while still improving battery life. Wi-Fi support on WOA also allows efficient offloading to maintain connections in connected standby while using very little power.
- **Embedded MultiMediaCard** storage (eMMC) is a *de facto* standard for storage on ARM devices (since most do not support SATA). This was an interesting challenge for us, since Windows expects a fast disk and very high bandwidth data transfer. In addition to supporting eMMC, we made several OS performance optimizations to reduce and coalesce storage I/O, resulting in fewer reads and writes to storage.
- The **General Purpose I/O (GPIO)** driver supports connecting buttons, interrupts or other I/O to the ARM processor.
- In addition to the GPIO driver, there's also a **button driver** for the Windows, power, and volume buttons. Buttons aren't standard on ARM devices. Each system requires a specific driver for all hardware buttons.
- We built a **new power framework** for managing SoC-wide power, total platform power, and the connected standby on/off usage model.

Getting to the Start screen

Once the firmware, HAL, boot services, boot devices, and busses were up and running, we were ready to bring up the rest of the system and get to the desktop and the Start screen.

- ARM SoCs for WOA have **DirectX capable GPUs (DX)** for accelerated graphics in Internet Explorer 10, in the user interface of Windows, and in Metro style apps. Taking advantage of a DX capable GPU is essential for delivering a responsive user experience. For each WOA target, the ARM partner has created a DX-compatible graphics driver. This is a significant undertaking of very complex code since today's GPUs are even more complex than the CPUs. To bring up Windows 8 on these new SoCs that did not yet have a graphics driver, and since ARM SoCs do not have the industry-standard VGA subsystem to fall back on for compatibility mode, our graphics team wrote a soft GPU driver that was capable of working directly against the hardware frame buffer. Aside from enabling development, it also enabled us to reimagine other things in Windows using the soft GPU driver when the normal GPU driver isn't available. For example, when running Windows Setup, or in those rare cases when Windows has a "bluescreen," we were able to give it a friendlier look and even localize it, so that even bad news can be presented more nicely across all platforms. This is a small example of work which is common to the x86/64 architecture as well.
- WOA PCs use hardware support for offloading specific work from the main processor to **integrated hardware subsystems**. This improves performance and battery life. For example, while watching a movie, the processing is done with multimedia offload (to a dedicated processor for example), and all other processing is minimized. Since the multimedia offload is optimized for playback, you can watch several movies without running out of battery or the PC could be designed to be even thinner and lighter. Another example is if you're working on a document and watching a movie at the same time, the movie is running on the offload hardware, which helps the overall system responsiveness. WOA takes advantage of several types of offloads including multimedia encode and decode as well as security offload for Bitlocker and EAS. This type of engineering also applies to x86/64, which also support offloading, and was introduced in Windows 7.

Connected device services

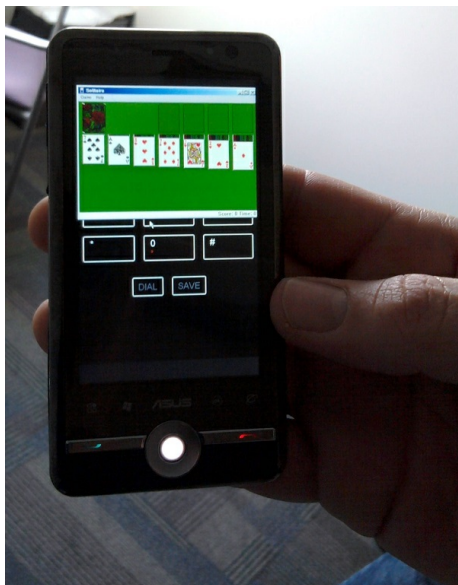
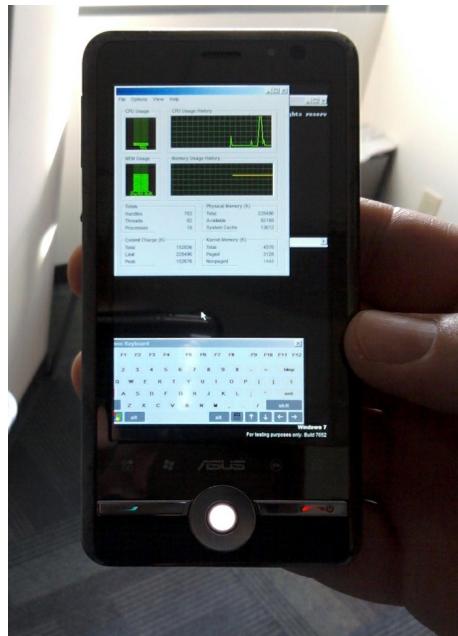
At this point we had the majority of the system running, and it was time to bring on the services to support the full breadth of Windows. These are common across the architectures that Windows supports, so that developers can take advantage of them in Metro style apps.

- **Mobile broadband (MBB)** class driver. By creating a class driver, we've made it much easier to add broadband capability to all Windows PCs. While WOA was a catalyst for this work, the entire ecosystem benefits.
- **Printer** class driver. For Windows 8, we rearchitected the print infrastructure to add class driver support. The majority of printers selling today are supported using the class driver, which means you'll be able to "plug and print" on WOA without additional drivers. While the new architecture was needed for many reasons, we had printing from WOA PCs in mind from the beginning.
- **GPS**. Windows offers a location provider that can triangulate a PC's location via Wi-Fi access points and a backing database. In addition, systems that have Mobile Broadband will also have integrated Global Navigation Satellite System (GNSS, aka GPS in the US) receivers to provide accurate location while navigating outdoors. The location platform plays a pivotal role in optimizing for power and accuracy by choosing the right location data provider to use based on the precision requested by the application.
- **Sensors (accelerometer, rotation, gyro, compass, magnetometer)**. A recent post described Sensor Fusion and how we've added support for sensors in Windows. This work also applies across all SoC-based architectures and utilizes the HID over I²C protocol.
- **Bluetooth**. WOA supports Bluetooth LE and the same profiles as Windows 8 on x86/64 and connectivity to the Bluetooth radio using low-power UART.
- **MTP over USB and IP**. Windows on ARM provides users with the ability to connect their portable devices (like mobile phones, music players, cameras) to their systems using the Media Transfer Protocol (MTP). These MTP-compliant devices can connect over USB or IP by leveraging inbox Windows class drivers, and allow users to exchange data with their favorite Metro style apps.
- **Windows Update-based servicing**. For *all* platform code (OS, drivers, system and device firmware), each WOA system will be serviced through Windows Update (WU), from top to bottom. We've added support in WU for securely and robustly updating the system firmware on WOA systems, as well as driver targeting, which means that each device will get the drivers that have been verified to work best with it.

As you can see, some of this engineering work is strictly adapting to the new hardware platform. Some introduces substantially new types of hardware support. **In large part this work accrues to the x86/64 platform especially cutting edge products, such as the new low power ATOM® processors, demonstrated by Intel at CES.**

A significant amount also propagates to the application layer and becomes defining elements of the new WinRT APIs introduced at //build/. For example, while we engineered the kernel to support Connected Standby, delivering great battery life is really part of the overall WinRT application model and even the toolset, and all of that applies across WOA and Windows 8 on x86/64.

As we mentioned, a portion of Windows is generally built with code that can be made to work on ARM in a technically straightforward manner. These subsystems include the Windows desktop and applets and supporting APIs, though we needed to significantly re-architect all of them for better resource and power utilization. In fact, here is an early photo of an ARM device (an early Windows phone) running the full Windows desktop. Early in the development of WOA, the only hardware we had were existing ARM devices such as phones (ARM tablets didn't yet exist). We just thought you would enjoy a few **fairly early** photos I captured of debug WOA all loaded in RAM (unretouched). Note: *This is not a product plan or even a hint at a product.*



Testing

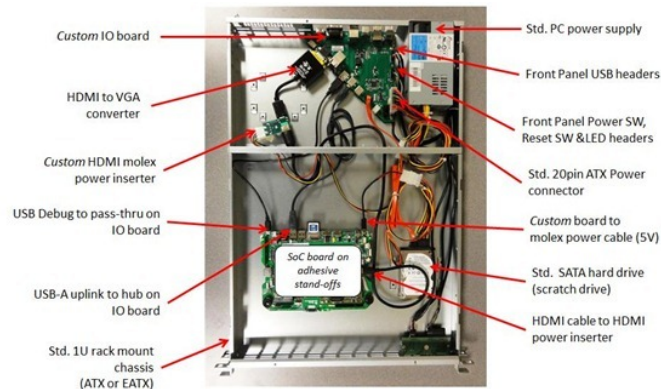
You might be wondering how we are testing WOA in our labs. For x86/64 we run massive labs (thousands of machines, real and virtualized) and highly automated test runs for every single daily build. WOA required us to reimagine our own lab and test processes. For testing x86/64, it is simple to just order thousands of rack-mounted servers, or even virtualize them; for ARM, there are no standard rack-mounted servers that can run WOA. Since we're doing highly integrated hardware/firmware/software development, and virtualization is not helpful, we had to devise our own approach.

We consolidated hundreds of ARM development boards along with a custom I/O board into a rack assembly connected to our testing infrastructure. Our original design focused on density supporting 300 ARM devices in one rack, but we ultimately preferred the diagnostics and availability of a custom I/O board in the 1U setup.

We designed our own 1U chassis that fits into a standard server rack. Either a full form-factor device or just the motherboard can be dropped into this chassis. Once fully assembled, the SoC board in conjunction with the IO board and chassis looks, feels and operates like a standard rack mount PC and fits right in with existing lab infrastructure.

Each 42U rack holds 32 WOA chassis plus network switches, debug host PC, and USB hubs. By March we will have over 100 fully populated *racks* for WOA testing.

We also had to port our test tool infrastructure and tests too, which was no simple challenge, but this ensured that we would cover WOA with the same rich automation used to validate Windows 8. Here's a photo of our newly devised test rack, and the board and debugging ports that it hosts:



Developing for ARM

In practice all of this is *even* more in-depth than it looks. We also took this chance to do a very significant re-engineering of every Windows subsystem. In the course of building WOA and Windows 8, we invested a huge amount of energy into changing all of Windows to work better at minimizing overall power consumption and resource utilization, while simultaneously delivering improved real-world performance for existing application workloads. In previous posts on boot, power management, and memory usage, you have seen the results of some of this work.

Previously we have detailed that **WOA will not support any type of virtualization or emulation approach, and will not enable existing x86/64 applications to be ported or run.** Supporting various forms of emulation runs counter to the goal of delivering a product that takes a modern approach to system reliability and predictability—by definition, existing code has not been optimized for the platform the way WOA has. Virtualized or emulated software will consume system resources, including battery life and CPU, at unacceptable levels. Emulation and virtualization of existing x86/64 software also **require** the traditional PC environment of mouse and keyboard, which is not a good assumption for WOA PCs.

If we enabled the broad porting of existing code we would fail to deliver on our commitment to longer battery life, predictable performance, and especially a reliable experience over time. The conventions used by today's Windows apps do not necessarily provide this, whether it is background processes, polling loops, timers, system hooks, startup programs, registry changes, kernel mode code, admin rights, unsigned drivers, add-ins, or a host of other common techniques. By avoiding these constructs, **WOA can deliver on a new level of customer satisfaction: your WOA PC will continue to perform well over time as apps are isolated from the system and each other, and you will remain in control of what additional software is running on your behalf, all while letting the capabilities of diverse hardware shine through.**

Our focus on delivering a new level of security for consumers using WOA is paramount. In one public event, we were asked if we would “make it easy for existing viruses and malware to run.” Now you can see the answer is decidedly, “no.” In fact, WOA only supports running code that has been distributed through Windows Update along with the full spectrum of Windows Store applications. As we all know, security is an industry-wide, multi-dimensional challenge and no system or platform can make broad claims without considering many factors.

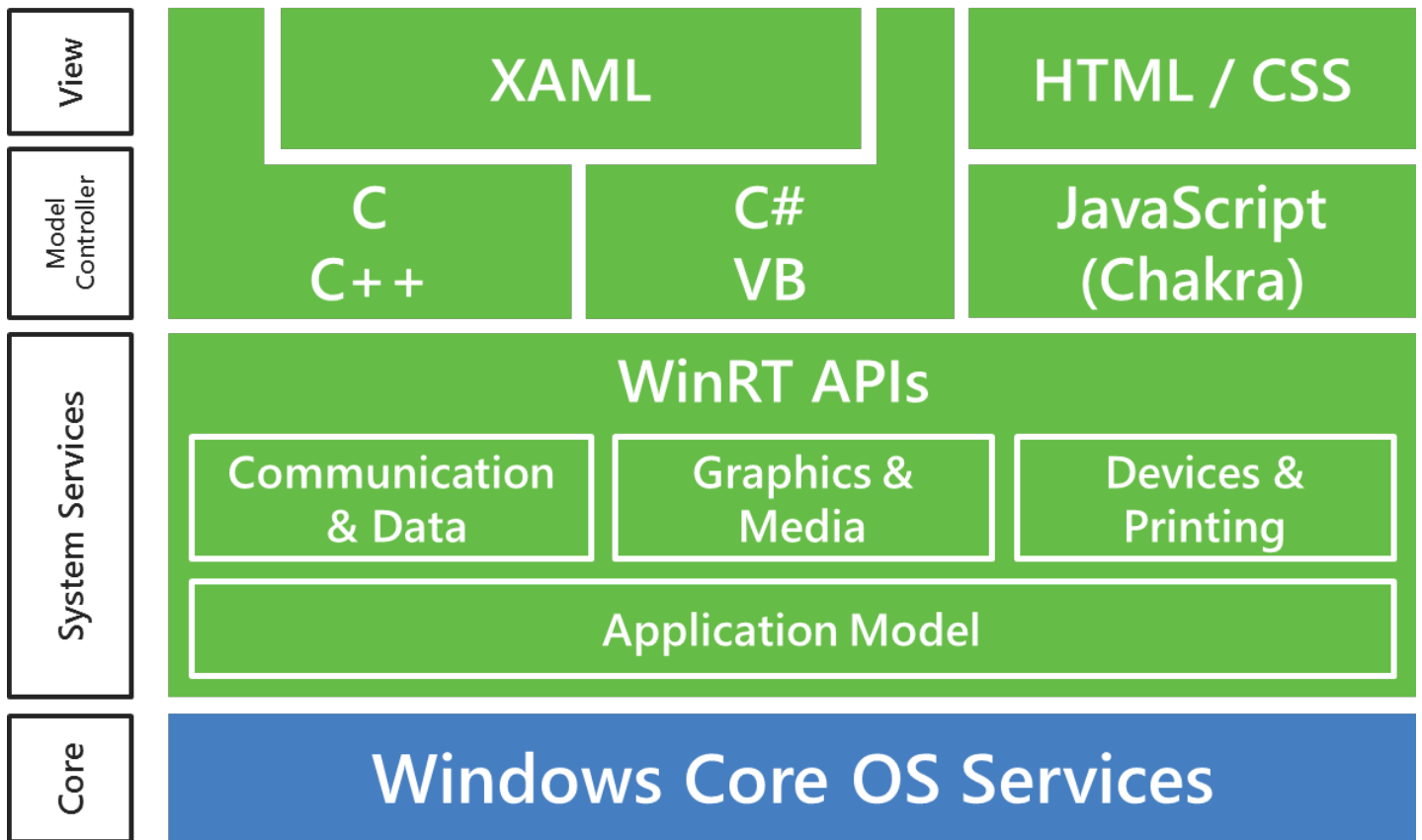
If you need to run existing x86/64 software, then you will be best served with Windows 8 on x86/64. If you're already considering a non-Windows device, then we think WOA will be an even better alternative when you consider the potential of form factors, peripherals, Windows Store apps (and developer platform), and Office applications as well as a broad set of intrinsic Windows capabilities.

Of course, we recognize that many developers at Microsoft and elsewhere rely on existing programming techniques, and that a transition to WOA will require an investment. Developing apps based on WinRT addresses these and many other challenges from the start—WinRT is designed to provide the full expressive power required for modern software while avoiding the traps and pitfalls that can potentially reduce the overall experience for consumers and does so with a deep commitment to tools, languages, and developer support.

Developers wanting to reach WOA with existing apps have two options. Many apps will be best served by building new Metro style front ends for existing data sources or applications, and communicating through a web services API. This approach will be quite common for line-of-business applications and many consumer web properties, and represents the best way to tap into the power of a rich user interaction model where you can also interact across and share information with other new apps. Of course, these do not need to be just front-ends, but could operate on local data too, since WOA provides full access to files and peripherals. Other existing applications will be well served by reusing large amounts of engine or runtime code, and surrounding that with a Metro style experience. This will take some time, and represents a way for applications that are composed of significant intellectual property to move to WOA and WinRT. In all cases, WinRT represents the new set of Windows OS services that developers can use to build software that is *Designed for Windows 8*.

Returning to our architecture diagram from //build/, from a third-party developer perspective, the best way to think of WOA is as the expression of the Metro style platform, which shares the Windows Core OS with all the other Windows products. The Windows Core OS has been tuned and architected to support the ARM platform and is there to support the WinRT APIs and programming model used by third parties.

Metro style Apps



The topic of engineering for ARM is a broad one and has occupied many on the Windows team for the course of the project. The next step is delivering WOA code more broadly, but that starts with how we're going to bring WOA PCs to market.

Delivering WOA PCs

Since the conclusion of the Windows 7 project we have been working with PC makers on the evolution of Windows and the creation of Windows 8. There is a vast amount of joint work that goes on in order to bring new PCs to market—the “Designed for Windows” logo that you see on a PC represents the collective work of a wide array of partners sharing a commitment to bring new and exciting PCs to market. The model we used and will continue to use to bring x86/64 PCs to market as an industry is the same as we have always used—we will introduce new technologies such as USB 3.0, UEFI, touch, and sensors and support those in a new release of Windows with new hardware. This is a collaborative and ongoing effort, with a great many improvements to be introduced with this product cycle.

Delivering WOA PCs is building out a new system for the first time—a completely new ecosystem of PCs providing opportunities for PC makers to bring to life a new generation of PCs with new capabilities. **We describe these PCs as being focused on achieving new levels of capability along three dimensions: thin and light in industrial design, long battery life, and integrated quality.**

Because of the necessarily tight connection between SoC, peripherals, firmware, and the OS, WOA PCs should be thought of as joint engineering that goes well beyond industry partners merely collaborating. This is an effort where software people on the Windows team end up debugging silicon with soldering irons, and hardware engineers end up in Visual Studio, debugging timing issues with user interface code. Thus every WOA PC is a new engineering effort that starts with the selection of components and continues through with firmware, drivers, final assembly, and unique apps from PC makers. We also bring new ARM designs up on simulation and emulation platforms to get it right at the start, even before the silicon is available. And we are bringing the ecosystem together to do total platform design for low power, which includes not just a great SoC but efficient radios, sensors and even higher efficiency DC power infrastructure. It all matters for super thin and light PCs, with great battery life, and high-quality engineering providing a great experience with apps and services that are *Designed for Windows 8*.

While each WOA PC offered will be unique, the role of Windows is to present a consistent experience to customers while **allowing the unique and innovative hardware to shine through—the very definition of an OS**. To achieve this we have been working with multiple ARM licensees as mentioned—Texas Instruments, Qualcomm, and NVIDIA. Each has been working with partners that will bring WOA PCs to market. **These PCs have all been designed and manufactured expressly for WOA**. From the chipset through the firmware and drivers, the work is optimized to be great for WOA. Partners are working hard on creative industrial designs and form factors that will include more than tablets. These are all under development today. **Our collective goal is for PC makers to ship WOA PCs the same time as new PCs designed for Windows 8 on x86/64, using the latest generation of those platforms from low-power to high-performance.**

While not the topic of this post, we do want to assure you that, **when a consumer buys a WOA PC, it will be clearly labeled and branded so as to avoid potential confusion with Windows 8 on x86/64**. The PC will come with the OS preinstalled, and all drivers and supporting software. WOA will not be available as a software-only distribution, so you never have to worry about which DVD to install and if it will work on a particular PC.

WOA PCs will be serviced only through Windows or Microsoft Update, and consumer apps will only come from the Windows Store, so you never have to worry if a program will run because you are not downloading or installing from a DVD outside of the store experience. A WOA PC will feel like a consumer electronics device in terms of how it is used and managed. For example, as previously detailed, the new refresh and reset functionality will be available, and for WOA this provides the equivalent of a “clean install” or imaging.

Next steps

There's much more to this story, and we plan on more posts detailing the engineering of WOA and all the work that went into building the OS based on the dialog that follows this post. Many are keen to get their hands on the software. But of course there is no existing hardware to use as there is with Windows 8 on x86/64, which can make use of PCs designed for Windows 7. We are approaching a step in the project where we intend to broaden the distribution of the WOA software with development hardware.

To run this release, a low volume of test PCs specifically designed for WOA will be made available starting around the next Windows 8 milestone. These devices are for developers and hardware

partners, and do not represent consumer form factors, by any stretch of the imagination. They have diagnostic tools and ports. They are designed to be opened and debugged. They do not have the final components or firmware (or power or thermal management) that a commercially available device will use. They are made of low-cost plastic. You might have seen devices similar to these on display at CES or demonstrated there, and all of our previous demonstrations have used some form of these test PCs. These PCs do represent WOA and the experience—but they no more represent the final experience than does the current state of x86/64 Windows 8. They will be running the same branch of Windows that will be made available to x86/64 testers at our forthcoming development milestone.

These PCs are expensive to make and distribute because they are basically low-volume custom PCs. They will be made available through our developer evangelism efforts. We are talking about this not to tease you or to solicit nominations, but because we know word will get out and images of these will be on the web. The devices are all already spoken for and allocated. On the one hand it seems a little cruel to dangle this in front of you, but on the other hand it is worth considering that this level of transparency is a hallmark of how we develop Windows. The scale of the Windows 8 project is significant, and combined with the degree to which we are forthcoming with information and dialog about our decisions, it is without precedent.

By the end of the month, the *Windows Consumer Preview* (the beta) of Windows 8 on x86/64, will be made available for download. We changed the name of the beta because recently the term has come to mean something very different than just a "testing release available for free to try out," so we did not want to add to the confusion. In keeping with the level of openness described, there is no pre-registration or admission to a test program—just download it and install it on a Windows 7 logo PC (although VMs are supported, that is not the best way to try out the consumer experience). We have made a ton of progress and there are many significant changes since the Windows Developer Preview 5 months ago. **As a reminder, we're still building Windows 8 and WOA, and there is much work to be done to go from pre-release to release. Quality remains our priority. The code is not done.**

We are very excited to be approaching this milestone. The responsibility of developing a new release of Windows is humbling, and the challenges of releasing an entire new platform such as WOA are both energizing and daunting. We look forward to welcoming everyone to the Windows Consumer Preview in short order.

On behalf of the Windows team,

Steven Sinofsky

Enabling accessibility

Steven Sinofsky | [2012-02-14T10:00:00+00:00](#)

Windows 8 is a product we design for an incredibly broad spectrum of people around the world. One of the areas where we have worked to deliver an even greater level of innovation is in ensuring that Windows 8, particularly the new Metro style experience, is accessible to everyone regardless of their physical abilities. In this post we will talk about the engineering work that goes into the features we refer to as “accessibility” – though as you will see, many of these features are broadly applicable and just make the product better for everyone. If you are interested in Microsoft’s overall efforts in accessibility and related topics, please be sure to check out www.microsoft.com/enable. This post is especially important for developers building Metro style apps for inclusion in the Windows Store, as we are asking you to test the accessibility of your application prior to submission. I encourage folks who have never seen these tools in action to learn about them through the video. The upcoming beta will be a great chance for everyone to experience the product.

An important note. With the next public release of code (later this month) we will see a significant improvement in the capabilities described in this post, but we still have work to do between beta and RC especially with regards to working with the latest releases of third party tools. I just want to make sure folks know that this post talks about improvements in the next release as well as functionality that will still be improving as we get to the release candidate.

This post was authored by Jennifer Norberg, a senior program manager lead on our HID team.

–Steven

We want all users to be able to experience Windows 8 Metro style apps on their desktops, laptops, or the new touch-capable devices. This includes people with disabilities who rely on assistive technologies to use the PC.

About 15% of the world’s population has a disability¹. In the United States alone, 49.6 million people have a disability² and 45 million in Europe³. When it comes to interacting with computers, these disabilities affect individuals in a number of ways:

- **Visual impairments** include color vision deficiency, low-vision and blindness – all of which may impact the individual’s ability to see content displayed on the screen.
- **Mobility impairments** include arthritis, cerebral palsy, Parkinson’s disease, multiple sclerosis, and paraplegia, which impact the ability to use the keyboard and/or mouse to interact with the PC.
- **Hearing impairments** include conditions ranging from mild hearing loss to total deafness, and impact the individual’s ability to experience audio content generated by the computer.
- **Cognitive impairments** impact an individual’s learning and language skills, the ability to comprehend words, and difficulty with memory, solving problems, or perceiving sensory information.

The rates of individuals with disabilities are also increasing across the world due to the aging population and increases in chronic health conditions. One of the consequences of the global aging phenomena is the impact it will have on the workforce. For example, in the US, workers aged 55 and older are anticipated to increase from 18.1 to 23.9 percent by 2018⁵. That is more than one in five workers. Functional limitations as a result of aging (for example, presbyopia, the gradual loss of the eyes’ ability to focus actively on nearby objects, a condition that usually becomes noticeable in one’s mid-40s and continues to worsen until around age 65) will impact an older workforce’s ability to use technology that isn’t easy to see. As a result, there will be an increase in the number of working-age adults who are likely to benefit from the use of accessible technology.

New technologies and designs are especially difficult for people with disabilities to adopt because many new technologies are not made accessible when they are first released to the public. We have heard this concern about previous versions of Windows and we want to ensure that everyone can experience Windows 8 right away by providing a comprehensive accessibility platform for the desktop and Metro style features.

Our accessibility goals in Windows 8 are to:

1. Improve the assistive technologies that are components of Windows, and provide a good experience with the Metro style UI.
2. Provide developer tools that have baseline accessibility built in, so that accessible Metro style apps are available in the Store.
3. Engage assistive technology vendors (ATVs) to adopt Windows 8 and build upon the accessibility scenarios.

Each of these goals and audiences are discussed in detail in this blog

Past investments in accessibility

Before we look forward, let’s look back on the history of accessibility in Windows. In past releases, we established a foundation called UI Automation (UIA). UIA is used by developers to provide information about their code, and it’s how assistive technologies (ATs) access and use the information from the developers applications.

We’ve also shipped ATs as components of Windows:

- **Narrator** is Windows' built-in screen reader that allows people with visual impairments to interact with their system and applications. User feedback on previous versions of Narrator has consistently been that it needs to respond faster, read more controls, and support more languages.
- **Magnifier** is a tool in Windows to make text and graphics large enough to see for people with low vision. This was initially shipped in Windows 98, and was updated significantly in Windows 7 with the ability to magnify the full screen. This change received positive feedback. However, there were still issues with Magnifier, as it sometimes conflicted with settings for High Contrast colors.
- **Speech recognition** initially shipped in Windows Vista to aid people with mobility impairments to navigate and use their PC. User feedback on this feature has been really positive, telling us that the accuracy in speech recognition is good, it transcribes your voice to text quickly, and it is able to handle some uncommon words.
- **On-screen keyboard** has been available to those with mobility impairments since Windows XP.

While these Windows ATs cover a range of impairments, Windows depends on the rich ecosystem of AT vendors to cover the broad diversity of disabilities, and fully supports innovation in the ecosystem. This does not change with this new release of Windows. While we have focused on improving the ATs that we provide as Windows components and are providing support for new scenarios like the Metro style UI, we are also continuing to provide a rich platform and ecosystem where AT vendors can thrive.

Accessibility improvements in Windows 8

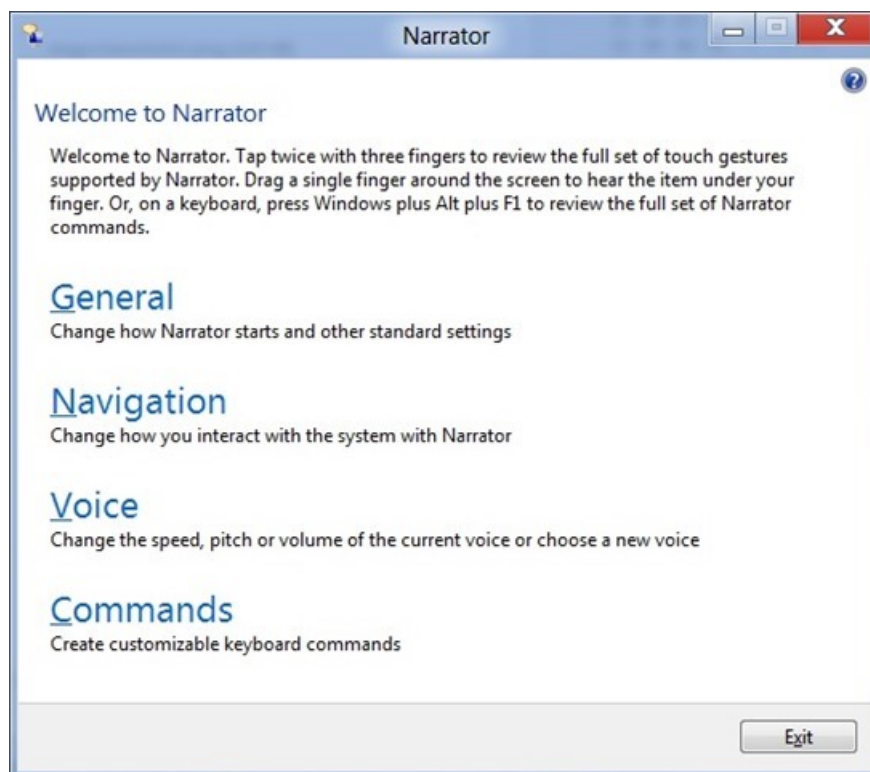
With each new release, we collate and respond to user feedback. It is clear that users want richer AT offerings to be included with Windows 8. In this release, we invested in the following areas to support this feedback:

- We redesigned Narrator to improve its performance so that it quickly reads out what you have selected.
- We added more languages and voices to Narrator to support additional countries and preferences.
- We updated components and features within Windows to leverage UI Automation that allows them to be read by Narrator.
- We updated UI Automation (UIA) with more text patterns and document content so that Narrator can use it to read the outputs from applications.

We focused the above improvements specifically to address two key scenarios:

1. **Installing, setting up, and configuring your PC:** Using an existing Windows 7 PC, turn on Narrator by opening Ease of Access and selecting Narrator. Then go to the webpage that hosts the Windows 8 download and install point ([download Windows 8 Developer Preview here](#)), and walk through the setup with Narrator speaking to you. There are still a few bugs in the process that we are working on. But this now provides you with the ability to install using Narrator.

Narrator has some new configuration options in Windows 8. You can select a voice, change the speed at which it speaks, create customizable commands, and specify some other aspects of Narrator's behavior.



Narrator main screen to configure settings

Right out-of-the-box with a new Windows 8 tablet, you will be able to press the Windows logo key and Volume Up to launch Narrator and walk through the setup of your machine. Whether you're blind, have low vision, or are fully sighted, you'll be able to start experiencing a Windows 8 tablet from the moment you get it.

2. **Web browsing:** Previously Narrator didn't say much on webpages, and it was slow. But with the updates in Internet Explorer to leverage text patterns built into the UI Automation platform, and with additional performance updates, Narrator keeps up with you as you explore text on a webpage. Narrator provides you with the ability to continuously read a page (Use the Windows logo key + Alt + \ to invoke the reading) and then responds quickly to commands such as Ctrl, which will instantly stop Narrator from speaking. This allows you to interact with a control like a hyperlink (Windows logo key + Alt + Enter tells Narrator to select the hyperlink, and Windows logo key + Alt + Space navigates to the linked page).

In addition to addressing user feedback, a significant amount of work went into making sure that Metro style apps could also be accessible.

Evolving the accessibility platform for developers

Making Windows accessible while features are being built is challenging, and doing this while introducing a whole new development platform is even more difficult. However, we wanted users with disabilities to enjoy Metro style experiences right away (compare this to the Win32 platform, which took many years and multiple releases to become accessible).

As a start, we updated our accessibility foundation with support for industry standards. By supporting standards from the Web Accessibility Initiative, Accessible Rich Internet Applications (ARIA), HTML5, and XAML, it is easier for developers to code accessibility into their applications and for the ATs using UI Automation to consume the information that makes accessibility scenarios work on Windows 8.

This is in contrast to previous releases, where AT vendors used different "creative" ways of getting information from the system, in order to manipulate it and present it to their users. While a variety of approaches can provide rich experiences for users, it also creates a problem when non-standard approaches have to change in a new release. This is why we needed to create a strong foundation within the platform that leverages the existing coding standards (to which developers should adhere), and that can also be consistent from release to release. AT developers who use the platform can then reliably get accessibility information and don't have to do any special tricks or coding.

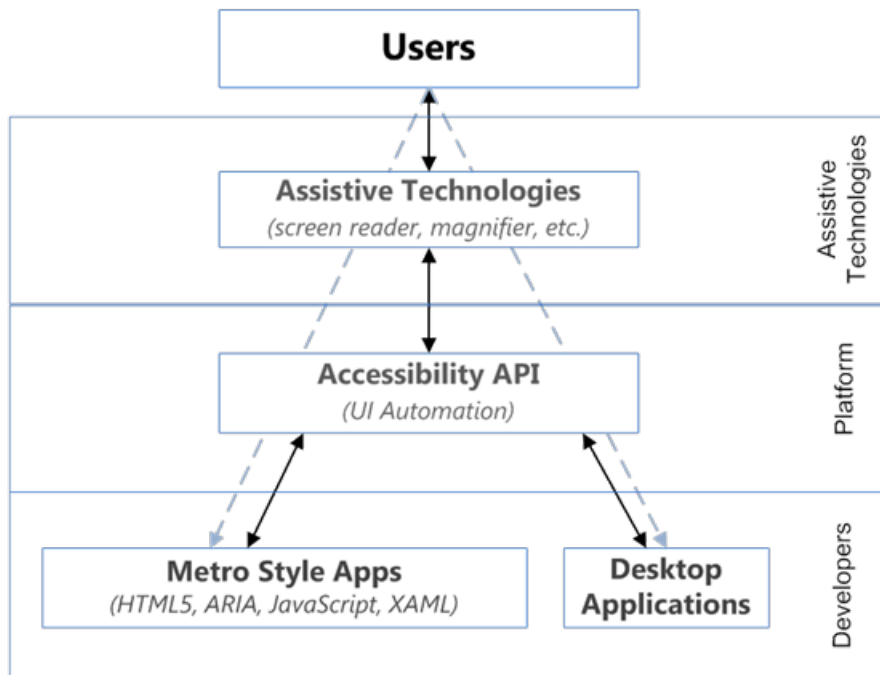


Diagram of developer, platform, and AT required to provide information to the user

With a consistent platform, developers of Metro style features within Windows can now leverage the standards and platform to ensure their components are accessible. While the features are designed, developed, and tested, we continually track the progress made towards accessibility. When we released the build for the Windows 8 Developer Preview, the team had been working on accessibility. However, we still had bugs that impacted High Contrast, keyboard navigation, and programmatic data for the ATs to consume. We are by no means done, and we continue to drive the accessibility requirements across the team to ensure we meet our goals. In each public release of Windows 8, you will see improvements being made in this area.

The Metro style UI is a new experience for Windows, and gives us an opportunity to present accessibility settings in a new way. This opportunity allowed us to simplify and optimize the key settings that people with disabilities depend on to manage their experience.

For example, we have a new way to toggle settings for high contrast, which is easier to discover and simpler to apply. We also made it easier to adjust the size of UI elements to be bigger, and take care of the DPI scaling settings for you, so you don't have to manage it manually. We think simplifying these settings will help a large set of users.

Developers creating and selling accessible apps

With Metro style apps, developers have an incredible opportunity to improve the accessibility ecosystem by creating and selling apps that meet a [baseline of accessibility](#).

Fortunately, developers don't need to learn new technologies to make their apps accessible. We rely on existing standards to reduce the learning curve for building accessible apps. HTML apps rely on the public HTML5 standard, which includes [ARIA](#) (a markup schema designed for declaring accessibility information). Likewise with XAML apps, we use the well-known markup schema used by similar platforms like Silverlight and Windows Presentation Framework (WPF). Additionally, the dev platform and tools shipped for Windows 8 support making an accessible app through every step of the development process:

- **Creating:** When creating a project using one of the [project templates](#) from Visual Studio Express, the code is accessibility-ready. This means that you can immediately use it with a screen reader (Narrator), it is fully usable with a keyboard, it works well in High Contrast mode, and it is visually accessible for text contrast and color. This gives the developer a great starting point towards building an accessible app.
- **Coding:** During coding of an app, there is additional support offered by the platform and tools:
 - Use Visual Studio Express IntelliSense to type [accessibility attributes](#) quickly and declare accessibility information in the markup.
 - Accessibility support is built into the Windows 8 controls. In most cases, all you need to do is define a good [accessible name](#).
 - Use the Dev Center [guidelines](#) and [samples](#) to learn best practices and copy/paste accessible code.

At this point you are probably thinking: how can these efforts possibly work for interactive games or HTML5 Canvas based apps? You're right; there are still classes of apps in which implementing accessibility will be more challenging than just leveraging the tools and templates. To help address these cases, we will continue to work with the developer community, post custom solutions, and expand accessibility guidelines with more examples.

- **Testing:** When your app is ready for testing, use the Windows SDK accessibility testing tools to validate the markup. The Dev Center documentation also offers guidelines about [testing a Metro style app for accessibility](#).
- **Selling:** Once the app is complete, if it meets the baseline accessibility scenarios, you can declare it as accessible during the Windows Store publishing process by selecting the Accessibility check box. This will allow users looking for accessible apps to easily find them in the Store.

When developers build an application for Windows 8, they should follow this process and ensure their apps do the following to reach the accessibility community:

- **Support the standards.** Ensure people with low vision or those who are fully blind can use a screen reader such as Narrator to accomplish the main scenarios offered by the app. The screen readers will leverage UIA and the standards discussed above to get information from the apps.
- **Make keyboard shortcuts.** Ensure people with mobility impairments or users of screen readers that prefer keyboard navigation can use a keyboard to interact with the app and its UI elements. This includes navigating with the Tab and arrow keys; activations with Spacebar and Enter keys; and the use of shortcuts (access keys and accelerators).
- **Support high contrast and "make it bigger."** Ensure people with moderate visual impairments can distinguish the UI and text with sufficient text contrast ratios, and a good high contrast mode; and respect layout settings when the "Make everything on your screen bigger" mode is active.

For more information, check out this //build presentation on [creating accessible Metro style apps](#), and get started creating your own app.

Discovering accessible Windows 8 apps

Users will be able to set an accessibility filter in the Windows Store that will allow them to discover the apps that have been declared accessible by the developer. Additionally, users will be able to provide comments and ratings to help each other find the apps that are most accessible, and to help the developer understand how well they did in making their apps accessible

Adapting accessibility features for new form factors

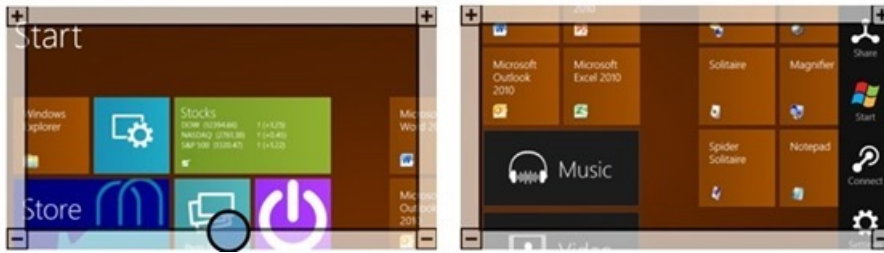
One of the most exciting changes in Windows 8 is the introduction of touch-only devices into the Windows family. And, as with all form factors that Windows supports, we want these new touch-only devices to be accessible. As a result, we spent a considerable amount of time planning what it would take to make our Windows ATs useful on touch-only devices, mainly through the adaptation of the Magnifier and Narrator features.

Magnify your screen and navigate using touch

Magnifier can be used in different ways, but one of the most popular ways to use it is with keyboard shortcuts (Windows logo key + and Windows logo key -). However, on a touch-only device, you don't have the keyboard available to input shortcuts, so we had to figure out how to make Magnifier work well in this scenario. We wanted to create a touch-based solution that was simple, fast, and unobtrusive. If you've used Magnifier before, you may have experimented with different modes in Windows 7. We chose to focus on full-screen mode for touch because of the data we gathered through the Customer Experience Improvement Program, which showed full-screen mode was the most commonly used. It's also the best mode to leverage touch gestures because it spans the whole screen.

One of the great benefits of using touch is that you can directly interact with everything on your screen. There's no need for separate devices like a mouse and keyboard – just touch exactly what you want. The downside we've heard from users who rely on magnification is that it can be hard to see *and* touch simultaneously because your hand is on the screen and it blocks you from seeing what's behind it. But the entire goal of Magnifier is to *help* users see the screen – not to hinder. Therefore, one of our design principles for touch-enabled devices was to make sure that you can control Magnifier entirely from the edges of the screen.

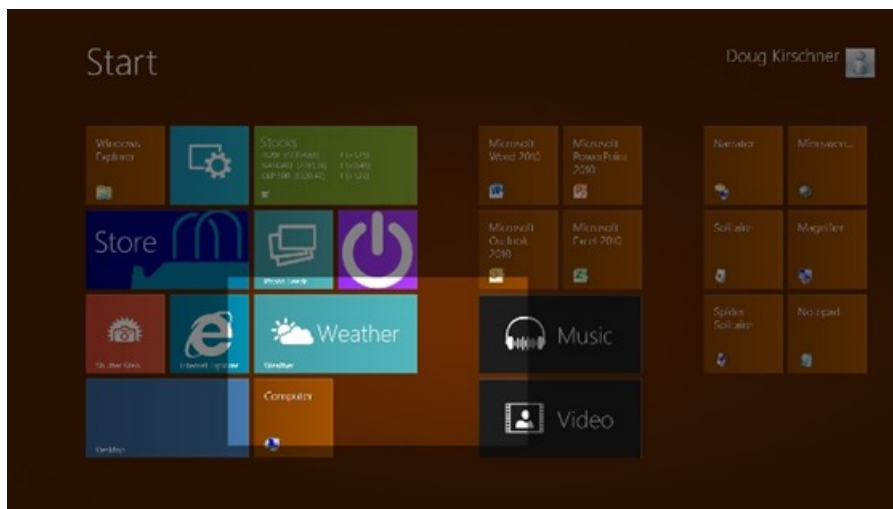
When you start Magnifier on a touch-enabled device (in the Ease of Access panel, set Magnifier to start when you press the Windows logo key + Volume up), you will immediately notice a border that appears around the edges of the screen. We know you will need to access all areas of the screen, so we made it easy to move the Magnifier around the screen using these borders. Simply drag your finger along the border to move Magnifier in that direction. When the border disappears, you are at the edge of the screen.



Drag your finger along the border to move around the screen. Borders disappear when you reach the edge of the screen.

The plus (+) and minus (-) buttons in the corners allow you to zoom in or out. We also built in support for multi-touch zoom using these same borders. Moving two fingers closer together or farther apart on the border allows you to quickly change the zoom level.

When you are zoomed in, sometimes it's confusing to know where you are on the screen. To remedy this, Magnifier has a preview feature that shows you exactly where you are in the context of the entire screen. Activate this by tapping with a thumb or finger on opposite borders at the same time. The preview will zoom out to show you exactly what part of the screen you're on, then it will zoom back in to your current location.



Tap on two opposing borders at the same time. Full screen preview highlights where you are on the screen.

You can even drag the highlighted region while it's zoomed out to move the Magnifier around the screen.

Most importantly with Magnifier, you don't need to change the way you interact with your device to use it with touch. Once it's turned on, it will work with all of your apps. For users with low vision who have trouble seeing their devices, Magnifier makes it easy to see the screen and touch it, too.

Explore and learn the UI with Narrator

In Windows 8, Narrator has been redesigned to be substantially faster and support many new features. To support Narrator on touch-only devices we've implemented a standard way to launch Narrator, by holding down the Windows logo key and pressing the Volume Up button. Once Narrator is running, you can use Narrator's built-in touch commands to explore the screen and control your device.

If you're blind, then the challenge with touch is that there's no way to find something on the screen without activating it. On a Windows 8 device, Narrator addresses this challenge by allowing you to drag a single finger around the screen. Narrator will read what is under your finger but won't activate it. Users with vision will notice that the Narrator cursor will follow your finger as well. We refer to this as "exploring." A good way to understand this is to imagine there is a sheet of glass on top of your screen – Narrator will allow you explore what is underneath by touching the glass but without touching the screen directly. Once you've found the item you're looking for by exploring with a single finger, you can activate it by tapping anywhere with a second finger.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

These are just two examples of ATs that are shipping with Windows 8 and that are now optimized for touch-only devices. There are many other improvements across all the Windows 8 ATs, but we will save that to discuss at a later time.

Onboarding assistive technology vendors

There are many scenarios and a wide range of impairments to cover, and so we've engaged and partnered with AT vendors to ensure we are creating the best and most comprehensive experiences for the disability community. The assistive technologies that ship in Windows 8 will work across both the desktop and Metro style UI experiences, to provide seamless access to the PC. People who need advanced AT features may need or want to purchase solutions from specialty Assistive Technology vendors (AT vendors) to meet their specific needs.

AT vendors create sophisticated ATs that can provide richer experiences to the disability community. For example, they may provide in-depth support for specific applications and for legacy applications. The ATs shipped in Windows may not work well with apps that do not support industry standards or platform technologies, including for example, legacy applications that do not implement UIA.

In Windows 8, we invested heavily in building the foundation for the new Metro style UI and adopting the industry standards that will benefit application developers, ATs, and the disability community.

By providing a standardized way of getting the information, ATs can work with the standards that app developers are used to, but more importantly, AT vendors can rely on these standards to be supported through multiple Windows releases, to ensure their ATs don't break with each release. Since the //Build conference, we have partnered with leading AT vendors to help them get started with Windows 8. This has included support for previously used [mirror drivers](#) and UIA support.

We continue to sync up with the AT vendors to ensure that their questions are addressed, and we are working toward the common goal of an accessible Windows 8.

Windows 8 has been an incredible opportunity for us to improve our accessibility support. Not only have we evolved the platform, we have introduced new opportunities for developers to broaden their application's reach into the disability community. We have also focused a lot of attention on the ATs that are included with Windows 8, not only improving performance and language support, but also enabling new form factors including touch-only devices. We continue to be very committed to a rich and innovative third-party ecosystem, and with more standardized and consistent interfaces, we hope to help the ecosystem continue to innovate on Windows.

If you are a user with accessibility needs, we think you will like what we have done. If you are a developer, build an accessible app and reach a larger spectrum of users! If you are an AT vendor, come work with us and refresh your applications using our platform. This is an exciting and compelling release that will change how people of all abilities interact with PCs.

There is still work to be done in Windows to meet all the accessibility needs, but we would like to encourage people to try out the Metro style experiences with our free, updated Windows 8 ATs.

-- Jennifer Norberg, Lead PM, Human Interaction Platform team

Data

1. WHO: [Disability and health: Fact sheet Number 352](#)
2. US Census: [Profile America Facts for Features](#)
3. European AT Report: [Analysing and federating the European Assistive Technology ICT industry, March 2009 \(PDF\)](#)
4. [Lifekluger: The Touch Barrier – Accessibility and usability issues around touch technologies](#)
5. [Occupational Outlook Handbook, 2010-11 Edition](#)

Internet Explorer Performance Lab: reliably measuring browser performance

Steven Sinofsky | [2012-02-16T09:00:00+00:00](#)

A big part of this blog is going behind the scenes to show you all the work that goes into the engineering of Windows 8. In this post we take a look at something we all care very deeply about--as engineers and as end-users--real world web performance. We do a huge amount of work to get beyond the basics of anecdotes and feel as we work to build high performance web browsing. This post is authored by Matt Kotsenas, Jatinder Mann, and Jason Weber on the IE team, though performance is something that every single member of the team works on. --Steven

Web performance matters to everyone, and one [engineering objective](#) for Internet Explorer is to be the world's fastest browser. To achieve this goal we need to reliably measure browser performance against the real world scenarios that matter to our customers. Over the last five years we designed and built the [Internet Explorer Performance Lab](#), one of the world's most sophisticated web performance measurement systems.

The IE Performance Lab collects reliable, accurate, and actionable data to inform decisions throughout the development cycle. We measure the performance of Internet Explorer 200 times daily, collecting over 5.7 million measurements and 480GB of runtime data each day. We understand the impact of every change to the product and ensure that Internet Explorer only gets faster. This blog post takes a deep look at how the IE Performance Lab is designed and how we use the lab to ensure we're continually making the web faster.

In this post, we present:

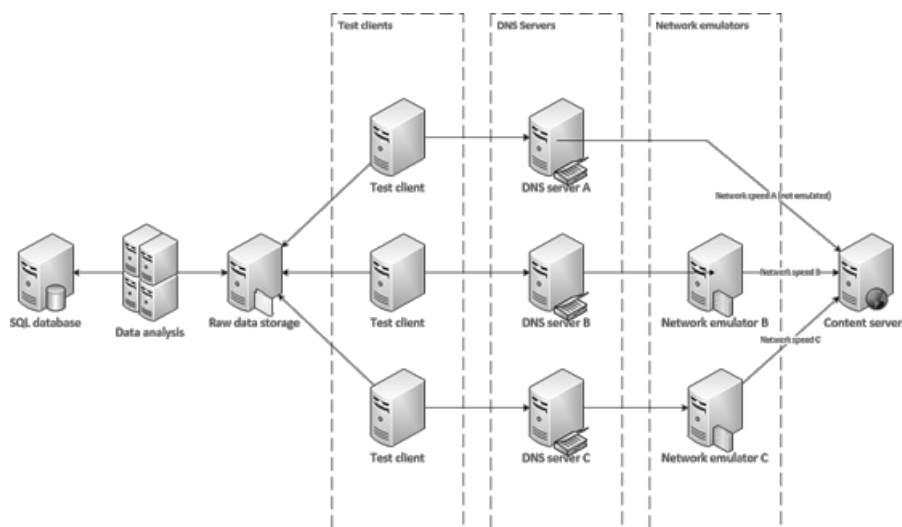
- Overview of the IE Performance Lab
- Lab infrastructure
- What (and how) we measure
- Testing a scenario
- Results investigation
- Testing third-party software
- Building a fast browser for users

Overview of the IE Performance Lab

In order to reliably measure web performance over time, a system needs to be able to reproducibly simulate real world user scenarios. In essence, our system needs to create a "mini version of the Internet."

The IE Performance Lab is a private network completely sealed from both the public Internet and the Microsoft intranet network, and contains over 140 machines. The lab contains the key pieces of the real Internet, including web servers, DNS servers, routers, and network emulators, which simulate different customer connectivity scenarios.

Although this may appear complex at first glance, this approach allows all sources of variance to be removed. By controlling every aspect of the network, down to individual packet [hops](#) and latencies, our tests become deterministic and repeatable, which is critical to making the results actionable. In the IE Performance Lab, activity is measured with 100 nanosecond resolution.



This type of network configuration also provides a great amount of flexibility. Because we're simulating a real world setup, our lab can accommodate nearly any type of test machine or website content. The IE Performance Lab supports desktops, laptops, netbooks, and tablets with x86, x64, and ARM processors, all simultaneously. Similarly, because the lab uses the [Windows Performance Tools \(WPT\)](#), we can run the same tests using different web browsers, toolbars, anti-virus products, or other third-party software and directly compare the results.

WPT provides deep insight into the underlying hardware. Using WPT, we can capture everything from high-level CPU and GPU activity, to low-level information such as cache efficiency, networking statistics, memory usage patterns, and more. WPT allows us to measure and optimize performance across the stack to ensure that the hardware, device drivers, Windows operating system, and Internet Explorer are all efficiently optimized together.

A single test run takes 6 hours to complete and generates over 22GB of data during that time. This highly automated system is staffed by a small team that monitors operations, analyzes results, and develops new infrastructure features.

Lab infrastructure

The Performance Lab infrastructure can be broken into three main categories: Network and Server, Test Clients, and Analysis and Reporting. Each category is designed to [minimize interaction across components](#), both to improve scalability of testing and to reduce the possibility of introducing noise into the lab environment.



Here's a view of the IE Performance Lab, including a number of test and analysis machines on our private network.

Network and server infrastructure

Let's start by discussing the DNS servers, network emulators, and content servers; all the components that together create the mini Internet. Over the next three sections we'll work our way from right to left in the architectural diagram.

Content servers

Content servers are web servers that stand in for the millions of web hosts on the Internet. Each content server hosts real world web pages that have been captured locally. The captured pages go through a process we refer to as sanitization, where we tweak portions of the web content to ensure reproducible determinism. For example, JavaScript Date functions or Math.Random() calls will be replaced with a static value. Additionally, the dynamic URLs created by ad frameworks are locked to the URL that was first used by the framework.

After sanitization, content is served similarly to static content through an [ISAPI](#) filter that maps a hash of the URL to the content, allowing instantaneous lookup. Each [web server](#) is a 16-core machine with 16GB of RAM to minimize variability and ensure that content is in memory (no disk access required).

Content servers can also host dynamic web apps like Outlook Web Access or Office Web Apps. In these cases, the application server and any multi-tier dependencies are hosted on dedicated servers in the lab, just like real world environments.

Network emulators

Since many sources of variability have been removed, network speeds no longer reflect the experiences of many users with slower connections. To simulate real world customer environments, a test can take advantage of network emulation to understand the performance across the wide range of networks in use today. The lab supports emulating several DSL configurations, cable modems, 56k modems, as well as high-bandwidth, high-latency environments like WAN and 4G environments. As HTTP requests are passed to the emulator, it simulates network characteristics like packet delay and reordering, then forwards the request on to the web hosts. Upon receiving a response, emulation is again applied and then passed back to the test client.

Using dedicated hardware for network emulation provides the most realistic testing environment possible, and significantly reduces the observer effect. Although dedicated hardware adds cost and complexity compared to proxy or software-based solutions, it's the only way to accurately measure performance. Browsers limit the number of simultaneous proxy connections to prevent proxy saturation, so using a proxy for network emulation has the unintended effect of sidestepping domain [sharding](#) and other optimizations made by the webpage. Additionally, local network emulation will compete with the browser for local machine resources, especially on low power machines.

DNS servers

Like real world DNS servers, the lab's DNS servers link the content servers to the test clients. The lab also uses a different DNS server for each network emulator, meaning that changing from one network speed to another is as simple as changing the DNS server. In these cases, instead of resolving domain names to the web hosts, the DNS server resolves all domain names to the associated network emulator.

Test client configurations

We want to ensure that Internet Explorer consistently runs fast across all classes of computer hardware. The lab contains over 120 computers used to measure Internet Explorer performance. We refer to these as test clients; they range from high-end x64 desktops, to low-powered netbooks, to touch-first tablet devices, and everything in between. Because repeatability of measurements is paramount, all test clients are physical machines.



Internet Explorer Performance Lab change comparison machine pool

Different machine classes contain both discrete and integrated graphics platforms to ensure that Internet Explorer continues to take full advantage of hardware acceleration across the ecosystem of devices. Above is our main machine pool. These PCs approximate the average consumer experience over the lifetime of a Windows 7 or Windows 8 PC. Machines are ordered from the OEM to be identical; they all come from the same manufacturing lot and their performance characteristics are verified prior to use.

Since the lab runs 24/7, hardware failures are inevitable. Replacing failed components with identical parts from a different manufacturing lot almost always results in the repaired computer running *faster* than the other machines in the pool. While this difference would be unnoticeable in the real world, when you're measuring down to 100 nanoseconds, even a few cycles can impact the results! If after a repair a machine no longer runs identically to the rest of the pool, it is removed from the lab and the pool's size permanently shrinks. In response, the lab's purchases include extra "buffer" machines, so that when a failed machine is removed from the pool, the excess capacity provides a cushion, and the lab's operations are not affected.

Pool Name	# Machines	Form Factor	Processor	Arch	Clock Speed	RAM	Graphics
Main Pool	32	Desktop	Core i5 750 (Lynnfield)	64-bit	2.66GHz	4096MB DDR3	NVIDIA GeForce 310

To add hardware breadth, we have additional machine pools that run the spectrum of consumer scenarios. Good performance on these machines ensures that IE uses the underlying hardware effectively across the PC ecosystem.

Pool Name	# Machines	Form Factor	Processor	Arch	Clock Speed	RAM	Graphics
High?end 1	20	Desktop	Core i7 870	64?bit	2.93GHz	4096MB DDR3	ATI Radeon HD 4550
High?end 2	4	Desktop	Xeon 5150 (Woodcrest)	64?bit	2.66GHz	8192MB DDR2	ATI Radeon X1950 Pro
Mid?range 1	6	Desktop	Core 2 Duo (Wolfdale)	64?bit	3.0GHz	4096MB DDR2	Intel GMA 4500
Mid?range 2	15	Desktop	Core 2 Duo E6750	64?bit	2.66GHz	4096MB DDR2	ATI Radeon HD 2400 XT
Mid?range 3	25	Desktop	Core i5 2500	64?bit	3.30GHz	4096MB DDR3	Intel HD Graphics 2000
Mid?range 4	6	Desktop	Core 2 Duo (Conroe)	64?bit	2.66GHz	4096MB DDR2	ATI Radeon HD 2400XT
Mid?range 5	4	Desktop	Core 2 Duo (Conroe)	64?bit	2.4GHz	4096MB DDR2	ATI Radeon X1950 Pro
Low?power 1	2	Laptop	Atom Z530	32?bit	1.6GHz	2038MB DDR2	Intel GMA 500
Low?power 2	4	Netbook	Atom N270	32?bit	1.6GHz	1024MB DDR2	NVIDIA ION
Low?power 3	2	Netbook	Atom N450	64?bit	1.66GHz	1024MB DDR2	Intel GMA 3150
Low?power 4	4	Netbook	Atom N270	32?bit	1.6GHz	1024MB DDR2	Intel GMA950
Low?power 5	4	Slate	ARM	32?bit	—	—	—
Prototype hardware	—	—	—	—	—	—	—



Low-powered test machines. Each one is in a different state of testing.

If even more diversity is needed, the IE Performance Lab can also make use of the Windows Graphics Lab. The Windows Graphics Lab stocks nearly every graphics chipset manufactured. PCs can be configured into nearly any permutation imaginable and then used for performance testing. The Windows Graphics Lab is invaluable for diagnosing graphics problems across chipsets and driver revisions.

Analysis and reporting servers

Collection and analysis of test results are divided into two separate steps. By offloading analysis to dedicated machines, the test clients can begin another performance run earlier, and more powerful server class machines can be used to perform the analysis more rapidly. The sooner the analysis completes, the more efficiently we can identify performance changes.

For analysis, we use 11 [server class machines](#), each of which has 16 cores and 16GB of RAM. During analysis, each trace file is inspected and thousands of metrics are extracted and inserted into a SQL server. Over the course of 24 hours these analysis machines will inspect over 15,000 traces that will be used for trend analysis.



Pictured are two of several server racks which contain file servers, a SQL server, and several analysis and content servers.

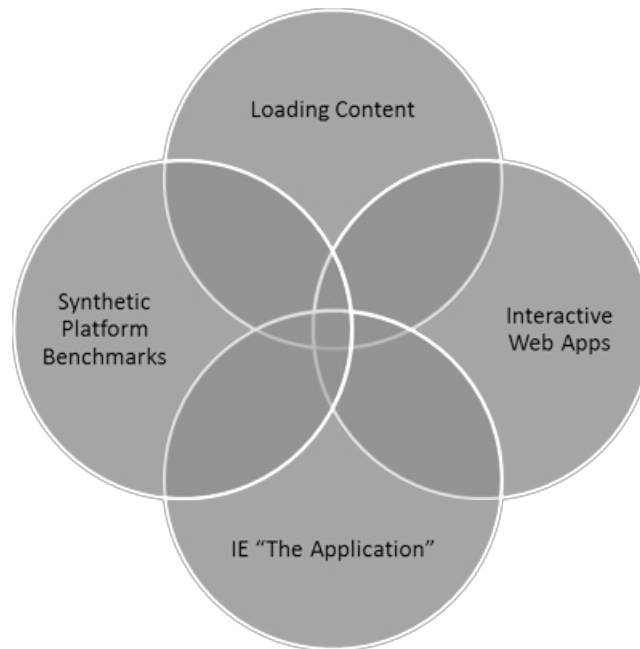
The [SQL Server](#) used to store the nearly 6 million measurements we collect each day is a 24 logical core machine with 64GB of RAM. Reports can be generated directly from SQL, or results can be inspected using either an HTML-based comparison application or [WCF](#) service that provides results in XML or JSON formats.

What (and how) we measure

With the infrastructure in place, let's review the different types of scenarios measured in the Performance Lab, and the tools we use to gather metrics.

Scenarios measured daily

The Performance Lab focuses on real world scenarios that matter to users. As a result, we run over 20,000 different tests daily. These tests fall into four, sometimes overlapping, categories:



Loading content – Navigating from one page to another is still the most common activity inside a web browser. Loading web content is also the only category that touches most of the browser's [eleven subsystems](#). Loading web content is a prerequisite for the other categories of scenarios.

Interactive web apps – This category covers what is sometimes referred to as content creation, AJAX applications, or Web 2.0 sites. It includes interacting with popular news and social networking sites as well as interacting with mail and document applications like Outlook Web Access and Office Web Apps.

IE "the application" – Important but often forgotten are scenarios that interact with the browser itself. Common interactions include opening or closing the browser, switching tabs, using browser features like History and Favorites, and panning and zooming with both keyboard and mouse, and touch inputs.

Synthetic benchmarks – Rarely forgotten but often overstated are synthetic benchmarks like WebKit SunSpider. Benchmarks can be a useful engineering tool as they are designed to stress individual browser subsystems and accentuate differences between browsers. However, in order to maximize those differences, benchmarks often resort to atypical usage patterns or edge cases.

Real world patterns

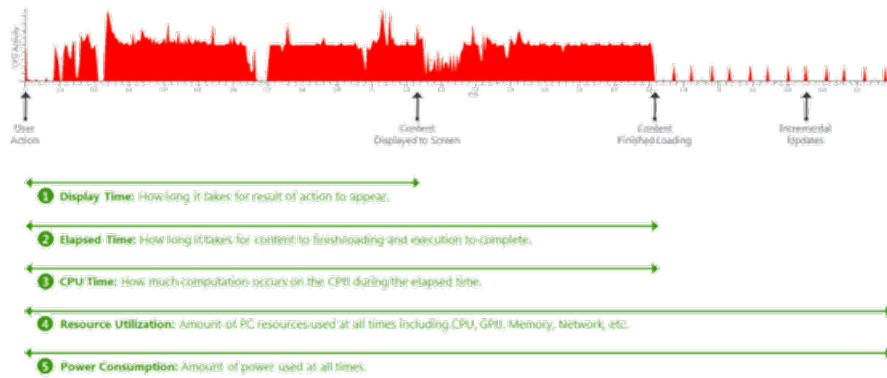
When measuring performance it is important to ensure that the tests reflect real world usage patterns. Most Software Engineering textbooks refer to this process as [workload modeling](#), or *application usage modeling*. To ensure that the Performance Lab measures real world patterns, the Performance Lab uses real Web pages that represent real world patterns and exercise different browser subsystems.

In order to determine which sites to use for testing, we regularly crawl millions of sites and compile a list of site attributes and coding patterns. We use 68 different data points to determine commonalities across sites – things like the depth and width of the resulting DOM, CSS layout patterns, common frameworks used, [international features](#), and more. From the results we chose sites that best represent the common patterns and diversity of the broader Web.

Engineering metrics

Performance is a multi-dimensional problem. The only way to get an accurate view of performance is to understand the scenario you're testing,

and how the hardware and OS interact with the browser. Here's a closer look at five important performance metrics in the context of loading a major sports site for the first time.



Display Time - Display Time measures the time from when the user performs an action until the user sees the result of that action on the screen.

Elapsed Time - Most sites continue to perform background work after content has been displayed to the screen. Examples might include downloading the next email in a web mail application or sending analytics back to a provider. From the user's perspective, the site might appear finished; however, significant work is often occurring which can impact overall responsiveness.

CPU Time - Modern web browsers are almost exclusively limited by the speed of the CPU. Offloading work to the GPU and making the CPU more efficient makes a large impact on performance.

Resource Utilization - Building a fast browser means ensuring resources across the entire PC work well together, including network utilization, memory usage patterns, GPU processing, graphics, memory, and hundreds of other dimensions. Since users run several applications at the same time on their PCs, it's important for browsers to responsibly share these resources with other applications.

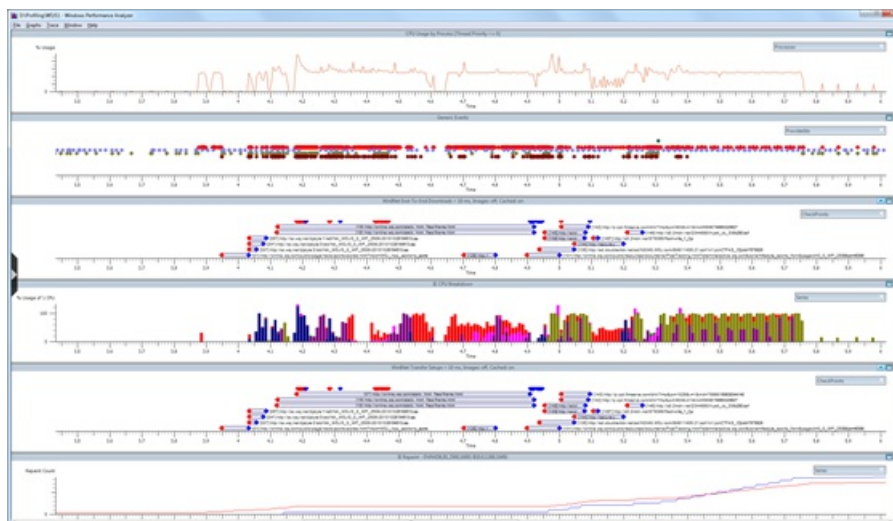
Power Consumption - Increasing power efficiency leads to longer the battery life in mobile scenarios, lower electricity costs for the device, and a smaller environmental impact.

Concentrating only on a single metric creates an overly simplistic view of performance. By focusing on a single metric, humans naturally tend to optimize for that metric, often at the expense of other equally important metrics. The only way to combat that tendency is to measure all aspects of performance, and then make the tradeoffs consciously, rather than implicitly.

In total, the Performance Lab measures over 850 different metrics. Each one provides part of the picture of browser performance. To give a feel for what we measure, here's a (non-exhaustive) list of key metrics: private working set, total working set, HTTP request count, TCP bytes received, number of binaries loaded, number of context switches, DWM video memory usage, percent GPU utilization, number of paints, CPU time in JavaScript garbage collection, CPU time in JavaScript parsing, average DWM update interval, peak total working set, number of heap allocations, size of heap allocations, number of outstanding heap allocations, size of outstanding heap allocations, CPU time in layout subsystem, CPU time in formatting subsystem, CPU time in rendering subsystem, CPU time in HTML parser subsystem, idle CPU time, number of threads.

Windows event tracing infrastructure

Metrics are gathered using [Windows Event Tracing Infrastructure \(ETW\)](#) and [VMMap](#). ETW is the Windows-wide event logging system that is used by many Windows components and third-party applications, including the [Windows Event Log](#). ETW logging APIs are extremely low level and low overhead, which is critical for performance testing.



The trace viewer included in WPT, `xperfview.exe`, is a powerful visualizer that allows correlation and overlaying kernel, CPU, GPU, I/O, networking, and other events. WPT also supports [stack walking](#). Stack walking takes a snapshot of the program's callstack at regular intervals and saves the stack as part of the trace. By correlating ETW events with stacks, WPT will display not only what work was being done, but the callstack associated with that work and the amount of time spent doing that work, with 10 microsecond resolution. Stack walking can be enabled on any process, even one that does not use ETW events. The drawback to stack walking is that it requires [debugging symbols](#) to decode the stacks, and is susceptible to [aliasing](#).

Testing a scenario

The final piece of the puzzle is the actual test process. Testing can be broken into 3 phases: setup, testing, and errors and cleanup. Here's a flowchart of the entire process to follow along.



Setup

The process starts when a user requests a run through the lab website or automation framework. The run is placed into a priority queue with other pending runs. When a test client becomes available, it checks the queue and starts the highest priority job that it can. First, the test client installs the Test OS specified. The IE Performance Lab supports testing on Vista, Windows 7, and Windows 8. The test client installs a fresh copy of the Test OS for every run so the machine always starts in a known good state.

Once the Test OS is installed, the client configures WPT, VMMap, and the test harness. The run also specifies a number of IE settings such as the homepage, use of Suggested Sites, InPrivate browsing, and others. Any third-party software is also installed and configured at this point.

The final step before testing is ensuring that the test client is idle to minimize test interference. Windows defines a concept of [idle tasks](#). Idle tasks are a way for Windows and other developers to schedule non-critical work to happen at a later time when the user is not competing for resources. OS idle tasks include [prefetching or SuperFetch](#), disk defragmentation, updating search indexes, and others, depending on OS version and configured services. To ensure that no idle work is done during the tests, the idle task queue is flushed. Additionally, Windows Defender is paused and the log location for the test harness is marked as excluded from the Windows Indexing Service to prevent log and trace files from causing the indexer to start during a test run. Testing is done in multiple passes to minimize the number of providers needed, since additional providers increase the [observer effect](#). The first pass is always a warm-up pass. Warm-up ensures that the browser binaries are “warm” and that the maximum amount of cachable page content is available in the WinINET cache. Subsequent passes each focus on a specific type of instrumentation, like stack walking, memory tracing, and I/O and registry tracing.

Errors and cleanup

If at any time during the test the browser crashes, the test pass is considered failed and the run moves on to the next test pass. If at any time during the tests Windows crashes, the computer reboots and the OS is reinstalled, since its state cannot be guaranteed. If the number of retries exceeds the threshold, the whole run is considered failed and the machine moves on to another run to prevent endlessly trying to test an unstable build.

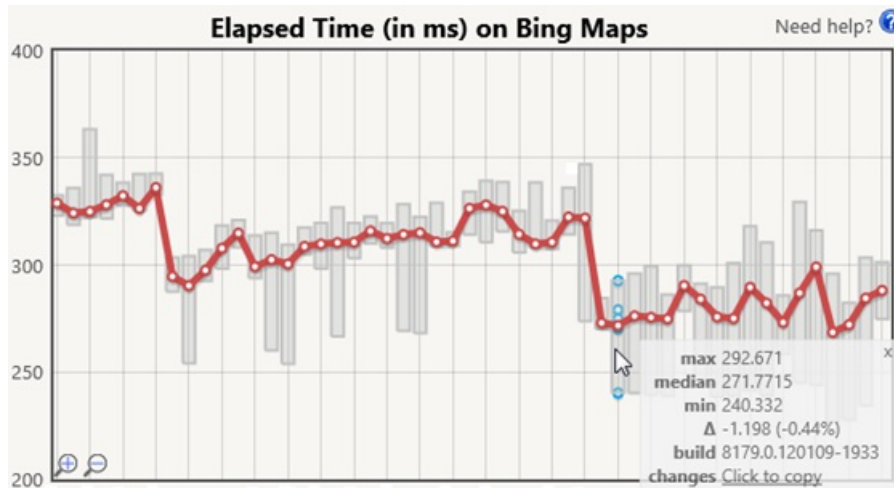
When all the test cases are complete, the test client uploads the logs and traces for analysis. The test client then returns to an idle state and begins polling for a new run.

Results investigation

Each metric is tracked change-over-change. We run each test case a minimum of ten times, and duplicate runs on at least two different machines to

create the sample population. Using statistical tools, uncharacteristic results can be automatically flagged for investigation. A variance change is also considered a regression. Users interact with IE under a wide range of circumstances and on a wide range of hardware, and one of our goals is to ensure a smooth and predictable experience every time.

In addition to automated analysis, a triage team investigates the daily results to watch for trends and other interesting behaviors. Manual investigation cannot be eliminated because many statistical tools assume both a [normal distribution](#) and that all samples are independent. Neither assumption may be strictly true for our measurements. Some activities in IE are driven by a timer from the OS, meaning results are also dependent on when (along the timer's cycle) the page load begins. A page load that starts right before or after a timer interrupt may do more or less work because IE must service the interrupt at different points in the page-load process. This interruption can have a rippling affect that leads to a [bimodal distribution](#). Also, because we use repeated trials (and we don't wipe the machine between iterations) the next trial is influenced by previous trials. Here's a sample Elapsed Time graph for Bing Maps for change-over-change comparison.



The red series shows the median value of each test run, and grey bars show the range. Hovering over a test run will show the iterations for the metric (in blue) as well as a tooltip that provides the exact values for minimum, median, max values, as well as the absolute and relative difference with the previous test run. The tooltip shown in this image also provides additional context like the build being tested, and a quick link to our source control system to view the changes in the build.

The combination of automated analysis and manual investigation provides the IE team with reliable and actionable data for performance tuning.

Testing third-party software

Many third-party applications depend on Trident, the network stack, and other IE components. Extensions like [BHOs](#) and toolbars load within the IE context. Other applications, like security software, can inject themselves between IE components. These applications become part of the IE stack, and can lead to poor performance. The Performance Lab is capable of measuring the impact of third-party software on browsing real world content in a controlled environment. These studies are important to IE and the ecosystem because users generally cannot quantify the impact of popular software on their browsing experience.

When testing third-party software impact, we compare a run with third-party software installed, with a clean run with only IE installed, to determine the impact of the software. In particular, we are interested in measuring two metrics: startup time and navigation time. Startup time measures the time it takes to launch the browser and navigate to an URL, whereas navigation time measures the time it takes to navigate to an URL when the browser has already been launched. Startup will also include the time that third-party applications take to load their IE extensions.

Using cached content allows repeatability in our measurements. Further, by measuring a cached site, we can definitively know that a performance regression is caused by the third-party software and not by differences in the site. Whenever measuring the impact of third-party software, we also validate our findings by testing startup and navigation on a direct connection to the Internet, to verify that the testing environment is not responsible for any deltas.

Many third-party applications offload work during a page navigation to cloud services. While parallelization of work and use of cloud services are excellent techniques to improve performance, some applications wait synchronously for the results from the network, blocking the navigation in the process. There are many real world scenarios, like strict firewalls, WAN connections, and offline scenarios, where such patterns can lead to poor performance for users. Third-party software should never process synchronously in response to an IE or user action, and should batch UI and DOM updates to minimize disruption.

Building a fast browser for users

Real world browser performance matters. Measuring performance at scale is a significant investment and a full-time job, but the results are well worth the effort. The data gathered by the Internet Explorer Performance Lab is instrumental in our understanding of browser performance and of the underlying PC hardware, and in developing a fast, fluid, and responsive web experience for users.

—Matt Kotsenas, Jatinder Mann, and Jason Weber for the Internet Explorer Performance Team

Connecting your apps, files, PCs and devices to the cloud with SkyDrive and Windows 8

Steven Sinofsky | [2012-02-20T09:00:00+00:00](#)

Many folks reading this blog are active users of SkyDrive and Mesh, both part of the broad set of Windows Live services (like Hotmail), and the Windows Live Essentials programs (Messenger, Photo Gallery, Movie Maker, Mail, and more). With their introduction and with Windows 7, we have talked about how these services really complete the Windows experience. As we developed Windows 8, we thought deeply about how these services can take an even more active role in completing the experience—offering a cloud service for each and every Windows 8 customer and all their PCs (and phones), should you choose to use it. In previous posts ([Signing in to Windows 8 with a Windows Live ID](#) and [Extending "Windows 8" apps to the cloud with SkyDrive](#)) we kicked off talking about cloud services, how you can automatically roam your settings to multiple PCs, and how applications can take advantage of this roaming. In this post, we will show you three cool things that you will get by choosing to sign up and use SkyDrive with Windows 8.

*This post was written by **Mike Torres and Omar Shahine**, Group Program Managers for SkyDrive.*

--Steven

A few months ago we [published our vision](#) for designing personal cloud storage. While SkyDrive can store all types of files, the category of personal cloud storage is focused on the content that people create or capture themselves. Today we're going to provide an update on how we expect to deliver our vision for Windows 8 and simultaneously scale to meet the needs of billions of customers who will store hundreds of petabytes of data in our service.

We think what people want in personal cloud storage is a single drive that's available across all of their devices, tailored to the experiences they're using, providing instant, secure, and private access to their files, and sharing files and folders with people they choose. To bring this to billions of people, our approach is to seamlessly connect the files (and behaviors) that people have today on the PC with the app and device experiences that they will use in the future. Rather than using a patchwork of services, people can use one service to connect to their files – with no compromises. No copying files from one cloud to another just to share or collaborate. No [converting files](#) or having to switch to new apps. No searching across different storage areas to find files.

Delivering personal cloud storage for billions of people

Today we provide personal cloud storage for 17 million SkyDrive customers. These are active customers who use it every month to privately share photos and collaborate on Office documents. We currently store approximately 10 petabytes of user data (one petabyte is a million gigabytes, or a million billion bytes) and we expect to grow that beyond what some of the largest scale services on the Internet support today (for example, Hotmail stores over 100PB of user data, and since we share much of the same infrastructure, we have a lot of deep insights and knowledge about how to scale SkyDrive). These are important numbers for us. We aspire to be a service that people find useful and valuable on an ongoing basis, not just a service that people happen to register for when they get a new device.

Growing our infrastructure is one of many things we have been busy working on, but most important of these are our investments in sync and cloud scenarios for Windows 8, which will finally bring the DNA of SkyDrive and Mesh together into one service. Given our goal to be the world's hard drive, we will need to continue to build out the service, and you can expect our pace of improvements to continue through 2012.

This post will cover the ways that SkyDrive will evolve with Windows 8 from a website today into a true [device cloud](#) for Windows customers. We've organized the post around the three biggest things we're doing:

- SkyDrive Metro style app on Windows 8
- SkyDrive files integrated into Windows Explorer on the desktop, and
- The ability to fetch remote files through SkyDrive.com

We've also prepared a short video demonstrating these three concepts in more detail:

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

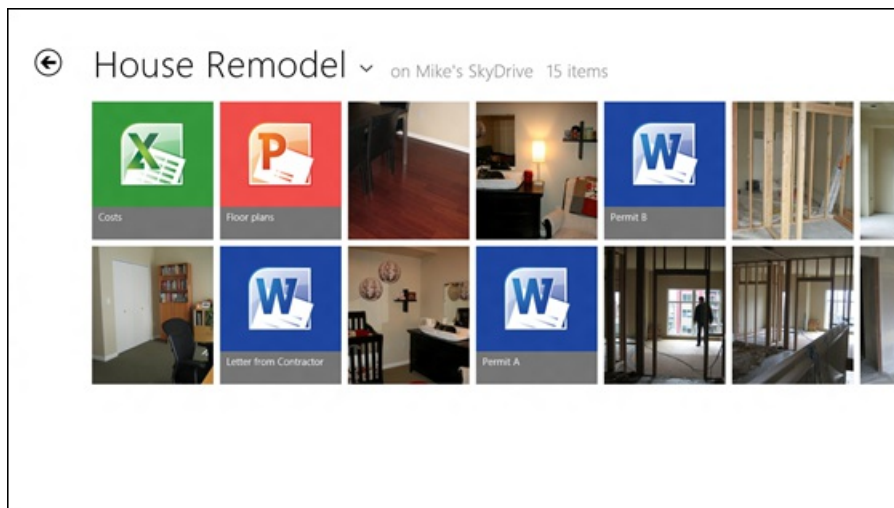
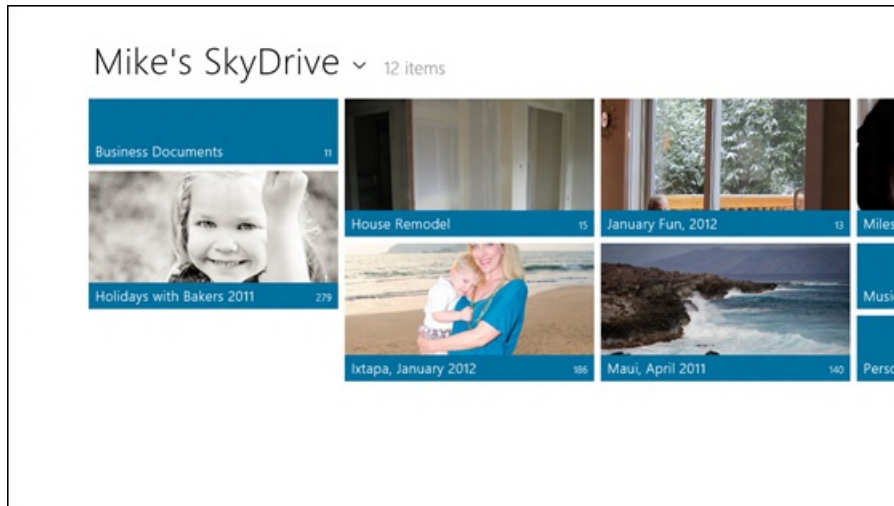
SkyDrive as a Metro style app for Windows 8

Inside Microsoft, we've all been using Windows 8 on multiple PCs for some time now. We clean install Windows 8 on a new PC and sign in, and all of our settings, browser history, and customizations just show up. In addition, one of the most important steps we take to make a new PC "ours" is to copy over our personal files – like documents and photos. With Windows 8, we wanted to make sure that your files would be instantly available and up-to-date as you move between PCs – without configuring add-ons or using a USB drive.

One approach to building a modern device cloud is to build a set of proprietary apps that hide files from users. But because we are building a cloud that can store different types of data and lots of it, we decided to take a different approach. We designed a no-compromise cloud experience where enthusiasts will be able to control files the way they want, while others who are less familiar with the file system can still take advantage of the cloud simply by accessing SkyDrive through the apps they use every day.

Enter the new SkyDrive app. With the SkyDrive app, an early version of which will be available at Consumer Preview, we focused on two things: 1) designing a fast, fluid, touch-first version of SkyDrive that makes it quick, easy, and even fun to browse and access your files, and 2) making your SkyDrive available for use from any Metro style app via the file picker (open/save) and the new Share charm in Windows 8.

To build a SkyDrive experience on WinRT, we took an approach that we expect many web developers will choose to take on Windows 8. We built the entire app using modern web technologies like JavaScript, CSS, and HTML5, and because of our recent updates to SkyDrive.com, we were able to use the same JSON APIs and JavaScript object model that the website uses. The only difference on Windows 8 is that we bind the results to modern controls that were built for touch. This is part of the reason it's so fast, and the touch behavior works so well (and works on Windows on ARM too). Over time, we fully expect the Metro style app and SkyDrive.com to "converge" on functionality so there won't be a question of which experience someone should use. When using Windows 8, the SkyDrive Metro style app will be the best way to browse and manage your SkyDrive.



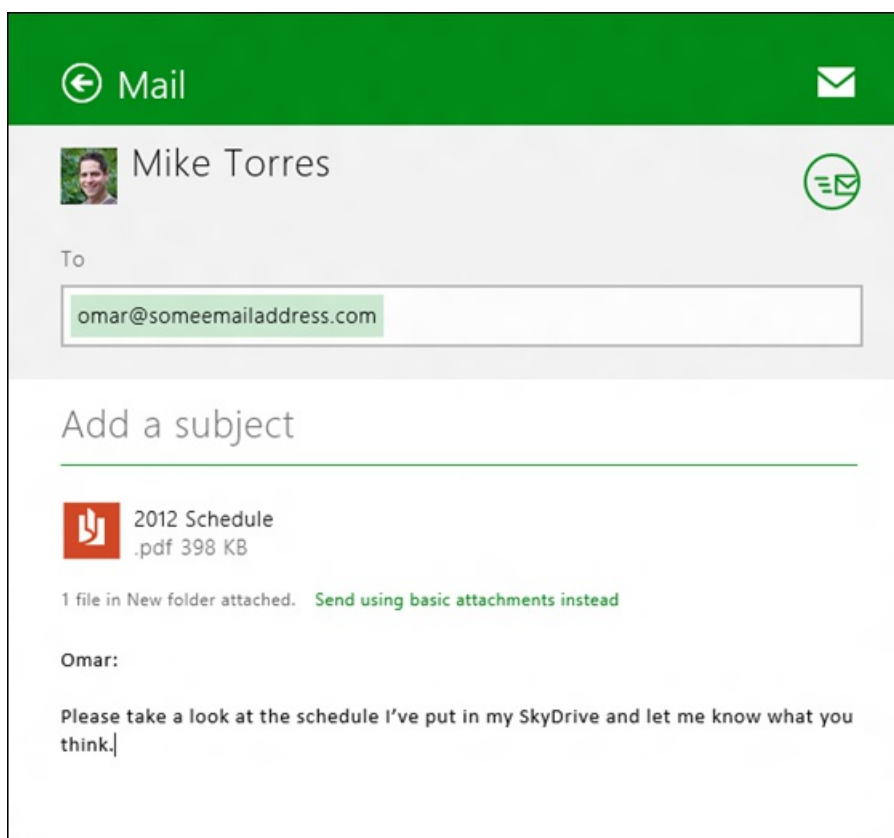
One of the most powerful features in Windows 8 is the ability to integrate SkyDrive functionality across other apps using charms and contracts. SkyDrive is available as a file picker, so from any app you'll be able to open files from, and save files to your SkyDrive. If the app developer chooses, you'll even be able to save files automatically back to SkyDrive. Alternatively, if you're using a content creation app like a document editor, you'll be able to save files back to SkyDrive in any folder that you choose.

Together, this will bring a [file cloud](#) to every Metro style app, allowing you to open files in your SkyDrive and save them right back to your SkyDrive just like you would on your local hard drive. This will work with any app that supports open and save for documents and photos, and will be the first time anything like this has been possible without any setup or configuration. All you need to do is register your email address on a PC that's running Windows 8 and then, whenever you save files on SkyDrive, every Windows 8 device you use will provide seamless access to those files.

For app developers, this means that, so long as your app supports opening and saving documents and photos, it will automatically support SkyDrive without any additional work.

SkyDrive will also be available via the Share charm, which allows you to send documents or photos through the Mail app on Windows 8. With

one tap, you will be able to choose to share files through SkyDrive [instead of sending them as attachments](#), which means you won't have multiple copies of your files, each with their own set of changes. And of course, you won't be limited to the small number of attachments and total file size of most email services since the files are stored in your SkyDrive.



The best part about building this on the Windows 8 platform is that any Metro style app that enables its users to open and save files will get SkyDrive integration for free. Without adding a single line of code, users of the app will be able to access and store files on SkyDrive. So essentially, SkyDrive is one touch away from any Windows 8 app.

SkyDrive on the desktop

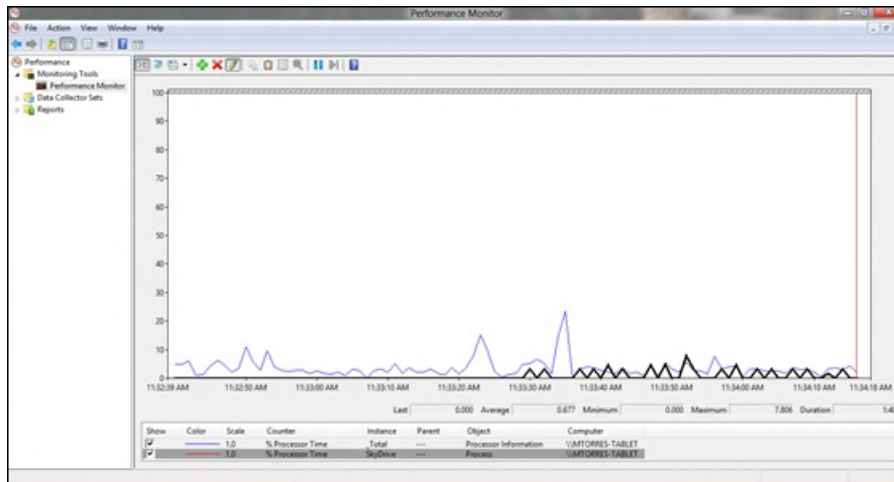
We're also bringing the power of SkyDrive to every Windows 8 desktop through Windows Explorer and to desktop apps like Office as well.

Since the start of SkyDrive, we've consistently heard from our most loyal customers that you want SkyDrive on the desktop, and we're happy to announce that we will be releasing a desktop app. The benefits are obvious: easy drag-and-drop upload and download support for SkyDrive, anywhere access to your data, offline access, and the power of Windows Explorer to manage your files and folders. All of these things will be available with SkyDrive on the desktop.

To start, we took what we learned from Mesh and FolderShare/Sync and built a very simple, highly efficient app. This app will be available with an installer that's less than 5MB and that takes about 10 seconds to install. You'll only need to install it once per PC, as SkyDrive will always keep itself up-to-date. Once installed, your entire SkyDrive will start syncing into the folder you choose (the default location is in a SkyDrive folder under your user folder: %userprofile%\SkyDrive\) and it will always be up-to-date with your latest changes. As you update files on your PC, they're uploaded immediately to the cloud—and as changes are made in the cloud or on another device, they'll sync back down to the PC. There's

very little to manage or control and you won't be bugged with pop-ups or dialog boxes. You won't even need to know it's running, as we've also spent a lot of time on network efficiency and overall performance.

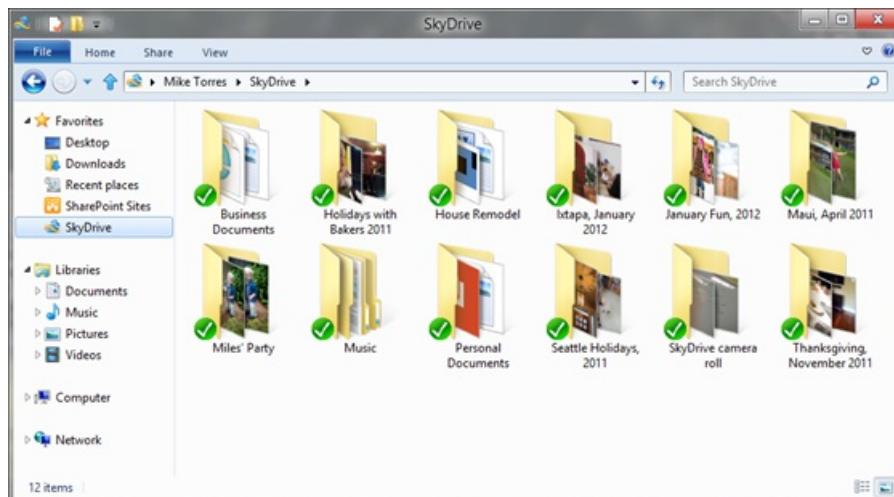
As an example, here's an image from Performance Monitor showing performance while a 500MB file is progressively uploading from the desktop to the cloud. You can see how little of the CPU is being used for SkyDrive during idle as well as for the file transfer halfway through (the blue is total CPU, the black is SkyDrive).

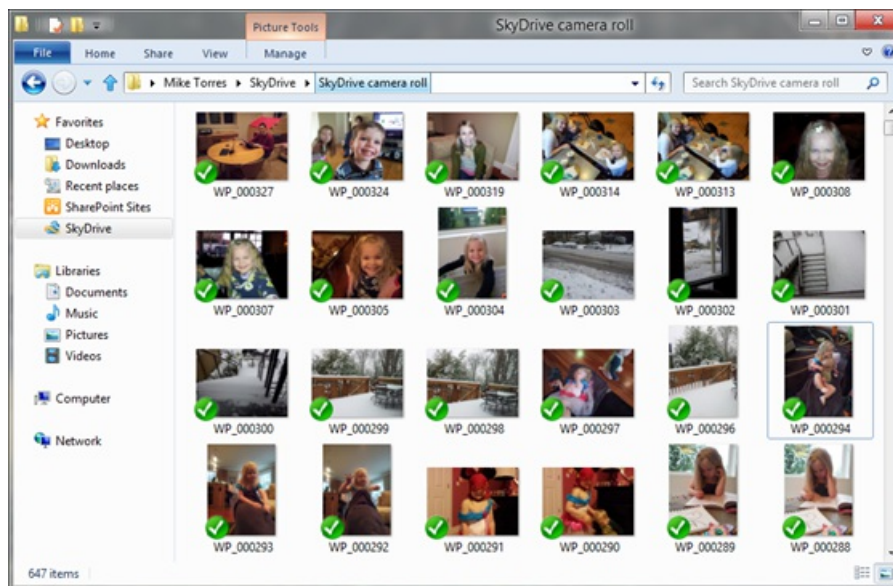


Of course, there are people who will use older versions of Windows for a while, so SkyDrive on the desktop will also run on Windows Vista and Windows 7. If you want to make sure your files come with you to Windows 8 and you're still running Windows 7, you'll just put them in your SkyDrive folder. This makes it easier to upgrade to Windows 8 or make sure you can access your files across all of your PCs.

SkyDrive for the desktop will also provide the ability to sync up to your available quota of storage (and the ability to unlock more), along with unmatched performance on your PC. Oh, and we will also have **support for uploading large files (up to 2GB)** through Explorer, another big request from SkyDrive.com users over the years.

SkyDrive on the desktop gives you a no-compromise cloud experience. Here are some preview images of SkyDrive in action on the desktop. The first one is the SkyDrive folder visible in Windows Explorer, and the second shows the entire Windows Phone camera roll synced automatically to the desktop:





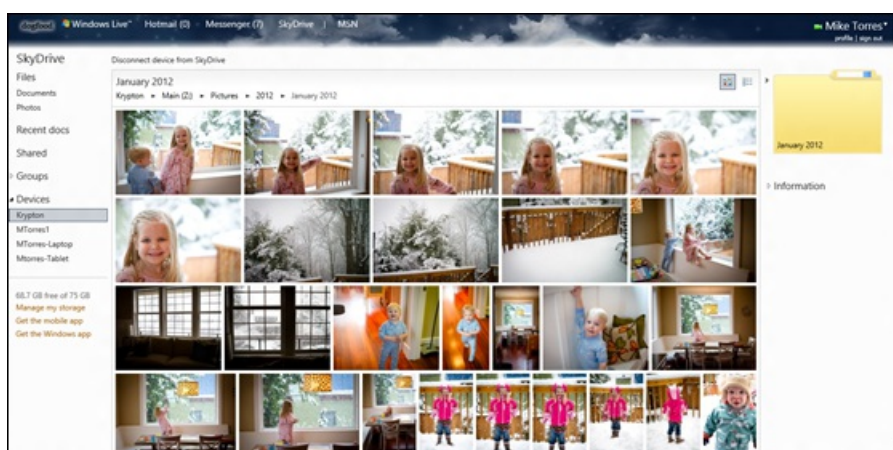
Fetching files through SkyDrive.com

With SkyDrive in Windows 8, you won't just have cloud files synced to your PC. You can also turn your entire PC into your own private cloud, and use its terabytes of local storage to easily access, browse, and stream your files from anywhere by simply fetching them from SkyDrive.com.

When designing the new SkyDrive, we knew not everyone would want to put 100% of their files in the cloud just yet. People are selective, and while some will move all of their files into SkyDrive, others will want to start slowly and use SkyDrive just for roaming some important documents and pictures from their Windows Phone camera roll. Knowing that most people would still have files on a remote PC that weren't available through SkyDrive, we built a new feature that allows you to "reach across" the Internet to access any file, stream videos, or view photo albums from a remote PC that is running SkyDrive on the desktop. For any remote folder or file, you can also choose to "copy to SkyDrive," so that you'll always have it across your devices.

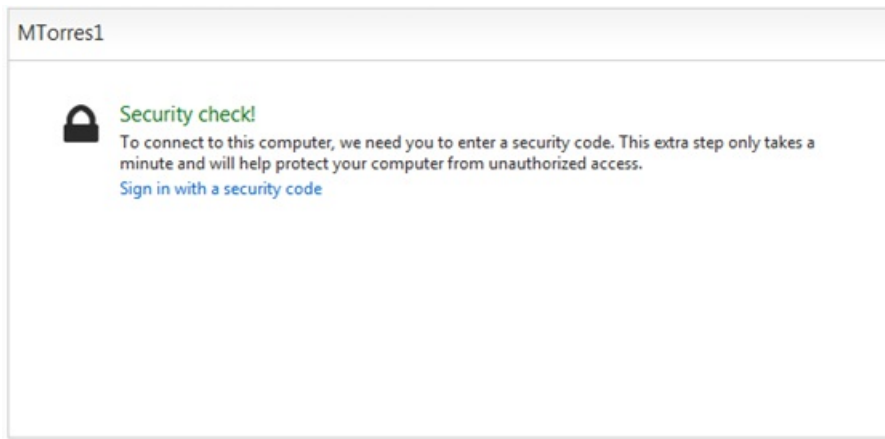
Internally we call this the "Forgot something?" feature. If you forget to put something on SkyDrive, you'll still be able to access it on your remote PC through SkyDrive.com. We've done special work to enable remote streaming of video, and we'll treat photo albums on your remote PC exactly as we do photo albums in SkyDrive, with the same beautiful full-screen viewer. We realize this is more of an enthusiast feature, as most people won't have an always-on PC at home, but for those who do, fetching files works like magic.

Here's how your remote PC will look in SkyDrive.com, and what it will be like to browse your Pictures folder remotely:

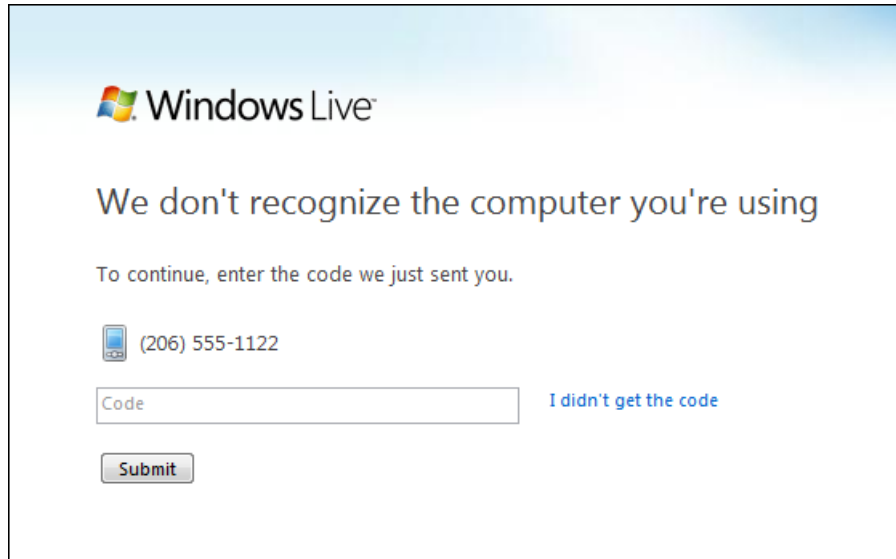


Of course, being able to remotely access a PC from a web browser has the potential to be abused. If someone has access to your account, without further protection, they'd be able to browse the entire contents of any SkyDrive-connected PC that happens to be online at that time. Naturally that isn't something we're comfortable with, so we've added another layer of protection for your remote PCs.

In order to access a remote PC, you will have to provide a second factor of authentication. This requires that you type a code that we send to your mobile phone or alternate email address in addition to having access to your account (if you're already on a [trusted PC](#), you won't have to do this every time). This means that anyone wanting access to your remote PC would have to have access not only to your account, but also to either an alternate email or your phone (which they would need to physically possess). Here's a preview of this:



Two-factor authentication is required before you can connect remotely



We'll verify your identity using a code that we send to your phone in a text message

This is just the beginning

This is just a brief overview of some of the things SkyDrive will do to make personal cloud storage a seamless part of the Windows 8 experience.

Of course, you will still be able to access all of your files via SkyDrive.com from any web browser with one of the fastest sites on the web. You'll also be able to work together online from anywhere using Office and the Office Web Apps. We will continue to make SkyDrive available on the devices you use every day, like Windows Phone and others, so you can access your entire SkyDrive on-the-go.

Over the next several months you'll have access to all of the things we've discussed in this post. While there will be a lot more to say about SkyDrive as our story evolves, we hope a small peek under the curtain has helped get you excited for SkyDrive on Windows 8.

Mike & Omar (aka "Momar" around the office)

Running the Consumer Preview: system recommendations

Steven Sinofsky | [2012-02-29T07:00:00+00:00](#)

*We're very excited to get to the point in the project where we can provide a pre-release version of Windows 8 that is broadly usable by the tech community on a daily basis. We know folks who are anxious to run the Windows 8 Consumer Preview are probably interested in suggestions around what hardware to use. This post provides the technical details behind system recommendations. These are **not system requirements**, and are not final, but simply a view of what works best for running the Consumer Preview. **Grant George, our corporate vice president of Windows Test, authored this post.***

--Steven

In the beginning of the Windows 8 project we started with the notion that you should get all that you need for computing from a single PC—that you do not need to choose between two desirable options, but that you can indeed have what you want in a single PC. The Consumer Preview is about saying “of course you can” to all those things you want to be able to do with your PC. You don't need to choose between consumption and productivity, between portable and powerful, or between touch and keyboard. Windows 8 doesn't force you to compromise. It isn't about “modes” of working, but is about seamlessly moving through all the things you want to do on your PC in the way you want to do them.

And the first step in achieving that was making sure Windows 8 works great with the hardware and peripherals you already have. To run the Consumer Preview, we recommend you start with your Windows 7 PC identified by a Windows 7 logo. While some software and devices will require updates provided by the manufacturer, we are committed to supporting any device with a Windows 7 logo. Occasionally the manufacturer will decide that a PC or peripheral is not supported and will say so on their website. However, since we are still in pre-release, not every manufacturer has this information available and so we ask your patience during this testing period. Software utilities such as security, management, and disk tools are normally tied very closely to the specific Windows version for which they were designed and will require updates from the manufacturer.

Whether you have a logo PC or you've built your own PC, the recommendations for the Consumer Preview include:

- 1 GHz or faster processor
- 1 GB RAM (32-bit) or 2 GB RAM (64-bit)
- 16 GB available hard disk space (32-bit) or 20 GB (64-bit)
- DirectX 9 graphics device with WDDM 1.0 or higher driver

This setup gets you going with Windows 8 such that it is functionally equivalent to Windows 7, and as we have talked about previously, you should see measureable improvements in performance in a number of dimensions with a system at this level.

One new element to Windows 8 is the requirement that **Metro style applications have a minimum of 1024x768 screen resolution, and 1366x768 for the snap feature**. If you attempt to launch a Metro style app with less than this resolution (e.g. 800x600, 1024x600) you will receive an error message. Since the software is in everyone's hands now, we will follow up with a more detailed blog post where you can learn more about the work we did for scaling across multiple resolutions and why this is a requirement in order to make sure developers can easily build applications that scale well across resolutions. We chose to allow Windows 8 to install even when a system doesn't meet this requirement because, even without Metro style applications, your Windows 7 workloads on these PCs will improve and you can benefit from all the other features of Windows 8, including enhancements to the desktop. We have made sure that Start and Settings all scale well on 800x600 resolution screens.

Many of you asked about running on a virtual machine. **Our recommendation for the Consumer Preview is to run it natively on hardware if you intend to run Windows 8 on hardware when the product is final.** Some of you will run virtualized environments for enterprise workloads or specialized purposes, but we strongly recommend that you experience Windows 8 on hardware, as it was designed to run for the majority of consumer experiences. The most important reason is access to the rich experience powered by accelerated graphics, and the fast and fluid operation that you'll experience when running this way. If you do run in a VM, which is supported as expected, please be sure your screen size meets the minimum requirements.

We previously outlined some of the requirements for touch hardware if you wish to experience Windows 8 in a touch environment (a touch tablet, convertible, all-in-one, or touch-capable monitor). We want to provide a short update on touch below and will post more about this soon. With the Consumer Preview, if you want to support touch, you will need a screen that supports multi-touch.

Although there are a number of existing Windows 7 touch devices and many are fully supported, we do recognize the touch experience of Windows 8 places a greater demand on a high quality experience than could have been foreseen when manufacturers were developing hardware for Windows 7. Our data is showing that a vast majority of Windows 7 touchscreens will perform well for Windows 8. This means that touch drivers continue to load, and you'll be able to perform basic touch interactions with a reasonable degree of success. The following systems are a few that we have been using widely in our internal testing and self-hosting, although of course, this is not a specific endorsement of these PCs:

- HP Elitebook 2760p convertible (Note: This PC is 1280x800 and so does not support snap.)
- ASUS EP121 tablet (Note: his PC is 1280x800 and so does not support snap.)

- Dell Inspiron Duo convertible
- Lenovo x220t convertible
- 3M M2256PW 22" display (Note: The raised bezel can make it harder to swipe along the edges)
- Samsung Series 7 slate (Note: This PC has two models, one was provided to attendees at //build/ and the other is a commercial release; the latter has slightly different peripherals and firmware.)

It is also worth noting a couple of other features that have specific hardware requirements. (Note: be careful whenever you adjust your BIOS settings.)

- Secured Boot requires a new **UEFI BIOS**, which is not available broadly on PCs yet, but is starting to be made available. If your machine does have UEFI, you can enable it via BIOS settings.
- BitLocker does not require but performs more seamlessly if your PC has a **Trusted Platform Module (TPM)**. Machines that have this sometimes require it to be enabled via BIOS settings. BitLocker To Go requires a USB flash drive that meets performance criteria evaluated at installation time.
- Hyper-V requires a 64-bit system with **second level address translation (SLAT)** capabilities and an additional 2 GB of RAM. You can also enable SLAT via a BIOS setting.
- Some games and other software require **graphics capabilities compatible with DirectX 10** or higher (including some games available in the Consumer Preview and in the Windows Store. We will continue to improve the verification of your system prior to downloading or running software with these requirements). Some games and programs might require a graphics card for optimal performance.
- If you clean install instead of upgrade (see below), you should check your PC manufacturer's website to make sure you **install any specific drivers** that they provide there. Many laptops will get better battery life with a power-optimized driver that is specific for that PC (often known as ACPI, Power, or Chipset driver).

Oh, and for a lot of things you'll probably want an active Internet connection! Be sure to check out the [metered network support](#) if your connection has limits.

For those of you who have already been running the Windows 8 Developer Preview, you can install the Windows 8 Consumer Preview using the migrate option (just keep personal files), but not the upgrade option (keep personal files, apps, and settings). Or if you prefer, you can of course do a clean installation (keep nothing). The Consumer Preview release does permit upgrading from Windows 7, and will run the integrated upgrade advisor to check on any things you might need to look into. Please keep in mind that **there is no rollback after an upgrade installation**. We also strongly recommend that you **perform a system backup** prior to an upgrade, migrate, or clean install of Windows 8 Consumer Preview.

Also note that the *final* release of Windows 8 will not support upgrading from any prior Windows 8 "Preview" release, though the migrate option will still be supported. In any upgrade scenario, you can run the Disk Cleanup Wizard to remove the previous installation in order to free up disk space. The download will also support boot from USB for a completely clean installation as well.

Happy downloading and installing!

Grant George

Welcome to Windows 8 – The Consumer Preview

Steven Sinofsky | [2012-02-29T06:45:00+00:00](#)

Today is a big day for the Windows team. At Mobile World Congress in Barcelona, Spain a few moments ago, we unveiled the Windows 8 Consumer Preview to our partners and press. Based on a broad range of feedback, **we have made over 100,000 code changes** and the Consumer Preview represents a refined product ready for broad and daily usage by those of you willing to test a pre-release OS. You can **download the Consumer Preview starting now at <http://preview.windows.com>**. If you tried the Windows 8 Developer Preview, then you are going to be delighted to see a broad range of product changes and improvements based on a feedback from many sources.

Windows 8 reimagines Windows, from the chipset to the experience. With the Developer Preview we focused on presenting the new APIs and amazing new tools for developers. Today's Consumer Preview is focused on a broader audience, and along with improvements to the WinRT APIs based on developer feedback, we are introducing the full user experience, the Windows Store for apps, and early previews of some first- and third-party apps.

With so much to dive into, let's talk about what is different in the Consumer Preview at a high level:

- **Broad range of product changes and improvements:** Since the Developer Preview in September, designed to preview the programming platform, Windows 8 has progressed across every dimension. From completing the user experience for touch, keyboard, and mouse, to refining the development platform, to improving performance, quality, and reliability across all subsystems as well as new features, the Consumer Preview represents a complete view of the capabilities of Windows 8.
- **Windows Store with an “App Preview” of new apps:** The Windows 8 Consumer Preview marks the opening of the Windows Store for testing. You'll see a variety of new Metro style apps from both third-party developers and Microsoft. During the Consumer Preview, these apps are available to try and experience at no cost to users. Please note, these apps and the set of preinstalled apps are at an early stage of development and are available as an early *App Preview*, and will be updated via the Windows Store. In addition, the Store will offer personalized recommendations, and Windows 8 gives users the ability to take their apps and settings with them across multiple PCs, making it easy to discover and try new apps while offering developers the greatest opportunity of any platform
- **Connecting to the cloud across Windows PCs and Windows Phones:** You'll experience seamless integration with the content across your web services. Optionally signing in with a Microsoft account provides access to features including the ability to roam all settings, use cloud storage, communicate with email, calendar, and contacts, and connect to a broad range of services. Your connection to the cloud works across your Windows PCs and your Windows Phones. You'll also experience early previews of the Metro style apps for Mail, Calendar, People, Messaging, Photos, and SkyDrive.
- **Internet Explorer 10 Platform Preview 5:** With IE10, we reimagined the browser to create a new experience designed specifically for Windows 8 devices. It provides an edge to edge interface that is all about less browser, and more web. Fast and fluid, IE is hardware-accelerated to enable web performance. The same rendering engine and high-performance script engine is available on the Windows desktop as well.

We've detailed many features in this blog across all the subsystems of Windows 8. From the kernel, networking, file system, graphics, and the user interface across all of those. There's no easy way to enumerate the depth and breadth of Windows 8 in a post. The best thing to do is experience it yourself. We encourage everyone to check out [our demo video](#), and all the videos and information on <http://preview.windows.com>. From there you can also download the Consumer Preview for x86/64. For developers, there is also a [beta of Visual Studio 11](#).

We'll publish a quick look at system requirements for this release, but the short version is that your Windows 7 logo PC is the perfect place to start as the system requirements have not changed. You can upgrade from the Developer Preview or from Windows 7, or install cleanly (we strongly recommend a hardware installation and not a VM install if you are looking to experience the release as the vast majority will experience it, and please keep in mind the minimum screen resolution required is 1024x768). We will be updating the release with various quality updates and drivers over the coming weeks/months just to exercise our overall update and telemetry mechanisms. Please keep in mind that this is a test release of a product still under development.

We've got a lot more blogging to do. So stay tuned for details of the changes we made and the features we haven't had a chance to talk about yet. This blog continues to be a big part of the development process. Now that we have this shared experience, we expect folks commenting on posts to be running the Consumer Preview so we're all sharing the same context. We know there will be a lot of feedback—that comes from reimagining a product used by a billion people!

Happy downloading and testing!

--Steven on behalf of the Windows 8 team

Going behind the scenes building Windows 8

Steven Sinofsky | [2012-03-06T10:00:00+00:00](#)

*As we start to talk about some more of the details of the Consumer Preview release and some of the changes, or just features we haven't had a chance to blog about yet, we wanted to take a step back and re-introduce the team a bit—a reminder of the people behind the features we talk about. Building Windows 8 is a pretty significant undertaking and involves a team with diverse backgrounds and experiences. We're proud of the fact that the diversity on our own team reflects the diversity of the customers using Windows around the world. Last release, one of the members of the team, **Larry Osterman**, wrote about working on Windows 7 compared to previous releases. In this post, Larry reflects on the Windows 8 project through two other members of the team.*

—Steven

Three years ago, I wrote a [post on the Engineering Windows 7 blog](#) about the Windows 7 development process. For this go-round, we thought we'd let you hear from some of the newer members of the team, by doing an informal Q&A session with two members of our Windows Runtime Experience team, both of whom started in Windows just before we started planning Windows 8 (so, Windows 8 is their first experience with developing Windows from start to finish).

Tell me a bit about yourself. Where do you come from and how long have you been at Microsoft?

Chris: Hi, my name is Chris Edmonds. A native Oregonian, I attended school at Oregon State University (Go, Beavers!) and have had internships at NASA and Garmin. During these experiences I worked on projects ranging from robotics to avionics and did research on high speed routing for many-core processors. Microsoft recruited me from Oregon State, and I arrived on the Windows team roughly two and a half years ago.

Mohammad: Hello, My name is Mohammad Almkawi. I am a software design engineer in the Windows division at Microsoft. I have also been at Microsoft for about two and a half years. I graduated from the University of Illinois at Urbana-Champaign (Go, Illini!) where I was working on fault-tolerance and real-time systems integration research.

What do you work on for Windows 8?

Chris: I started working with the Windows team a few months before Windows 7 was released to manufacturing. Shortly thereafter, I joined the newly created Windows Runtime Experience team. The Runtime Experience team builds many pieces of the Windows Runtime (WinRT) infrastructure. During Windows 8 development I had the opportunity to work across many parts of the WinRT.

In the first milestone (of three), I worked to define core patterns of the WinRT system. We break the project into three milestones and divide the architecture and implementation across these milestones to get us from a whiteboard sketch to a finished product. We have to include all the work it takes to coordinate the different technologies across Windows 8. In first milestone (M1), we designed patterns for events, object construction, asynchronous methods, and method overloading. It was important to define strong patterns for these basic concepts in order to allow each programming language that interoperates with the WinRT to expose these concepts in natural and familiar way for their developers.

In the second milestone, I had the opportunity to build part of our deployment story for Metro style applications. Specifically, I worked on registering Metro style applications with the WinRT so that they can be launched and can interact with contracts.

The third milestone included lots of cross-group collaboration, which I learned is crucial to a project as deep and as broad as Windows 8. I worked with a team to define and implement core pieces of the application model for Metro style applications. This work ensured that Metro style apps written in different languages and on different UI platforms behave in a consistent manner in regards to contracts and application lifetime.

Mohammad: I had the opportunity to participate in Windows 8 since the very beginning. We had three major feature milestones (M1, M2, and M3) to realize the goals of Windows 8. Each of these milestones consisted of:

- Specification and design phase to tease out requirements through feature crew meeting and active engagement with partner teams in Windows and across the company. A feature crew is made up of the developers, testers, and program managers who work on a specific feature—usually 4 or 5 people. The outcome of this phase was a set of specification documents—functional (pm), development design (dev), test design and a threat model (test)), as well as an execution plan (all of us). This lets us better understand features details and allows us to execute at high confidence with focused efforts.
- Coding phase to implement the features identified in the specification phase along with their unit tests and functional tests.
- Integration and stabilization phase to integrate the various pieces from various teams together and to fix bugs.

In the first milestone, I worked on the design and development of the discovery and activation of application extensions. This WinRT infrastructure allows applications to participate in [OS-supported contracts](#) (such as search and share) and serves as a basis for exciting Windows features, including the search and share charms.

In the second milestone, I was in charge of implementing the Windows metadata resolution feature, which is a key API that ties the Windows metadata generated by the WinRT tool chain and JavaScript and C# language projections.

And in M3, I was in charge for the design and development of the namespace enumeration API, which enabled the Chakra JavaScript engine to support reflection functionality over WinRT namespaces and types. CLR also uses this API to implement metadata resolution and Visual Studio uses it to support Intellisense for WinRT types.

What's a normal work day like?

Chris: Normal day? One of the things that I really like about working in Windows is that there is rarely a normal day. Depending on the period of the product cycle, I may spend my day writing specifications, writing code, hashing out ideas with people on my team, fixing bugs, or one of many other activities. Even though the activities are varied, my day almost always involves problem solving in some form. Whether it is figuring out the cause of a crash or helping to design features, I get to work with smart people to solve interesting problems each day.

What has been your biggest surprise?

Chris: I think the biggest thing that surprised me about working in Windows was the size of the team and the number of activities that are going on at any point in time. In working on the few features assigned to me I had the opportunity to interact with hundreds of other people across the team to come up with specifications and solutions. It sounds really hectic (and it was a little overwhelming at first) but it always amazes me how well teams communicate together to come up with some really cool solutions to problems. When I think of the number of people who use Windows and the number of ways Windows is used, I guess it seems incredible that we get all this done with *as few* people as we do.

Mohammad: The thing that surprised me the most at Microsoft is that you get thrown at real-world problems and you will be given the opportunity to own critical pieces from the very beginning. You learn on the job as opposed to training, which is also available if you need it.

Of course you are not left alone in the dark, as there are lots of support channels, domain experts, and senior engineers who are there to provide help when you need it.

How was Windows 8 different from other projects you've worked on?

Chris: Having worked mostly on smaller projects at Oregon State and in my prior internships (most code projects are small compared to Windows), the biggest difference is how much code I read every day. I find that I spend a good portion of time reading and debugging code written by other teams before I came to Microsoft, as well as going over code I wrote myself in a previous milestone. This has really made me appreciate well-written code.

What has been your biggest challenge that you had to solve?

Mohammad: Soon after joining the team, I had to make fixes in unfamiliar code in COM activation. This code is very infrastructural, as a lot of components in Windows are built on top of it, so it was crucial that my changes would not cause regressions.

This code might have seemed straightforward to experts in my team, but certainly that was not the case for a new guy like me. I had to read a lot of code, step through the debugger, and write lots of test cases to improve my understanding and confidence in making the necessary changes without breaking anything.

Can you tell me something about what it is like to come up with the plans for Windows 8?

Chris: Planning Windows 8 takes different shapes for different people on the team. As a part of the planning effort the newly formed Runtime Experience team took a week to build apps using a variety of languages, stacks, frameworks, and technologies. That's because a design tenet of Windows 8 is that it can be programmed in multiple languages. Part of the goal of this effort was to force each of us use a language that we were not already familiar with so that we could experience the learning curve. I worked on a 3D terrain generation program using IronPython and XNA, a photo gallery app using HTML/JavaScript, and a simple 2D physics engine in C++ using GDI for drawing. From the app building exercises, we created presentations to give to the team on the experience of building each app, along with a list of the good, bad, and ugly of each experience.

What impressed you?

Mohammad: I was very impressed by the quality of the Windows engineering system that we have in place; it supports thousands of Windows software engineers and keeps millions lines of code in the operating system healthy with nightly builds and quality gate runs. The automated quality gate runs include critical end-to-end tests, performance tests, application compatibility tests, static code analysis and a few other tests that we use to quickly discover problems and to tightly control their propagation across branches via forward and reverse integrations.

What is milestone quality (MQ)?

Chris: This milestone is all about getting the code base, engineering tools, and engineering processes ready for the next product

cycle. As I learned, MQ is a time to look across the code and do some housekeeping—from just cleaning up source files, to redoing abstractions that prepared us for the work we would do in Windows 8. Code is our asset so dedicating time to maintaining that asset is pretty important. During MQ of Windows 8 I participated in three different efforts. The first was to create a system that automatically reported back code coverage numbers via an internal dashboard for our team based on our daily test passes. This was one of the first things I worked on at Microsoft, and it gave me a great opportunity to learn about our engineering systems. The second effort that I participated in was a code sanitization practice to help standardize the way we use asserts across the code base. Finally, I worked on a prototype system that would use some pieces of our IntelliSense infrastructure to automatically catalog all parts of our SDK.

What are you focusing on now?

Mohammad: Performance, Performance, and Performance!

The features I owned are close to the bottom of the software stack and used very frequently, so their performance is very critical. Therefore, my focus now is on analyzing performance, and prototyping and integrating various performance improvements. We built things from the start to be high performance, so now we are fine-tuning that performance, given the tons of code that has been written to the infrastructure.

How do you validate the work end-to-end?

Chris: As part of a team dedicated to improving the application developer experience, it is important that we regularly take off our operating system developer hats and don our application developer hats. This is done in small ways in our everyday work, but one of the most structured forms of this are the application building weeks. Based on the initial application building week that took place during planning, we took the time each milestone to develop an application using the WinRT, with different teams focused on different languages and APIs. Writing apps on a platform that is still in development creates some interesting challenges, and these weeks are a fun change of pace. These app building weeks (some of which included more teams) have resulted in numerous bugs being filed, and have caused us to rethink and change some of our API guidance in order to make each developer's experience more natural and familiar. A "bug" can be anything from a fatal crash, memory leak, or security hole, all the way to a report that "something just doesn't seem right." We treat everything like a bug and go through a process of categorizing and prioritizing these reports. The reports come from the groups in Windows building on our APIs, other groups at Microsoft, early partners such as device and PC makers, our interns (as you saw at //build/), and from people in the forums who are building apps now on the Developer Preview.

What is the most important lesson you have learned?

Mohammad: I got to experience the idea that "anything that can go wrong will go wrong" given the size and scale of the product, and the large number of users (by the way, we do dogfood our work internally from the very beginning on our primary dev machines). This taught me that paying attention to details and focusing on quality in every line of code is very important for the overall stability of the product. Of course, that is just one of many important lessons I learned so far—I'm still working my way through my first Windows release and expect to learn a few more things during the upcoming phases of the product.

I can't wait.

Chris: Me too!!

Web browsing in Windows 8 Consumer Preview with IE10

Steven Sinofsky | [2012-03-13T15:30:00+00:00](#)

*In this post we want to talk about the new Internet Explorer 10 browsing experience. We have considerably improved the underlying browsing engine with [performance](#), standards, and features as we have previously blogged about. IE10 designed for a Metro style experience is a new and improved way of browsing, where you can truly focus on the information you want to browse rather than the task of browsing – a fully immersive experience. At the same time it provides all of the safety and controls you are used to – tabs, keyboard shortcuts, InPrivate browsing, and more. **Rob Mauceri, the group program manager for Internet Explorer, authored this post.***

--Steven

To deliver the best browsing across all Windows 8 devices, we re-imagined the architecture and experience of the web browser with Internet Explorer 10.

We built a new browsing experience in lockstep with Windows 8 to give you all the advantages that Metro style applications offer. We built that experience by extending IE's underlying architecture to provide a fast, fully hardware accelerated browsing engine with strong security and support for HTML5 and other web standards.

The result in Windows 8 Consumer Preview is a Metro style web experience. IE10 is designed to make website interaction fast and fluid for touch as well as for heavy mouse and keyboard use. With IE10, websites participate in the Metro style experience in Windows 8, including the Start screen, charms, snap, and more. IE10 also provides the best protection from malicious software on the web while providing real control over your online privacy.

While building and tuning the Metro style browsing experience for the Consumer Preview, we realized it is a better way to browse – whether on a desktop computer with a big screen, mouse and keyboard, or on a touch-enabled mobile device. As people browse more “chromelessly” on their phones, they've become accustomed to a more immersive and less manual browsing experience compared with the desktop. Metro style browsing offers you a full-screen, immersive site experience. With IE and Windows 8 you can always use the charms to accomplish what you want to do next with a website (e.g. share, print, search. . .). We've found that many people – even those with the most enthusiastic and intense browsing patterns – prefer Metro style browsing because it's less manual and more focused on what you browse than on how you browse.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

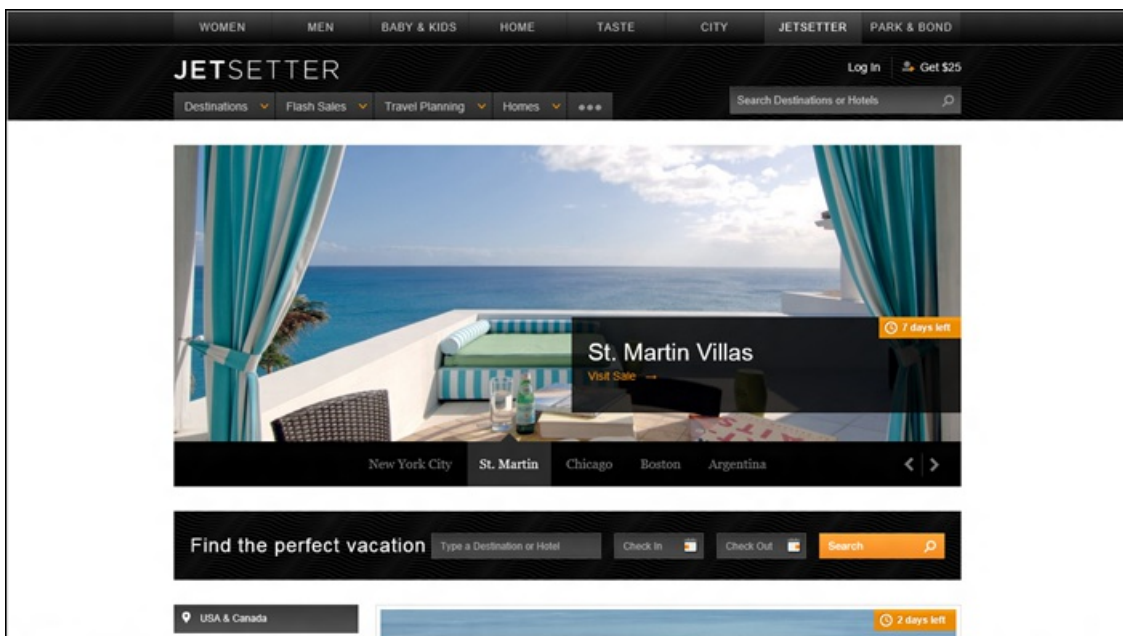
With IE10 in Windows 8 Consumer Preview, browsing the web is fast and fluid

You can read more about the technical details and architectural improvements to the underlying HTML5 “Trident” browser engine and Chakra JavaScript engine on the [IE Blog](#).

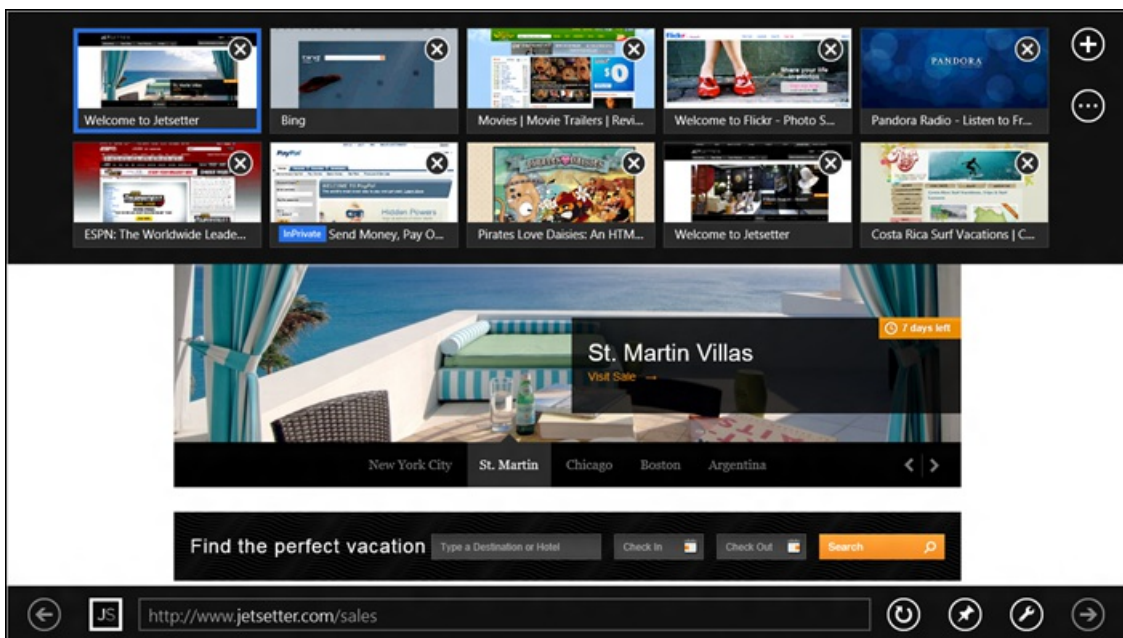
The Metro style web browser

We built IE10's user experience exclusively around all the Metro style design patterns to be fast and fluid for even the most intense everyday browsing.

We designed the interface and controls to be there when you need them and out of view when you don't. We also designed in the comprehensive functionality that people need for everyday heavy-duty web browsing: great touch keyboard support for forms, integrated spell checking with AutoCorrect, finding text on the page, etc. The user experience follows Metro style patterns and conventions for personality, animations, and command activation and support for Windows 8 charms, snap, and more.



IE10 puts the focus on your sites, providing a full screen edge-to-edge experience that uses every pixel for the web.

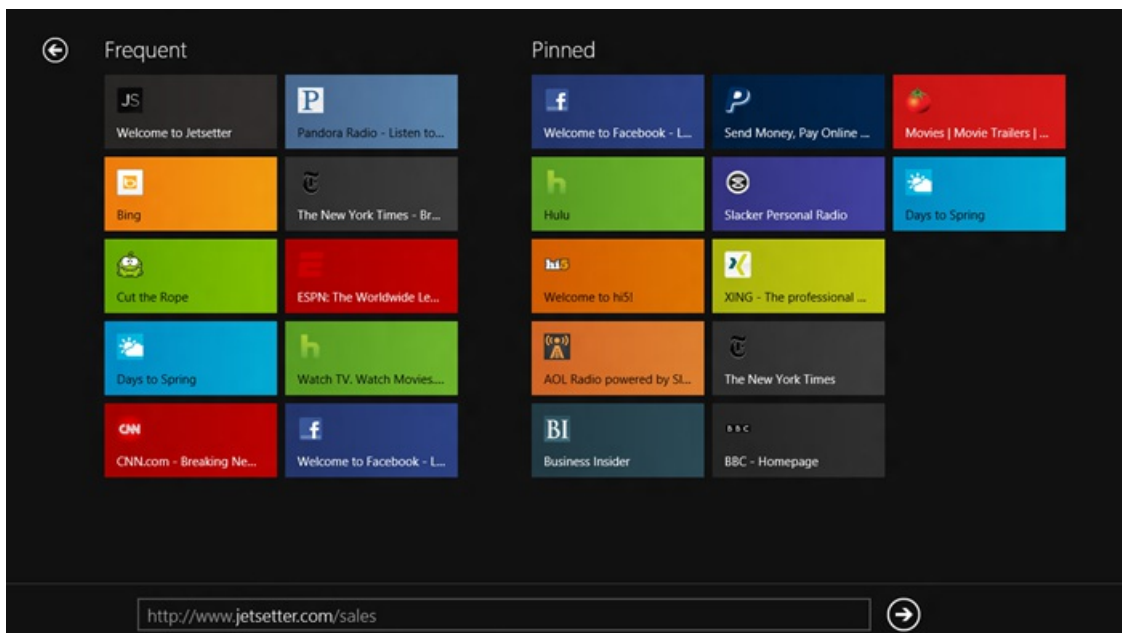


Tabs are available but stay out of your way until you need them.

IE10 is fast and fluid for the real web, not just the mobile versions of sites. We made IE super responsive to touch, mouse, and keyboard. The Metro style browser delivers on *touch browsing*, not just browsing on a touch device. You can feel it in the stick-to-your-finger responsiveness of the touch support for panning and zooming, swiping back and forward for page navigation, and double tapping to zoom in and out of content. Context menus and form controls are optimized for touch, and the browser responds fluidly to device orientation (scaling smoothly to landscape and portrait screen layouts) and “snapping” Windows 8 applications next to it. IE10 also improves on the experience of browsing the Web with mouse and keyboard with support for the keyboard shortcuts you expect, and convenient mouse activations for back and forward navigation.

Metro style IE10 takes a different, more modern approach to browsing. It puts the focus squarely on the websites you browse rather than all the tab and window management activity that has defined browsing to date. On our hallways, we’ve been using it as our primary browser on laptops and desktop workstations, with touch screens as well as with keyboards and mice. From tiles on the Start screen for websites to the immersive full screen web experience, we designed IE in Windows 8 to be your daily browser for the real web.

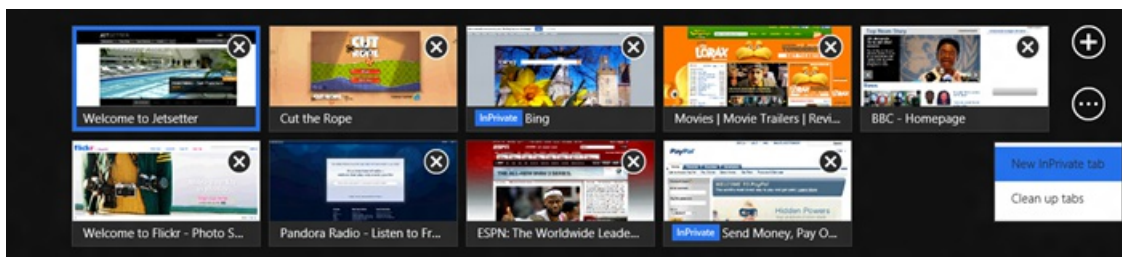
Navigation tiles are designed to help you find and navigate to sites immediately using the site’s icon and color while minimizing your typing. IE shows you frequently visited sites as well as sites that you’ve pinned to the Start screen.



Get to your most important sites quickly with navigation tiles

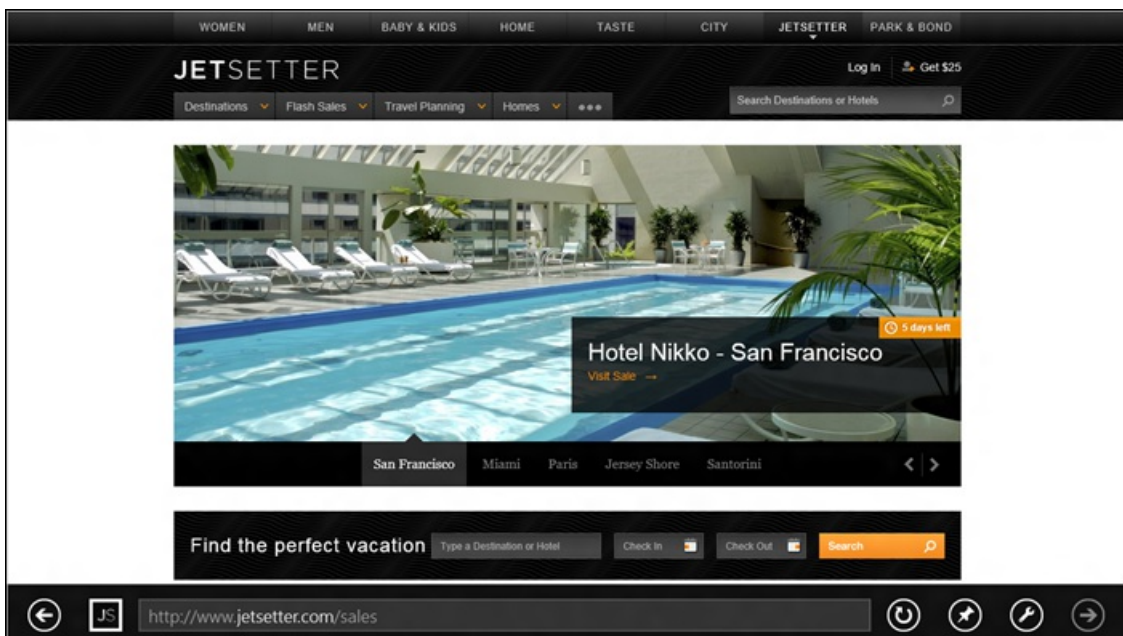
As you type in the address bar, the navigation tiles filter to show you sites from your history, favorites and even popular URLs. With Windows 8 roaming and connected accounts, your browsing history and favorites roam with you so that you can easily access recent webpages across all of your PCs.

Tabs: Browsing multiple web pages is core to any good web experience. The Metro style tab switcher appears when you swipe in from the bottom or top of the screen with touch, right-click with the mouse, or press Windows key+Z on the keyboard:



Active tabs are shown as page thumbnails with page titles in text overlays. Tabs have a touch-friendly button for closing, and button for creating a new tab, or a new InPrivate tab. IE10 shows the last 10 tabs you've used, reducing the need to actively manage your tabs. You can even clean up tabs quickly and easily with one command.

The **Navigation bar** in IE10 appears when you need it, again keeping the focus on websites. The navigation bar includes easy-to-use controls (touch or keyboard/mouse) for common operations like back, forward, stop/refresh, and pinning sites to the Start screen. The address bar shows badges and coloring for secure sites, SmartScreen, and InPrivate browsing. It also supports auto-complete as well as web search, matching the behavior of IE on the desktop. The address box shows a progress indicator when a page is loading, and includes indicators for site compatibility and tracking protection. The navigation bar includes commands for Find on Page, and Open in IE on the desktop, for compatibility with sites that require older plug-in technologies, or for when you are using desktop tools and wish to continue using them in your existing workflows.



Touch keyboard: IE10 works great with physical keyboards as well as the Windows 8 touch keyboard, which it automatically adjusts to make your experience easier. For example, when you set focus in the address bar, the “?” and “.com” keys become available to quickly enter URLs:



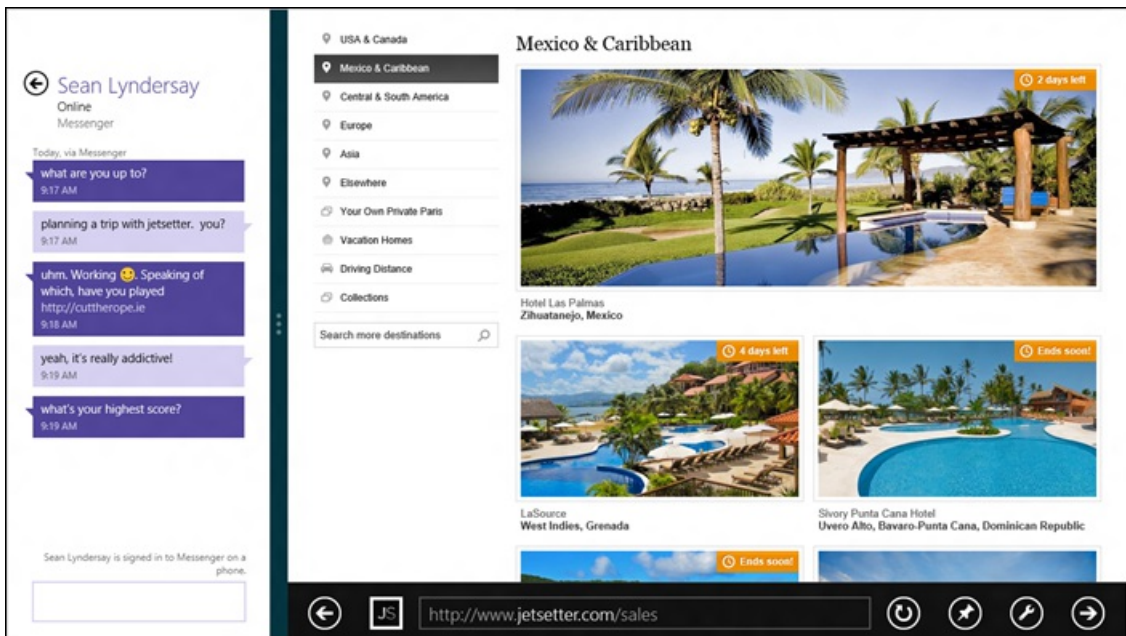
IE automatically adjusts the touch keyboard based on where you’re typing. For example, email form fields show the “@” and “.com” keys

IE10 takes a clean, “low nag” approach to **notifications**. All alerts and user prompts come through a notification bar at the bottom of the screen. IE uses Windows 8 Metro style “fly-outs” when more interaction is needed. Notification bars automatically dismiss as appropriate. Downloads in the Metro style browser protect you from malicious software via SmartScreen’s Application Reputation, as in IE on the desktop.

Connecting websites and apps in the Metro style

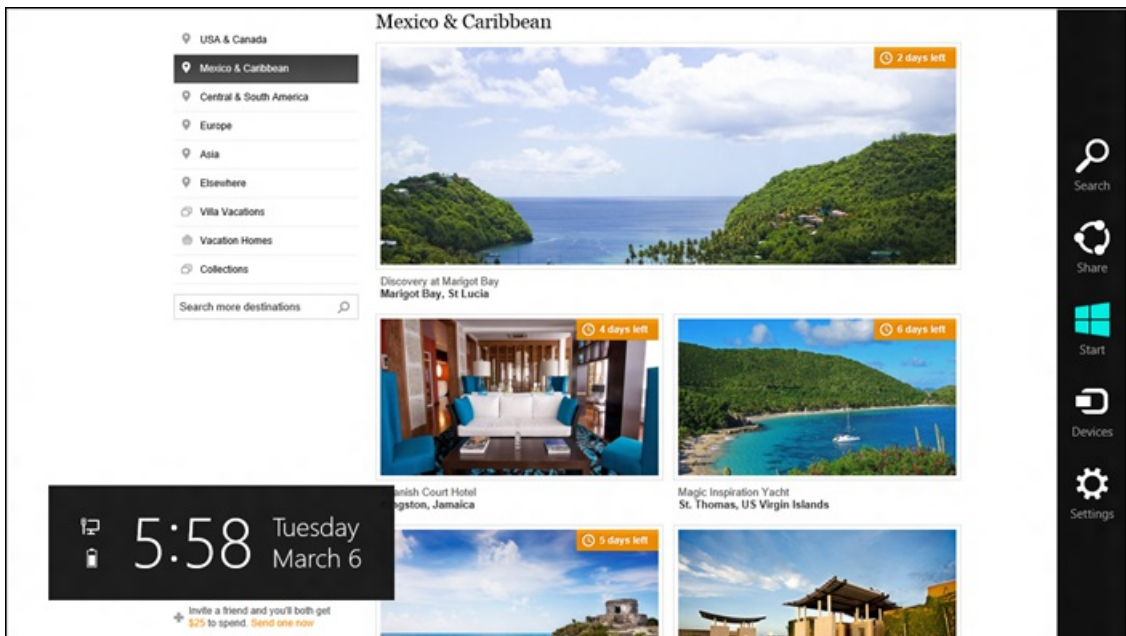
With IE10, websites are part of the Metro style experience in Windows 8. Through snap, charms, and integration with the Store and the Start screen, Metro style browsing blurs the boundaries between the web and apps.

Snap makes it easy to use Windows 8 for more than one thing at a time. You can browse in IE10 and have side-by-side access to your mail, music, or any other application. The browser adapts to the narrow “snap” size and automatically undocks when necessary for user interaction. All of the core browsing capabilities are available when snapped – panning, pinch and double-tap zooming, and following links.



Multitasking with Windows 8 “snap” lets you put your site side-by-side with other applications like the Messaging app

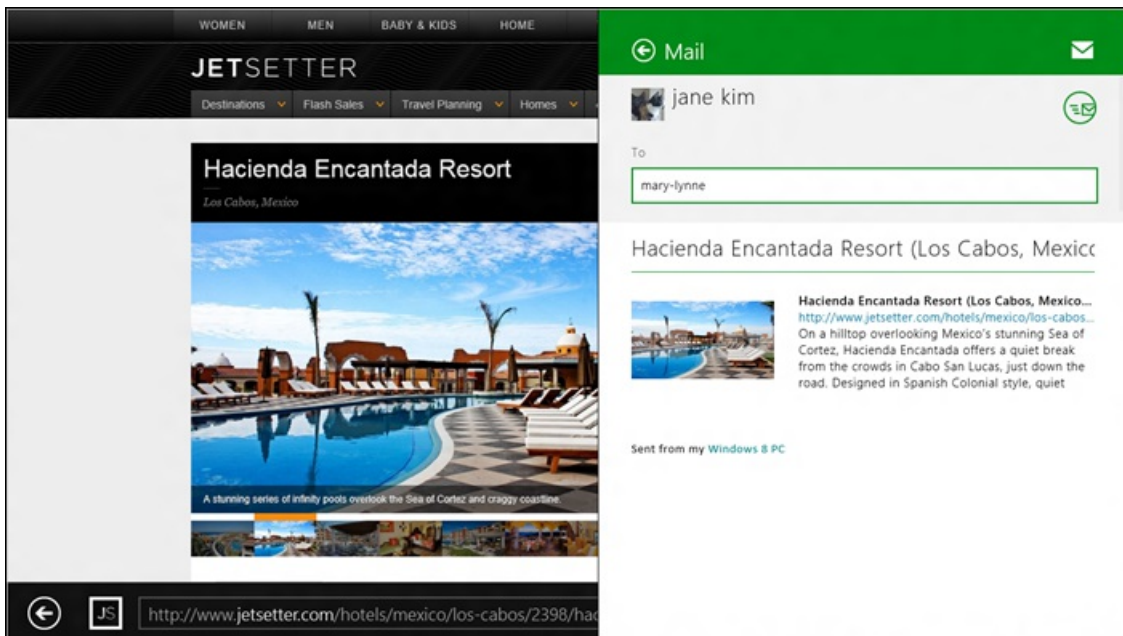
Charms provide a consistent way to perform common actions like searching and sharing in Windows 8. IE10 supports the Search, Share, Devices, and Settings charms:



The charms appear when you swipe in from the right edge, press Windows key+C, or move your mouse to the bottom or top-right corner of the screen.

For the **Search** charm, IE10 uses the default search engine, which you can set to your preference. After initiating a search in the charm fly-out, search results are shown as you type, including the same picture and instant results you see in IE on the desktop, if your search engine supports them

With the **Share** charm, you can access any application that supports sharing (like Mail). This allows you to send a rich link preview with image, description, and hyperlink so it's easy to share more than just a link.



IE10 and Mail support sending rich link previews with image, description, and hyperlink, you can share more than just a link with very little work.

The **Devices** charm makes printing, projecting, and playing to external devices easy and consistent. For example, you can print from any webpage from IE – handy for things like airline boarding passes – by tapping or clicking the Devices charm and selecting a printer.

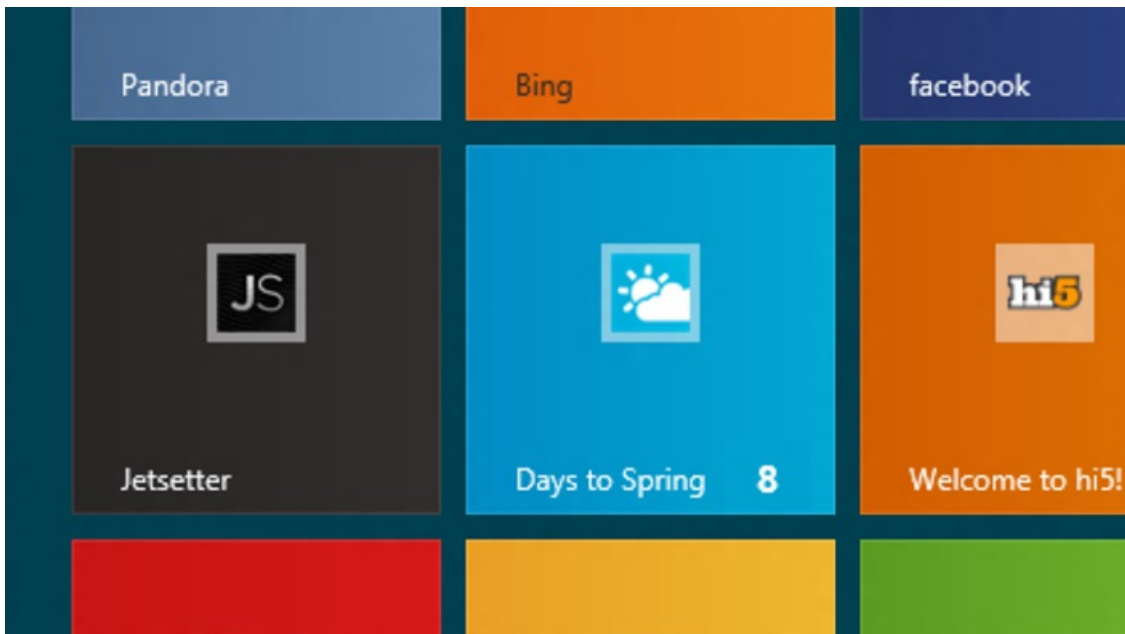
The **Settings** charm provides quick access to the most frequently used configuration settings for IE10. You can quickly clear browsing history, control location access, and more. Consumers get a simplified interaction with IE settings, while enthusiasts still have an easy way to access fine-grained controls through settings in IE on the desktop.

With **site pinning**, you can personalize your Windows Start screen with the sites you use all the time. You can pin any website to the Start screen from IE10, so you have one place to access all the things you care about or need.

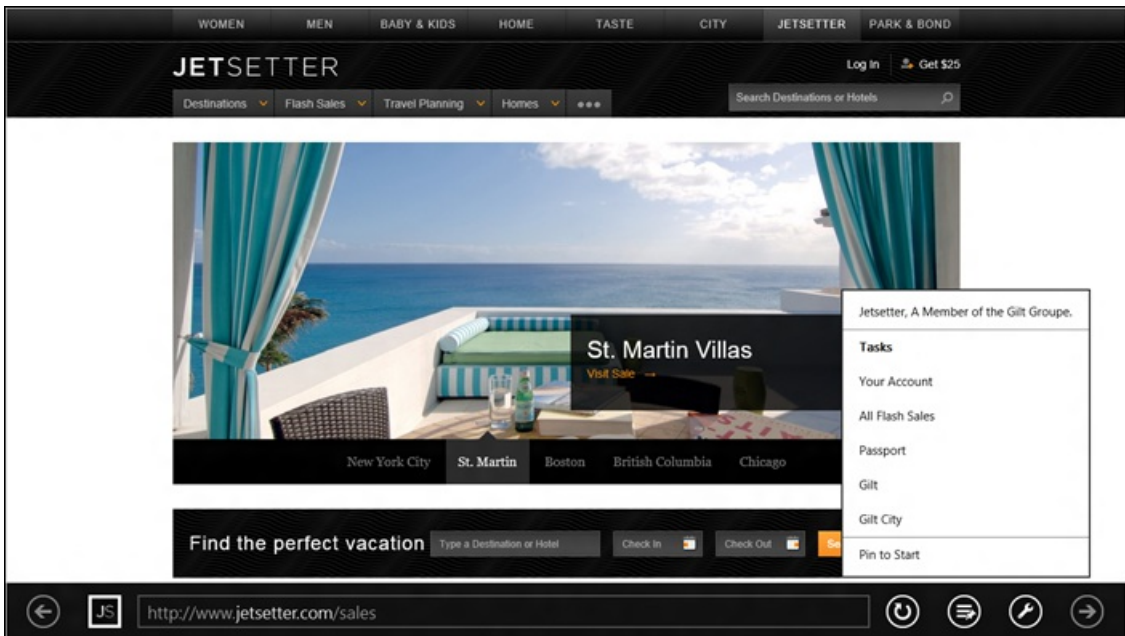
The tiles for pinned sites reflect the site's color and icon. With IE10, sites can provide background notifications for new messages and other account activity on the website. The site can also program additional commands that appear in IE's navigation bar in a touch-friendly way, the same way that sites can program jumplists for IE on the desktop.



Site tiles let you go directly to your sites from the Windows 8 Start screen



Pinned site tile notifications keep you up-to-date at a glance, without opening the site



Jumplists make it fast to get to site sections and information tailored for you

Integration with the Store makes it easy to discover and launch Metro style apps for the sites you visit in IE. The navigation bar shows if the site has an application available. One tap (or click) takes you to the app in the Store. Once an app is installed, you can launch it directly from the site. For example, here's cuttherope.ie in IE10:



Site favicon button lets you download and launch the app with a single tap or click

Protecting you from malicious web

IE10 offers the same industry leading security, privacy, and reliability features, building on IE9's SmartScreen, XSS filtering, Application Reputation, InPrivate browsing, Tracking Protection, and hang detection and recovery. In addition, IE10 takes advantage of Windows 8 to provide "Enhanced Protected mode" for better isolation of website content in each tab. InPrivate browsing is also extended to run per-tab, so you can run some pages InPrivate, leaving no history, cookies, or cached data.

Summary of changes from the Developer Preview

IE10 in the Windows 8 Consumer Preview brings a more full-featured Metro style experience to your browsing. Here are just some of the improvements to IE10 for fast and fluid browsing:

- Full, independent composition enables responsive, fast and fluid behavior on real websites (including pages with fixed elements, nested scrolling regions, animations, and video)
- Back and forward swipe navigation with preview
- Double-tap to zoom in on content
- Fast back and forward navigation controls for mouse
- Mouse (CTRL+scroll wheel) and keyboard methods for quickly zooming in and out to mirror touch interactions
- Automatic domain suggestions for faster navigation and less typing
- Share charm support for URLs, snippets, images and selection with Mail and other apps
- Search charm with visual search suggestions
- Devices charm for printing, projecting, and playing video to external devices like TVs
- Plug-in free support: notifications for sites requiring activeX
- Background notifications for pinned sites and other tile improvements
- Jumps lists for pinned sites
- InPrivate tabs that are easier to open
- Clean up tabs command, which quickly closes all but current tab

Metro style and no-compromise browsing

You used to have to make a choice between browsing the mobile web on small screens with good touch support, and browsing the full web with good mouse and keyboard support on big screens. The Metro style web experience in IE10 in the Windows 8 Consumer Preview means no compromises. You can browse and touch and multitask and print and share with all the power of Windows 8 and your PC. The web with IE10 is more fast and fluid, better connected to your applications, and more secure and private.

--Rob Mauceri

Scaling to different screens

Steven Sinofsky | [2012-03-21T17:35:00+00:00](#)

There's a ton of innovation in the world of displays—from pixel density, to aspect ratio, to core technologies. Windows 8 is designed to grow and improve as the display ecosystem grows and improves. Our goal is to support the broadest range of display technologies so PC makers can build PCs or you can use external displays that provide the best experience for your needs. To do this, we architected the WinRT to provide the platform necessary to support this diversity. This is a complex post that looks at the details and nuances around supporting many permutations of physical screen dimensions, pixel densities, and resolutions. There's a lot more to this than “my 27” monitor,” as you can see in [this post authored by David Washington, a senior program manager on our User Experience team](#).

—Steven

One of the core promises of the Windows platform has been its support for diverse form factors, allowing Windows to power over a billion PCs in the market today. In Windows 8, we set out to build upon this strength by delivering a great experience regardless of the form factor or screen size. Windows 8 PCs will come in a variety of shapes and sizes, from small tablet screens to laptops and large desktop monitors and multi-monitor setups. They will also scale to different pixel densities; from that of the typical tablet to new high-definition tablets. The following principles guided us in our design process:

1. Offer customers a broad choice of form factors while providing a polished, consistent, and predictable user experience.
2. Enable developers to easily build apps that look great on all form factors in the Windows ecosystem.

With Windows, you can choose a PC that works for you, with a screen that best meets your needs, preferences, or style. For example, a student might buy a touch-enabled laptop with a big screen because they want to be able to write papers but still have fun watching movies or playing games on a touch-screen. Families might opt for an all-in-one desktop with a huge touch screen to view and organize all of the family photos. An accountant with a long commute might pick up a small tablet that easily fits in her bag to surf the web or catch up on her reading during her train ride to and from work. A professional architect or financial trader might have three screens in a mixed portrait and landscape configuration, with one touch screen in the mix.

Windows 8 will power all these PCs and experiences, and as people transition between different sized screens in their day-to-day lives, they will be greeted with a consistent and familiar experience. This breadth of hardware choice is unique to Windows and is central to how we see Windows evolving.

In Windows 8, apps power the user experience, so providing a development platform that makes it easy for developers to create a beautiful user interface that scales to all screens is paramount. For this primary reason, Windows 8 was engineered from the ground up to be a platform for making great apps that work on a variety of screens.



Device diversity

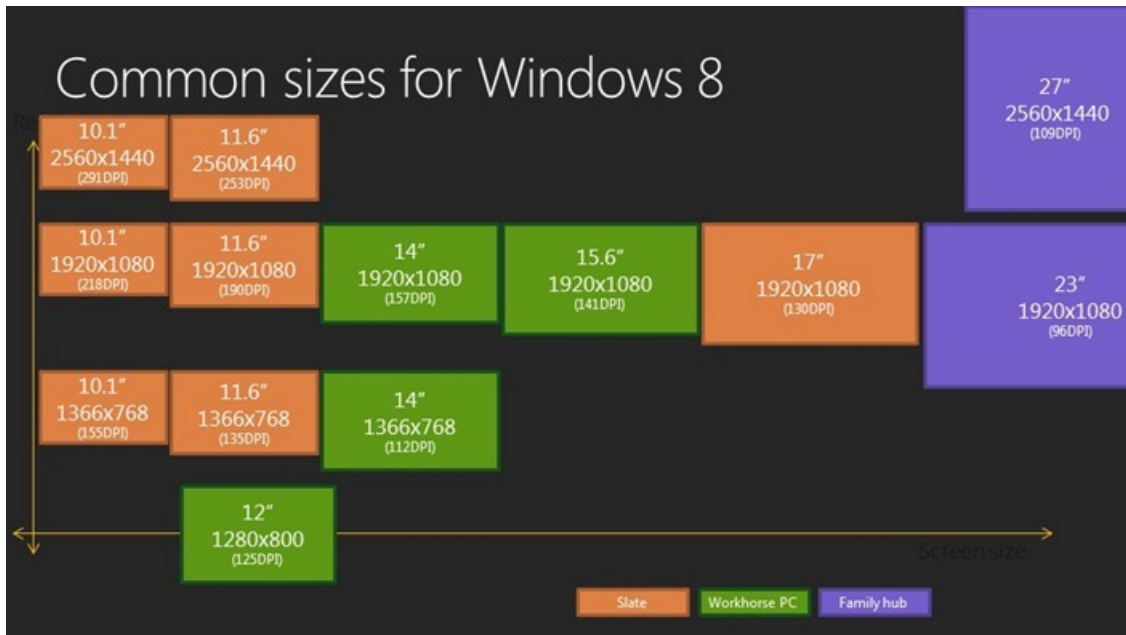
Looking at the breadth of devices that will run Windows 8, we can classify their screens in several ways.

- **Screen size:** There will be PCs with different screen sizes, from the smaller screens on tablets, to medium sized laptops, and large desktops and all-in-ones. These screens will also come in different shapes or aspect ratios.
- **Screen resolution:** Screens will have an increasing number of pixels on screen, or screen resolution. In general, the larger the screen, the higher the screen resolution, but this isn't always the case.
- **Pixel density:** Screens will also have different pixel densities, which is the number of pixels within a physical area, or dots per inch (DPI). The pixel density increases as the screen resolution increases, but the screen size remains constant.

Screen size, resolution, and pixel density were each considered carefully when designing Windows 8 for users and developers. When talking about screens, it is very important to be clear about the variable or dimension being talked about. For example, a 13” screen might be running at any number of resolutions (which means any number of pixel densities) and might have one of several different aspect ratios.

This graphic shows a sample of the diversity of common wide-screen aspect ratios and screen sizes that Windows 8 can run on. Windows will support just about any screen dimension so long as the graphics driver and hardware combination provide the correct information to Windows. In addition, some screens will scale to different aspect ratios via cropping and/or stretching. And although we indicate slate or laptop in the diagram

below, please keep in mind that these are “fuzzy” boundaries that are getting more fuzzy all the time.



Minimum resolution

I've seen a few blog comments that ask specifically about minimum resolution, for example on [Designing the Start screen](#) in October 2011, @wolf asked:

“A better idea would force all developers to make sure all Metro app[s] [are designed for] a minimal screen size of 800x600. Limiting Metro apps to only 1024x768 will cut out all netbook users as well as hurt the Windows App Store.”

We chose a minimum screen resolution of 1024x768 in order to make it as simple as possible for developers to create great apps that work on all the different screens that are available now and in the future. A minimum resolution provides a necessary starting place for developers, who can use it as a baseline to ensure that all of the navigation, controls, and content fit on screen. As we worked on different design layouts for apps, we found that the higher the minimum resolution, the richer and more tailored the app could be. We wanted developers to be able to tailor and refine their layouts to make use of every available pixel on 1024x768, without having to compromise the layout for a smaller resolution.



Windows 8 has a minimum resolution to allow developers to create rich layouts that make the best use of space at 1024x768

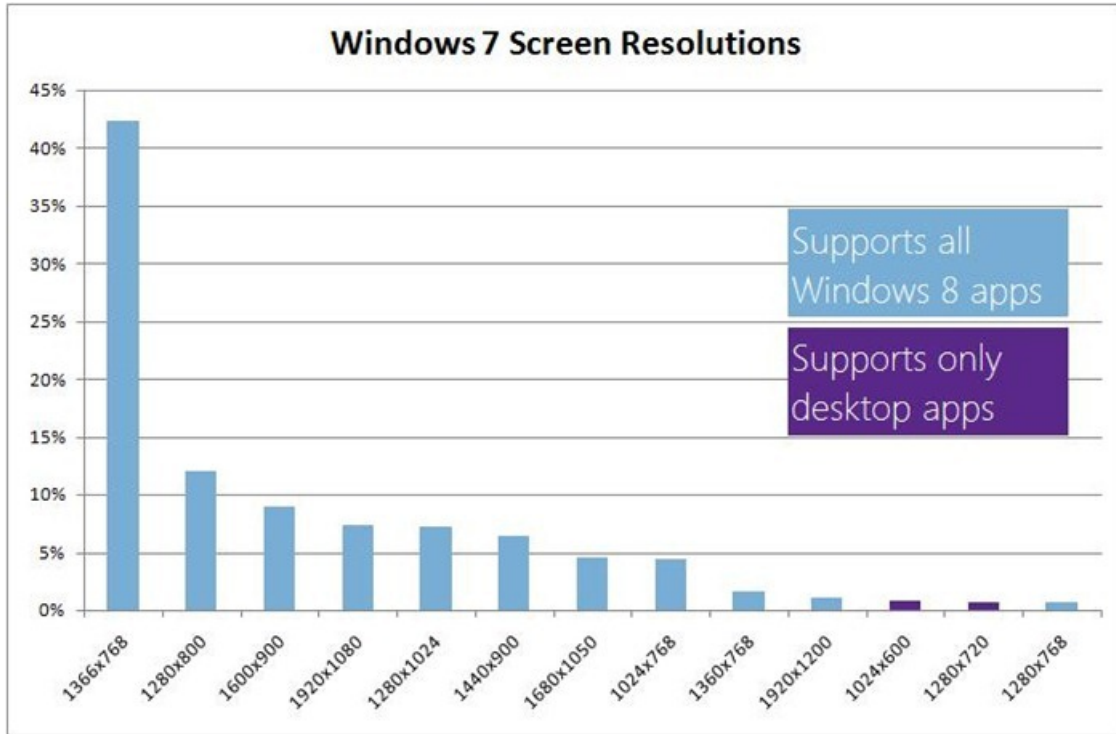
Why choose 1024x768 as a minimum?

We chose 1024x768 as a minimum for Metro style apps for three reasons.

- It is large enough to support the rich and beautiful layouts that we expect to see with Metro style apps. Lower resolutions, like 800x600 for

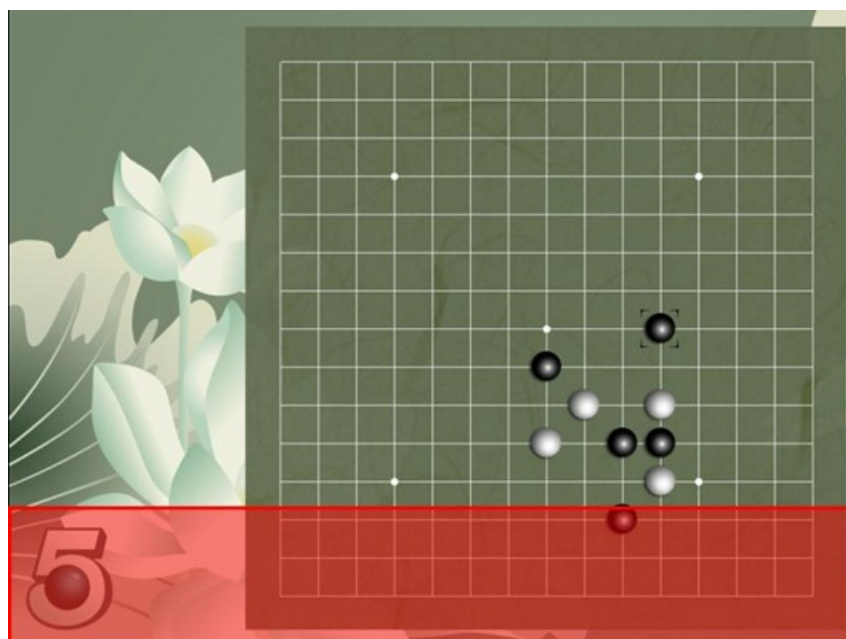
example, require simpler more basic layouts with less content.

- Websites are typically designed for 1024x768 as the minimum (or only) resolution, and web developers are used to targeting this resolution.
- Looking at the data about devices in the marketplace today, we see that only 1.2% of active Windows 7 users have screens with a resolution of less than 1024x768. When designing a new platform that supports the devices of today and tomorrow (with undoubtedly higher resolutions) we optimized for the majority of today's screens (i.e. 98.8%) without sacrificing the experience and complicating the developer story for legacy screens. In addition, the runrate of new PCs with screen sizes of 1024x600 and 1280x720 has dramatically fallen and, to the best of our knowledge, almost no new mainstream PCs are being manufactured with this resolution. We are aware of purpose-built machines that run at lower resolutions, which are built for specialized desktop apps as well. While many run virtual machines, VMs can easily support 1024x768 even though many default to lower resolution.



A world without a minimum

Some people have asked why we enforce the minimum resolution instead of just communicating it as a loosely supported recommendation. Enforcing the requirement simplifies the lives of developers as they never have to take these lower screen resolutions into consideration—they can just rule them out. If an app isn't designed with consideration for lower resolutions, some layouts could truncate, wrap, or break in unpredictable ways. Developers would not be able to confidently build apps to look good on all devices that Windows 8 supports. If we were to have a loose requirement, some developers might build and test for these lower resolutions, while others might not, yielding a fractured ecosystem where developers start targeting specific devices instead of the platform as a whole. Also, developers might target the least common denominator and pick the lowest possible resolution, which in turn would be detrimental to the user experience and quality of the apps.



The layout of this game would be truncated at the bottom if allowed to run on 1024x600

Minimum resolution and snap

The resolution that supports all the features of Windows 8, including [multitasking with snap](#), is 1366x768. We chose this resolution as it has enough horizontal pixels to fit the 320px width of a snapped app, next to a main app with a 1024px width. The specs of the Samsung tablet that we unveiled at the [//build/ conference](#) are 11.6-inches with a 1366x768 resolution (the Samsung Series 7 tablet in market today). These specs are the minimum screen resolution that supports all the features of Windows 8 on a useful physical size.



The snap view is always a fixed 320px wide, which allows developers to refine and create a targeted view for this size. A width of 320px is a common and familiar size that developers are already designing for on various phone platforms.

Some people have asked why we don't allow for the snap view to be arbitrarily sized, or offer a variety of different multitasking sizes. Supporting arbitrary sizes for this small of a layout can significantly increase the complexity of building an app, and would require a lot of additional work and complexity from the developer.

Although the width of a snapped app remains fixed, the vertical space increases to fit the screen, so on larger screens you won't have to scroll as much. The [//build/ talk 8 traits of great Metro style apps](#) provides many great snapped layout examples. We will discuss snap and multitasking more in a future post.

Below are several examples, with the snapped app layout on the left, and the primary app layout on the right.



Is there a maximum resolution?

You may be wondering why there isn't a maximum resolution. With higher resolutions there is more space, so the layout is really never broken or

truncated on higher resolution screens. You can run Metro style apps on a screen as big as 30" with a resolution of 2560x1600! But although apps aren't broken when they have more space, developers should give some consideration to these larger resolution screens, so that they make use of the space in a way that keeps their apps looking beautiful.

Larger screen sizes

On larger screens like desktop monitors, people generally expect to fit more content on the screens—as the screen size increases, screens have more pixels. The below diagrams demonstrate how, when the screen size and number of pixels increase, the number of objects of the same size on screen also increases. On the small screen below we can fit about 40 orange squares, and on the larger screen we can fit 84 squares of the same size.



Larger screens generally have more pixels and can therefore show more content

But just because more content *can* fit on screen this doesn't mean every app *will* make use of this space. If an app is designed with fixed dimensions or a specific form factor in mind, larger monitors may display a large empty region, as in the example below. This is not a good experience, as some have commented.

Regardless of your large screen resolution, today most websites are not particularly well tailored for large screens and tend to leave lots of space (many users prefer to zoom in on the text using the CTRL key and the mouse wheel on large displays, or the keyboard shortcuts CTRL+, CTRL-, CTRL 0). This is the same on the mobile web, when sites are too big to fit on a mobile display. More and more web developers are adapting their content to different form factors by using a combination of form-factor detection and the use of apps.



Without consideration for different screen sizes, many apps would have large empty regions when shown on larger screens

The Windows 8 platform makes building one app that scales to different screen sizes straightforward for the developer by providing built-in layout controls and techniques. Apps in Windows 8 fill the available space by bringing in more content where possible. A developer can easily build the same app to show more content as the screen size changes from a tablet, to a laptop with a bigger screen, all the way up to a desktop monitor. For example, this news app shows more articles on bigger screens. It should be noted that the underlying platform and tools have been developed to provide support for asynchronous programming which also enables “filling” larger displays, and making them just as fast and fluid as smaller displays—there’s no need to block the user while fetching and filling large amounts of content.



11.6" 1366x768



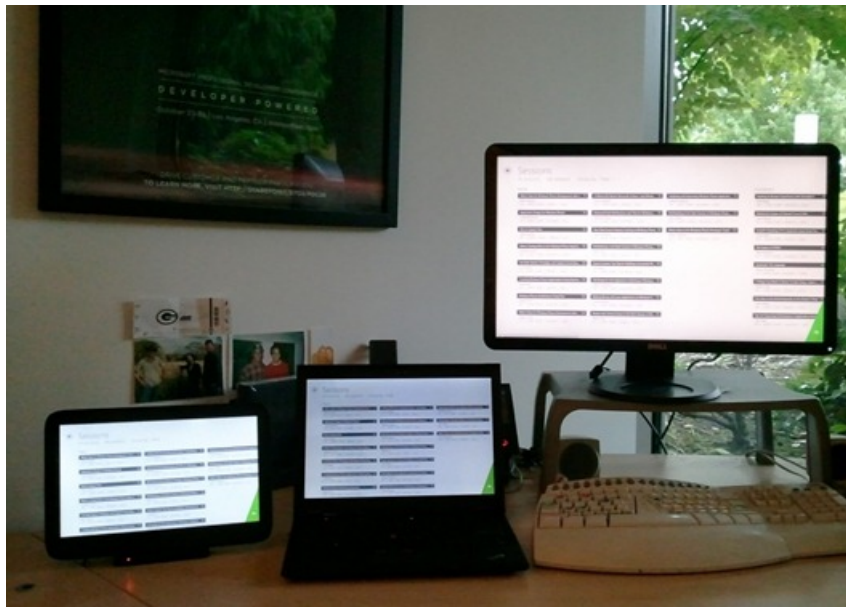
13" 1400x1050



20" 1920x1080

Building apps for larger screen sizes

Windows 8 has been designed to work in a predictable and consistent way for screens of different sizes and shapes across tablets, laptops, and desktop monitors. When a user changes to a different sized screen, it's important that the system and apps make the best use of the screen space that's available to provide the most immersive experience.



With adaptive layouts like this sample app created for the Developer Preview at //build/, users see more content on larger screens

One way that Windows 8 helps app developers to adapt their apps for this variety is through support in the app platform for standards-based adaptive layouts. Building an app layout that looks great on different screens has been a challenge in the past on the web. Rather than inventing a new, proprietary set of layout controls, Windows 8 has built-in support for the familiar adaptive layout techniques from XAML, and for the W3C ratified set of CSS3 features, which were designed specifically to make this easier for web developers.

In HTML, the CSS3 grid, flexible box, and multi-column layouts help developers use screen real estate more effectively across a variety of devices and resolutions.

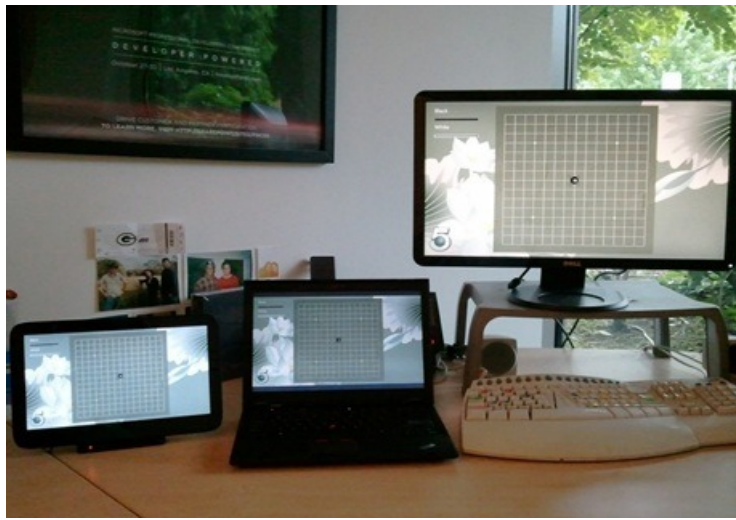
The CSS3 **grid** layout allows a developer to specify the rows and columns of their layout; it is similar to using an HTML table but provides much more control and flexibility. A grid is also great for defining a top-level adaptive layout that is similar to the ones that you see in the Windows 8 UI (like the Start screen and the file picker). You define the rows and columns, and then place your content into the cells of the grid. It is simple to define which cells should adapt and reflow to the screen.

1,1	1,2
1,2	2,2

CSS3 **flexible box** layout allows a developer to distribute margins and whitespace equally and predictably. It's great for laying out individual components and collections like toolbars and image collections.

Finally, CSS3 **multi-column** layout can be used to arrange content into multiple columns on the page, similar to the layout of a newspaper or magazine. All of the templates provided with Visual Studio 11 use these layout constructs and leverage the ListView and other controls to support different sized screens by default. Developers can use the same standards-based layout techniques and controls that help them accommodate different screen sizes to also help them adapt the layout to different orientations and snapped views. All of the layout constructs available in HTML are available for XAML developers as well.

Some apps, particularly games and game-like rendered UI, do not wish to take advantage of more space with higher screen resolutions. For these apps we provide a way to easily scale an app that was designed for 1366x768 to fit any screen. If the aspect ratio doesn't match the content, the system provides theme-able letterboxing regions as well. While this isn't ideal for all UI because it may make things appear quite large on desktop monitors, it does work well for many games and game-like UI that is composed mostly of bitmap graphics. This solution also allows apps to remain immersive on a variety of screens without needing significant work from the developer.



With fixed layouts like this Sinarow game, users see the game bigger on larger screens

We believe it is important for app developers to be able to choose which layout technique—adaptive or scaled to fit—makes the most sense for them, depending on their content and their workflow. If all apps were adaptive, it would be difficult to build game-like rendered UI that fills a 23” 1920x1080 screen without huge empty margins. On the other hand, if all apps were scaled to fit, users wouldn’t be able to see more email messages on their 23” 1920x1080 screen. We believe that our solution strikes the right balance, and provides developers with the choice and tools to optimize their apps for different screens based on the scenarios that are most important for them.

You might be wondering why we don’t just let apps arbitrarily resize and not worry about any of this. That is a reasonable question given the history of resizable windows in Windows. In fact, the first version of Windows supported "tiled" Windows and it was not until Windows 2.0 that overlapping windows were supported. We focused on tailored full-screen layouts for Metro style apps for all of the reasons you have just read, along with the desire to have reliable experiences at many resolutions.

This may seem counter-intuitive given our experience in Windows every day. But as we look across many apps and the ever expanding screen sizes available to us all, it has become clear that developers are no longer optimizing for the diversity of screens available. Though most software lists minimum requirements, in practice, we see many errors—with UI that is clipped, awkwardly placed, or just poorly rendered when windows are resized or maximized. We also see assets (icons and UI elements) that do not properly scale to a variety of pixel densities. Even in the designs of the ribbon in Office 2007, much effort went into scaling the ribbon, as you can see in this series of screen shots.

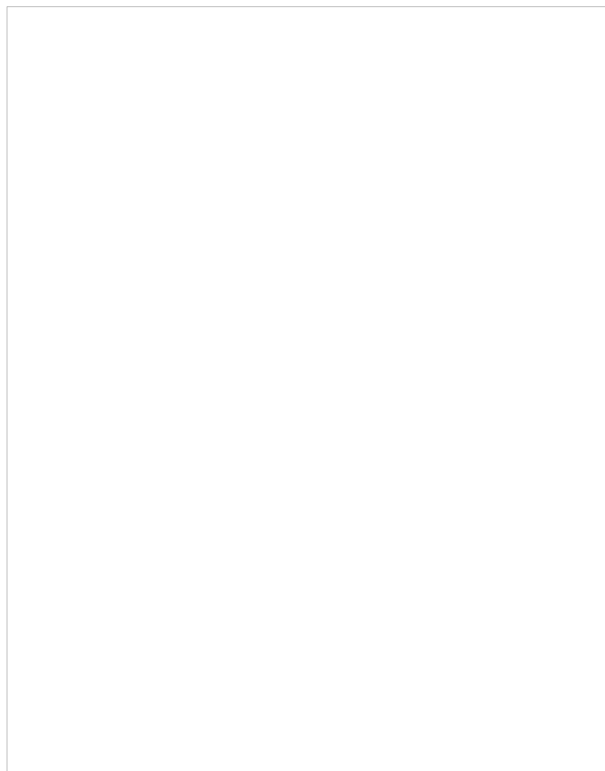


Image from [Scaling up, scaling down](#) on Jensen Harris: An Office User Interface blog

Unfortunately, most applications do not take advantage of controls that are already available (like the Windows ribbon) to successfully scale. As a result, end-users have to learn how big or small to make a window and developers have to deal with bugs and inconsistencies in resolutions that they might not be testing for, since they cannot prepare for all resolutions, aspect ratios, and pixel densities. As developers created their own

layouts, controls, and UI metaphors they also built in assumptions about screen resolution and pixel density required for their code, but rarely enforced these (even today, Windows property sheets clip at below 600 pixels as some have seen with early netbooks or on VMs).

In general, while many reading this blog find arbitrary window sizes something they can manage and arrange, data consistently shows two things. First, on laptops (over 75% of PCs purchased by consumers) most applications are run maximized all the time—this makes sense given the real estate available and the design points of most interfaces and web sites. Second, on large screen displays, most windows are sized to a reasonable number of *rough* dimensions primarily because most programs do not support “infinite” scaling.

We're going to see new user interface approaches and new ways to organize commands. Windows 8 contains a very rich control library and vastly more flexible tools and languages for coding user interface layouts than any previous release. And of course, the Windows desktop is still there (and is improved), where you can continue to work with the capabilities you are used to for the apps you currently use.

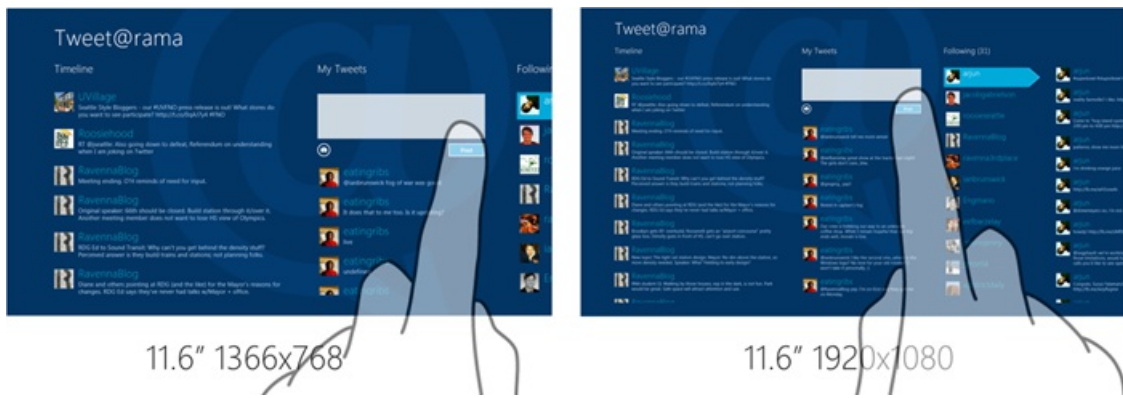
Different pixel densities

Pixel density is a new concept to a lot of people but it is closely tied to our discussion here of screen size and screen resolution. Basically, the pixel density is the number of pixels in a physical area. This is commonly described as dots per inch, or DPI. As the pixel density increases, the physical size of fixed pixels decreases. Some of you may have observed how text can be very small on very high-resolution laptops. Historically many are familiar with “large fonts” or “make text bigger” settings on the desktop to compensate for these physics. Windows 8 takes this to a new level of support for WinRT applications.



On higher pixel density screens, without scaling, physical sizes are smaller.

Most of us are used to fairly low-pixel densities in laptop and desktop monitors; for example, a common laptop with a 13" screen size and a resolution of 1280x800 has 116 DPI. Because of the active ecosystem bringing different displays to market we are seeing incredible advances in the pixel densities of screens on the market. Many Windows 8 tablet PCs will have pixel densities of at least 135 DPI - much higher than many of us are used to. Of course we've seen the introduction of HD tablets with Full HD 1920x1080 resolution on an 11.6" screen, with a pixel density of 190 DPI or quad-XGA tablets with 2560x1440 on the same 11.6" screen; that's a pixel density of 253 DPI. Pixel densities can increase even more on lesser aspect ratios and smaller screens as we see in the new iPad. As the pixel density increases, the physical size of objects on screen gets smaller. If Windows wasn't built to accommodate different pixel densities, objects on screen would be too small to easily tap or read on these tablets.



Without scaling, objects are too small to tap easily on a higher pixel-density screen, like the HD tablet on the right.

For those who buy these higher pixel-density screens, we want to ensure that their apps, text, and images will look both beautiful and usable on these devices. Early on, we explored continuous scaling to the pixel density, which would maintain the size of an object in inches, but we found that most apps use bitmap images, which could look blurry when scaled up or down to an unpredictable size. Instead, Windows 8 uses predictable scale percentages to ensure that Windows will look great on these devices. There are three scale percentages in Windows 8:

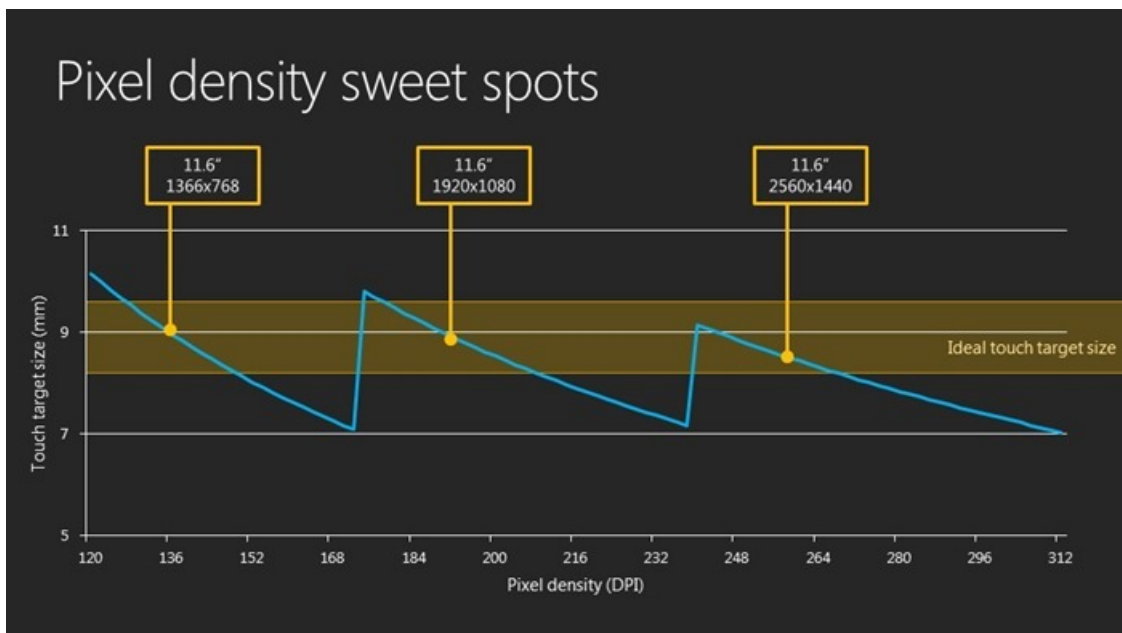
- 100% when no scaling is applied
- 140% for HD tablets
- 180% for quad-XGA tablets



With scaling in Windows 8, physical sizes are maintained on high pixel density devices, and text and content on screen is crisper.

The percentages are optimized for real devices in the ecosystem. 140% and 180% may seem like odd scale percentage choices, but they make sense when you think about how this will work on real hardware.

For example, the 140% scale is optimized for 1920x1080 HD tablets, which has 140% of the baseline tablet resolution of 1366x768. These optimized scale factors maintain consistent layouts between the baseline tablet and the HD tablet, because the effective resolution is the same on the two devices. Each scale percentage was chosen to ensure that a layout that was designed for 100% 1366x768 tablets has content that is the same physical size and the same layout on 140% HD tablets or 180% quad-XGA tablets.



The scale percentages in Windows have been designed to maintain touch targets and layouts, while optimizing for real tablets coming on the market in the near future.

Some might be curious about the new iPad screen. For this screen, Apple has chosen a scale factor of 200%. The new screen has twice the pixel density (132 PPI to 264 PPI)* on the same size screen. Because iOS and developers only need to support the predefined resolutions, they only need to design for this one additional scaling factor. In the case of iPad 2 compared to new iPad the 200% scaling factor means that what you see on 1024x768 is exactly what you see on the new resolution, only sharper because more pixels are used (as in the image of the app above). Additionally, on higher pixel-density screens like the new iPad, developers for games and other performance-critical apps may decide the right balance between letterboxing and running at a lower fidelity to deliver the best experience (frame rate, for example).

Scaling is invisible to the user and Windows implements it automatically based on screen dimensions, without the need for intervention from the user, IT administrator or OEM vendor. Developers just need to make sure images look great on each of the scale percentages. Because these scaling percentages are predictable, developers who provide images for each percentage can easily avoid any blurriness or artifacts due to image stretching.

Pixel density offers another variable where the existing paradigms of toolbars and menus are becoming increasingly burdensome to use. "Hacks" such as large fonts or tricking the system into using a different DPI are just that—hacks. As anyone who has used a high-DPI screen can tell you, existing applications and the UI paradigms simply don't function, and become unusable. A typical example is when a common toolbar button becomes a diminishingly small square, and menu heights and text become too small to read and navigate. Obviously personal preference plays a role, and the ability to tweak the system can help, but neither of these is a reliable way to make sure Windows is usable on a new generation of hardware.

Windows 8 has been designed to provide developers with the easiest way for to reliably build software that works on the widest variety of hardware, and top provide consumers with the most consistently rich experience using that software. It is important not to look at this in isolation as "no more resizable windows," but rather as part of a larger effort to provide a wider choice of screen sizes, resolutions, and densities, where developers can know their apps will work and consumers can be sure that their apps are compatible with their hardware. We do this so you don't have to compromise by using software that isn't fully functional or only getting to choose among a small set of screen sizes (and price points, power consumption, etc.)

Building apps for higher pixel densities

Windows 8 also makes it simple to develop apps that work across different pixel densities. First of all, no manual work needs to be done in order for the app to scale. Unlike previous releases, you won't need to do any work to make your apps DPI-aware; there are frameworks in place to scale the app for you. Just by using web-standard CSS pixel units or a XAML layout, app layouts will scale proportionally. When an app is scaled up, images are stretched and could get blurry, but Windows 8 makes it easy for developers to keep these images looking crisp and beautiful.

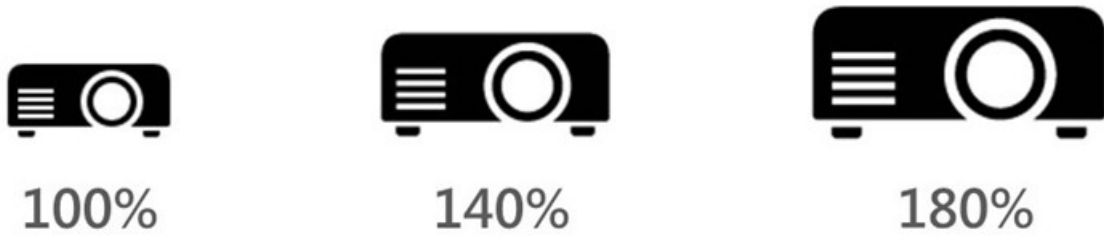


Stretched bitmap



Crisp 180% bitmap

Windows 8, the platform natively supports vector graphics. Any images exported as SVG (Scalable Vector Graphics) or XAML art will scale without getting blurry. Additionally, Windows 8 introduces automatic resource loading so developers can save three versions of images with a naming convention; images that correspond to each of the current scale percentages (100%, 140%, and 180%) load automatically to keep images crisp on high DPI. Developers can also use the CSS3 resolution media query or the system events to reload images at different scales. Windows 8 scaling to pixel density allows developers to achieve a baseline level of quality with little effort, and then tailor their images to look polished and crisp on high pixel density screens.



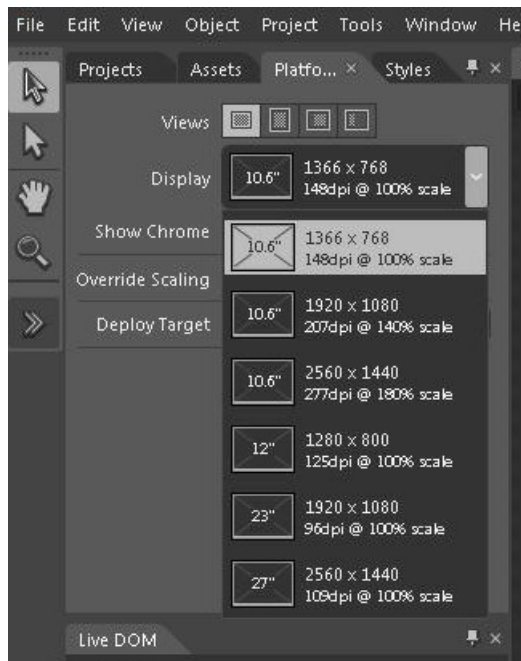
Testing apps on different screens

Even though Windows makes it easy to build beautiful apps that work well on different screens, it is still important to test apps on those different screen sizes. We realize that most people don't have a plethora of devices at their immediate disposal, so we built support for testing apps on different screens into the tools. Visual Studio 11 offers the Windows Simulator, which allows developers to run their apps on a variety of screen sizes, orientations, and pixel densities. Switching to a different screen size is just as easy as choosing an option from a menu.



The Windows Simulator lets you test for different screens

Microsoft Expression Blend 5 offers a platform menu that allows you to design your apps on different screen sizes and pixel densities as you go. The Blend canvas can update dynamically depending on the display dimensions you choose on the platform menu.



Microsoft Expression Blend 5 includes options to design for different screens

Recap

A lot of planning, thinking and development are involved in making sure that Windows 8 scales across different screens and form-factors. For users, Windows 8 offers an experience that is predictable and consistent across devices. On larger screens, they can see more content in each app. On higher pixel-density screens, they get a crisp, premium experience that is easy to read and easy to interact with via touch or keyboard and mouse. For developers, Windows 8 makes it easy to support different screen sizes and pixel densities through standards-based and well known layout techniques, and by automatically scaling to pixel density. All while allowing developers to tailor their experiences to be great on each form factor.

We look forward to you trying Windows 8 on different screens!

Thanks,
David

*A typo in the second PPI number was corrected on 3/22/2012. Apologies for the error.

Touch hardware and Windows 8

Steven Sinofsky | [2012-03-28T14:00:00+00:00](#)

*Last September we blogged about [experiencing Windows 8 touch on Windows 7 hardware](#), introducing the story of touchscreen hardware, how it is evolving, and what we expect Windows 8 will bring to the ecosystem of touch. We discussed how our engineering efforts (software and hardware) are driven by key user experiences, how these experiences play a big part in how we evaluate Windows 8 hardware, and how we communicate with hardware partners. Since that post, we've been working closely with our partners to build Windows 8 PCs. With the Consumer Preview, we want to update you on where we're at. **This post was authored by Jerry Koh, a group program manager, and Jeff Piira, a test manager, both on our Human Interaction Platform team.***

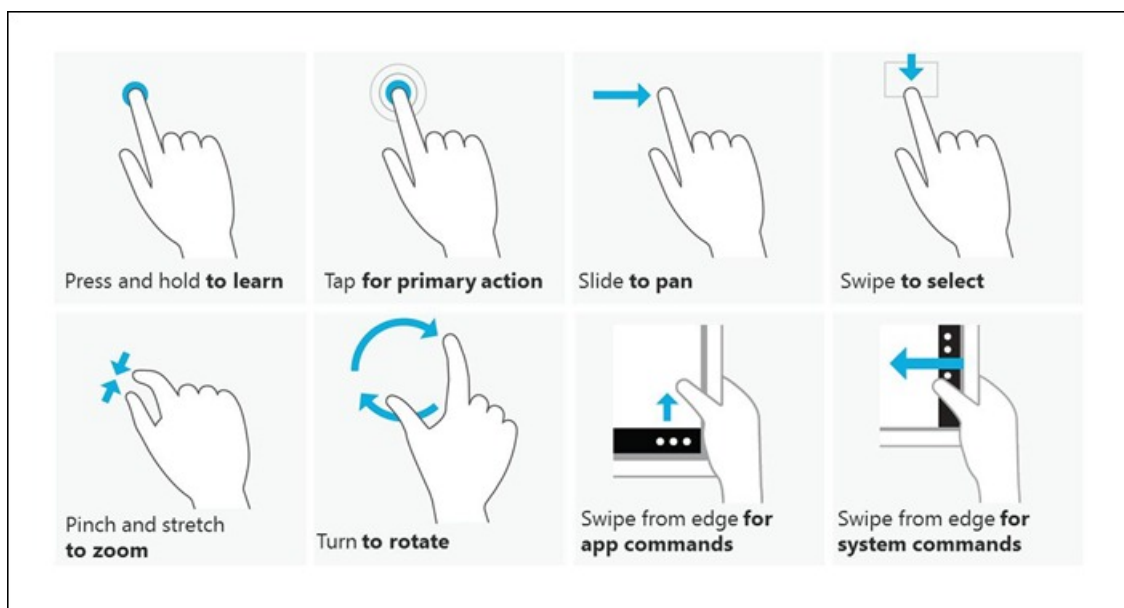
—Steven

The Windows team has continued to work in lock step with external hardware partners to fully embrace the experience we want for Windows 8. New Windows 8 PCs are coming, and while that is not a topic for this post, we at Microsoft are excited with what our hardware partners have in store for you.

It's worth reinforcing that **Windows 8 will run on the hardware available today**, and we are committed to making sure that happens. So you should feel confidence in installing the Consumer Preview on the machines that you own today. However, as much as we value compatibility, we also have to balance this with making Windows 8 really shine on new Windows 8 PCs. We'd like to provide you with some perspective on our efforts and how we will achieve this.

Making sure Windows 8 works on your Windows 7 PC

At the //build/ conference we [introduced a set of touch interactions](#) that make up the Windows 8 touch language. These core interactions form the basis of the Windows 8 user interface, and are reused heavily in the application frameworks within our [common controls and samples](#). The primary goal of our touch language is to promote ease-of-use and ensure user confidence. By confidence, we mean that all touch interactions work consistently and reliably all the time. Developers who consume our controls will automatically feature this language in their applications when they re-use the common controls or use the samples, and in doing so, they also minimize any learning required for users.



The touch language allows us to design a base user experience that is optimized for touch and works well on every PC, whether it was built originally for Windows 7 or for Windows 8. The fundamental gestures require no more than 2 fingers. However it is important to note that 2 fingers can be very limiting for a variety of applications. This is why Windows 8 PCs require digitizers that support a minimum of 5 fingers. The reason we went in this direction is a response to developer feedback. Developers do not want their creativity to be limited, and in particular, they let us know that they want to be free to use whichever multi-finger gestures or controls are useful. They do not want requirements for a minimum number of fingers that may not make sense for their application. As such, we focused on a minimum of 5 fingers to enable scenarios like whole hand interactions (all 5 fingers) or multi-finger/multi-hand scenarios. This will address the feedback, and unlocks opportunities for developers to push the envelope with multi-touch applications. So, while we ensure that the OS works well with a Windows 7 PC, a new Windows 8 PC is going to be much more consistent and predictable both from a user and developer perspective.

New UI concepts in Windows 8 also impact touch hardware design. This is another area where Windows 8 PCs will be more capable than existing Windows 7 PCs. For example, the edge swipe required to reveal the charms and app bars fundamentally changes all the assumptions made on touch hardware. Traditionally, the edges of the screen are where touch sensitivity drops off, and it's a place that hardware manufacturers have traditionally not placed much emphasis on. The center of the screen received all the innovation, while the edges have suffered. If you have

seen or experienced the Windows 8 user experience, the edge swipe is a critical part of using Windows. However, it also has a big role to play in our developer promise, as every pixel used to detect an edge swipe is a pixel taken away from the developer. For Metro style apps, where every pixel belongs to the developer, it is critical that we uphold and deliver on this promise.

We worked closely with our hardware partners to figure out a design that will allow all pixels on a touch screen to be accurate and perform well. There were many challenges here, but we were able to deliver on the promise of Windows 8 PCs that have the ability to trigger the edge swipe without taking any pixels from applications, and with extremely good edge sensitivity using touch—a promise that benefits developers and users alike. To make things work with Windows 7 PCs, we had to go in a different direction. In order to make edge swipe work consistently on Windows 7 PCs, we created a mode where there is a 20-pixel buffer to catch the edge swipe gesture. This allows a majority of PCs to reliably invoke the charms and use Windows 8 effectively. The downside of this buffer is that it takes away some real estate from the application, and from developers.

There is a broad set of Windows 7 PCs available in the market, and while this is a strong testament to the diversity of the Windows Ecosystem (as it offer more choices for our users), it also adds a degree of variance in touchscreen performance that must be accommodated. Here are some other examples of work we did to enable Windows 7 PCs to work well with touch:

- **Making gestures like press and hold and pinch to zoom more forgiving**

On some touch screens, the information reported from the screen is not consistent. We call this “jitter.”, When “jitter” happens, it’s hard for the system to know if the finger is actually moving or stationary. In some instances, a simple gesture like “press and hold” becomes extremely hard to calculate.

- **Determining user intent for sloppy or imprecise touches**

Although larger UI elements help improve touch targeting, we don’t have that luxury within the Windows desktop, especially with existing desktop applications. For this, we developed new ways to remap touch targets using the geometry of the finger, such that it becomes easier to invoke any UI that is within the radius of your finger contact. We will talk more about this feature in a separate blog post.

While you can see that there are a number of places where we’ve done work to accommodate the variance in hardware, there will be some areas where software cannot compensate. We will call out a few of them below. The good news is that in some cases users can learn to overcome these issues; in other cases, the experience will seem slow or imprecise, sometimes requiring you to attempt a gesture more than once before you succeed. We can overcome some of these issues with updated drivers, but this is entirely up to the hardware partners to evaluate and support. Here are some other examples of inconsistencies that we see when comparing touchscreens on some Windows 7 PCs:

- **Individual taps** do not always work, especially when typing quickly in the touch keyboard.

There are generally 3 things that impact this: the touch screen response rate, typing speed, and number of touch points. As you start typing faster on a touch screen, the screen has to match the speed of switching keys. The response rate in a touch screen is usually more optimized for detecting dragging gestures than rapid taps; this will manifest itself as missed taps. When you start typing faster, the chances of having more than 2 simultaneous fingers down also goes up. On systems that do not support more than 2 touches, you will notice missed touches.

- **Swipe to select** is inconsistent on hardware that does not detect small touch deltas fast enough.

It takes a little time for touch to respond to an initial touch. In some cases, the touch screen ignores the first few values of the initial touch, and the system then has difficulty interpreting the swipe correctly.

- **Swipe and slide** can be misinterpreted as a tap, especially on hardware that is not sensitive enough.

This stems from the same issues as above, in which touch screens take some time to respond, and cannot send a consistent stream of data once a swipe or slide begins. In this case, it can result in the system reading the data as taps instead of slides. When this happens, swiping and sliding more slowly may help.

- **Swipe from edge** does not always work, especially with faster swipes.

Although we have the buffer accommodation described above, fast swipes from the edge sometimes suffer from response rate as well. Fingers that come in too fast from the edge don’t get picked up by the touch screen until it is past the buffer. Trying again at a slower speed usually helps here.

Here is a video that shows some examples of how hardware can affect the Windows 8 touch language.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

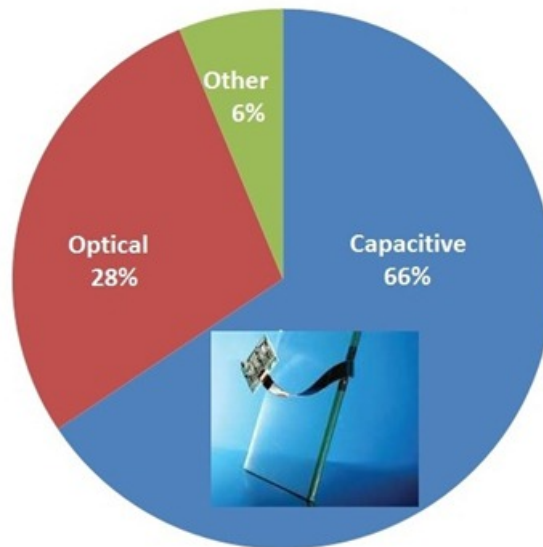
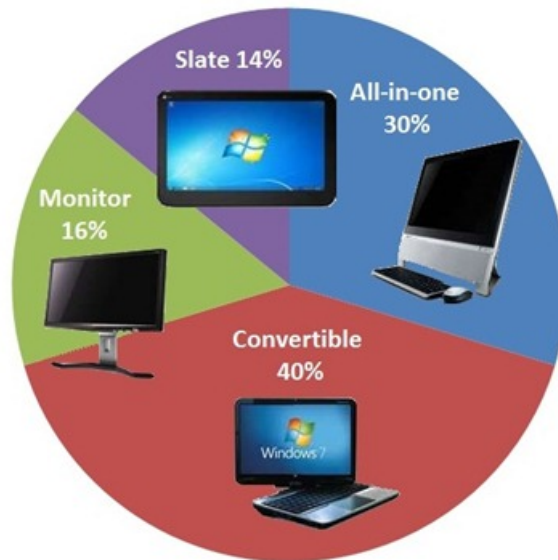
Note that touch variability does not necessarily block usage, since you can adapt the way that you touch or interact with touchscreens to work around different timing or movement thresholds. For example, if the charms don’t appear after an edge swipe, you can try again, but swipe more slowly. However, this variability does tend to make you less confident about using touch on Windows. However, we don’t want app developers to have to accommodate all the variances in touch hardware. Ultimately, we want to keep the promise of consistency, and the promise that applications work on all Windows 8 PCs. This is why we are working hard to ensure that Windows 8 PCs have a consistently good touch experience, and why you will want one of these new PCs when they are released.

Touch hardware testing

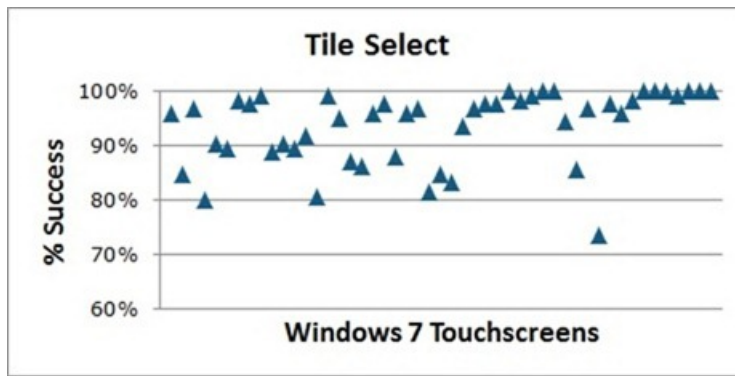
To make sure that the Windows 8 experience works well on your existing Windows 7 PC, we’ve been testing a bunch of them. Listed below are some of the newer Windows 7 systems that are commonly used within the Windows organization. This is not an endorsement, and users of these PCs should not expect official support from PC vendors when installing Windows 8.

- HP Elitebook 2760p convertible
- ASUS EP121 tablet
- Dell Inspiron Duo convertible
- Lenovo x220t convertible
- 3M M2256PW 22" display
- Samsung Series 7 slate

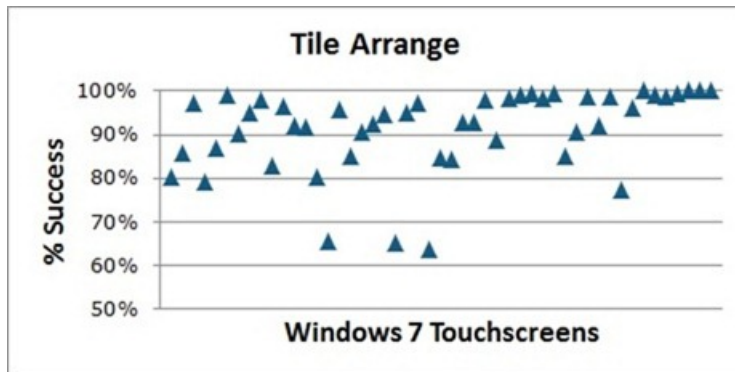
To support our commitment to customers who upgrade, we also [frequently test Windows 8 on a broader set of in-market systems](#). We listed many of these systems in the previous post, and will now share some of the data we collected. Our test team collected data on how the Windows 8 touch interactions perform on 64 different Windows 7-era touch screens. As seen in the pie charts below, the data covers a variety of different form factors and touch sensor technologies.



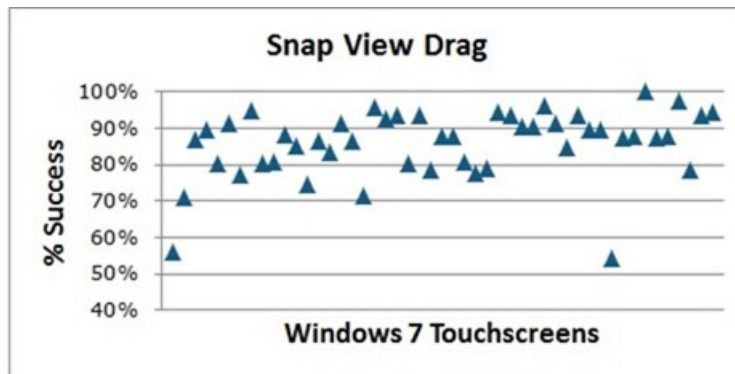
What we found was encouraging: the vast majority of Windows 7 touchscreens can be used with Windows 8. This means that touch drivers continue to load, and you can perform the basic touch interactions in Windows 8 with a reasonable degree of success. But, as described in the previous section, we did see significant variability in how touch interactions were interpreted across different Windows 7 touchscreens. For example, the same swipe gesture can be interpreted as selecting a tile on one touchscreen, as dragging it on another, and as activating (tapping) it on a third screen. The charts below show examples of how successful completion varied between touchscreens when performing Windows 8 interactions for swiping to select a tile, dragging a tile to move, dragging to resize a snapped view, and swiping the screen edge to invoke the charms. 100% means that all attempts succeeded (note that for Windows 8 PCs, we require all these tests to pass with at least 95% success*).



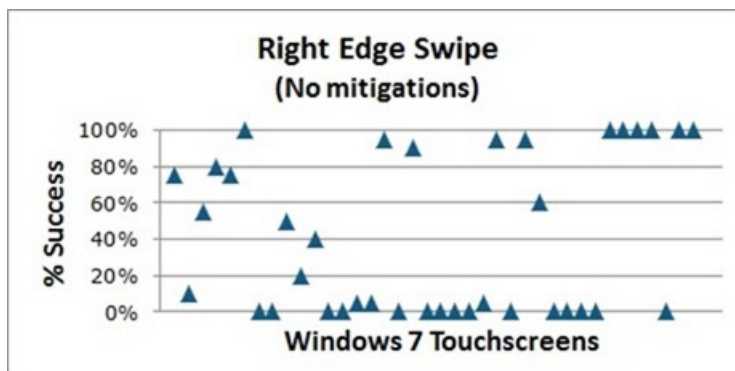
Test: Select a tile in Start
 Expected result: Tile is selected with a single swipe



Test: Move a tile in Start
 Expected result: Tile is moved on first drag attempt



Test: Resize a snapped application
 Expected result: Application is snapped to new size



Test: Swipe the right edge to view Charms
 Expected result: Charms bar appears on first swipe

* pass rates may vary between each test

The road to Windows 8 PCs

Throughout Windows 8 development, we worked closely with external hardware partners to reduce hardware variability and establish consistent Windows 8 requirements. At the time of the //build/ conference, we were still working with partners to establish these requirements in early prototypes. The Samsung slate that was handed out at //build/ was our first attempt to push new requirements into a production system. Those of you who had the opportunity to use the Samsung slate experienced a level of touch quality that is closer to what we expect for Windows 8. You'll also find some of these improvements in the commercialized version of the same slate hardware, which Samsung calls the Series 7 slate. Experiencing Windows 8 on this hardware will give you an idea of what touch will be like on Windows 8 PCs.

We have [published our requirements](#) for new Windows 8 hardware, and we continue to work with touch hardware partners, suppliers, independent hardware vendors, and PC manufacturers to ensure that new devices meet the requirements. Microsoft tests and certifies each new touch device before it can enter the market as a Windows 8 PC. This is how we will ensure consistency and quality in touch hardware for Windows 8. We will talk more about the certification process in a separate blog post.

Experience the Windows 8 Consumer Preview

So if you have a Windows 7 touch-capable PC today, don't hesitate to use the Windows 8 Consumer Preview and take advantage of the Metro style user experience that we've built. The core experience will work well, but you'll need to be mindful of some of the issues we've covered above. On new Windows 8 PCs, these issues won't be present.

If you don't have a touch-capable PC, you can still experience the Metro style UI with a mouse and keyboard. Last and not least, if you develop applications for Windows 8, the developer tools include an emulator that you can use to simulate touch. It will give you a close approximation of how your application will work, and we encourage you to take advantage of it. We hope you agree that we've come a long way in ensuring that Windows 8 has the best touch experience. We're excited to provide a way for you to be more hands-on with your PC.

Thank you!

- Jeff Piira, Test Manager, Human Interaction Platform Team
- Jerry Koh, Group Program Manager, Human Interaction Platform team

Reclaiming memory from Metro style apps

Steven Sinofsky | [2012-04-17T13:15:00+00:00](#)

Modern operating systems take a different view of the resources on the system. Regardless of the form factor, it is important for the OS to more effectively manage resource utilization than in the past. Currently, it is far too easy for a single process to consume available resources (memory, CPU, disk I/O) even when this does not improve the overall performance for end-users. The very role of an OS is to balance resources and make sure you can complete all the things that you want to do on your PC. Much of the manual control present in most OS implementations is designed to work around errant software—software that gets in a state where resource consumption is unbounded. Even if the software is not malicious, which is often the case, the ability to build well-behaved software was limited by the complexity of resource allocation APIs. The modern API set in WinRT is designed for programmers to more easily build software that gets the work done while not “taking over” your PC. This is something that makes all PC form factors and all software better.

*In this post, **group program manager Bill Karagounis** on our Fundamentals team details the behind-the-scenes efforts to reclaim memory even when apps are suspended, and how this all happens without developers needing to worry.*

--Steven

Previous blog posts have discussed the Metro style application model using Windows Runtime. An important attribute of this app model is that apps are suspended when they are no longer visible to the user. Suspending Metro style apps in the background is a good thing, as it conserves CPU for other apps and ensures that background apps don't cause activity that can consume resources, thereby improving the battery life and increasing responsiveness. This is outlined in detail in Sharif Farag and Ben Srour's blog post, [Improving power efficiency for applications](#).

But what about the memory these apps are taking up when suspended? We pointed out earlier that, as a practical matter, the OS will handle this and that your other processes will not feel memory pressure because there are suspended processes. This was an important design consideration. But we know that some of you are still curious how this will all work.

Starting with the Windows 8 Consumer Preview, whenever Windows detects memory pressure on the system, it will repurpose nearly *all* the memory that suspended Metro style apps would otherwise hold onto. Windows 8 can reclaim this memory without having to terminate an app.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

Memory, responsiveness, and Metro style apps

For any type of app (Metro style or desktop), Windows tries to regulate physical memory allocations on behalf of that app, regardless of the memory requests it has made. Windows has always behaved this way to keep memory available in anticipation of future app memory needs. Windows is careful to only allocate physical memory to an app when the app tries to touch it, even if the app had “allocated” it earlier. Windows will also page out or repurpose parts of memory from an app if the pages of memory haven't been touched in a long time.

It is important to understand the goal is not to deny memory to a requesting process, but to delay allocating physical memory as long as possible (until the user actually touches the app) because as with many resources, there is a tendency for software to over-budget. The OS is the place where all these requests come together and it has the information to see that meeting all the fully budgeted requests would ultimately starve every process. Not only would work not complete, but the system will essentially lock up. This happens even if the memory is virtual—instead of running out of RAM, your PC would simply swap pages in and out of disk.

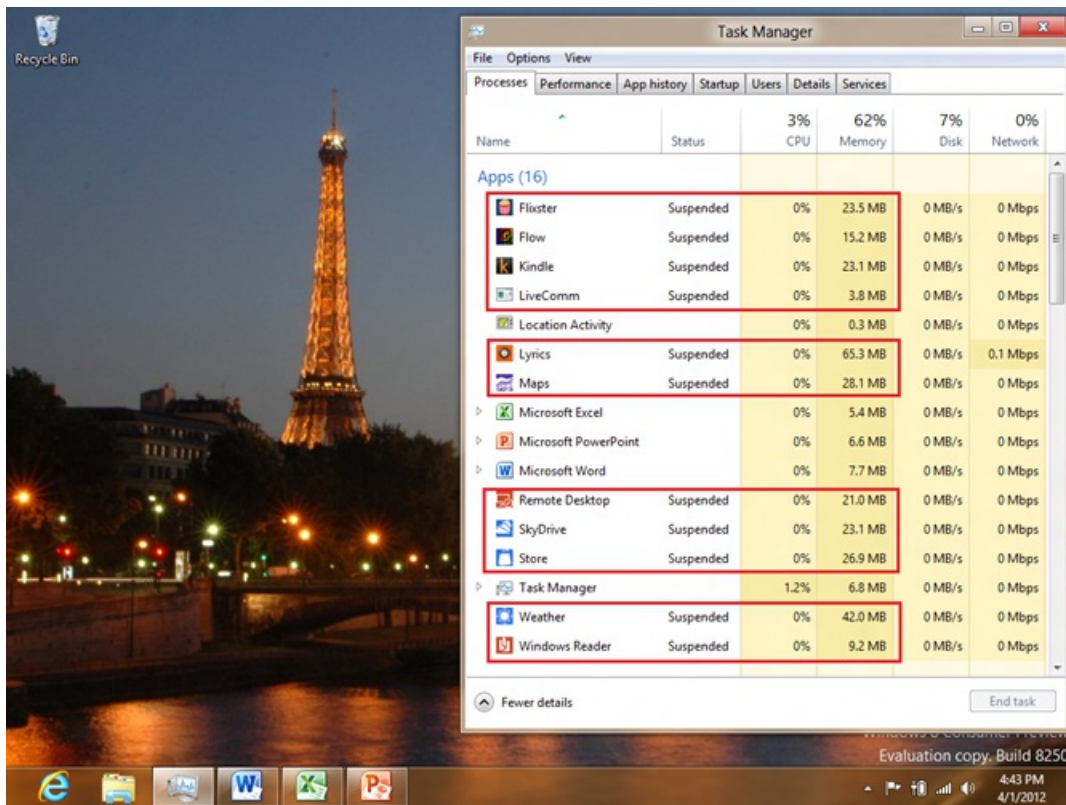
The physical memory given to a process at any point in time is referred to as the “working set” of the process. A *private* working set represents physical memory that is *unique* to a process. Processes also touch other pages of physical memory that are “shared,” which several processes can reference. When you look at the Processes view in Task Manager, the memory for a specific process is actually its current *private* working set. NOTE: For simplicity, when I refer to “working set” in this blog, I mean “private working set.”

When the system starts to run low on available memory, the OS will look in all processes for pages of physical memory that it can repurpose to satisfy other needs in the system, even by paging out memory when necessary. For desktop apps, Windows will try to keep the most important (frequently accessed) pages of memory in the app's working set; this is because desktop apps expect to be able to run code at any time, even when they're in the background. It's a fine balance though: if too many pages of memory were to be removed from a desktop app, it could affect the app's responsiveness due to additional disk I/O (as the app tries to touch memory that has been paged to disk under the covers).

For a deep dive on Windows memory management, see Mark Russinovich's “Mysteries of Windows Memory Management Revealed” talks: [part 1](#) & [part 2](#).

Metro style apps, however, are different from desktop apps, in that they are usually suspended whenever they are no longer in the foreground. When they're suspended, they aren't touching ANY of their memory.

To illustrate, I've highlighted a few suspended Metro style apps with memory held in their working sets in the screenshot below.



Suspended apps holding on to memory

NOTE: In the Consumer Preview build, the Suspended status does not show in the Processes view of Task Manager by default; you have to choose to show it via an option on the View menu.

If there isn't memory pressure in a system, it's actually a good thing (most efficient) to just leave memory in the suspended apps' working sets. But if there is some memory pressure, the fact that these Metro style apps are suspended presents an opportunity to get almost all the memory in these working sets back for other apps, without terminating them.

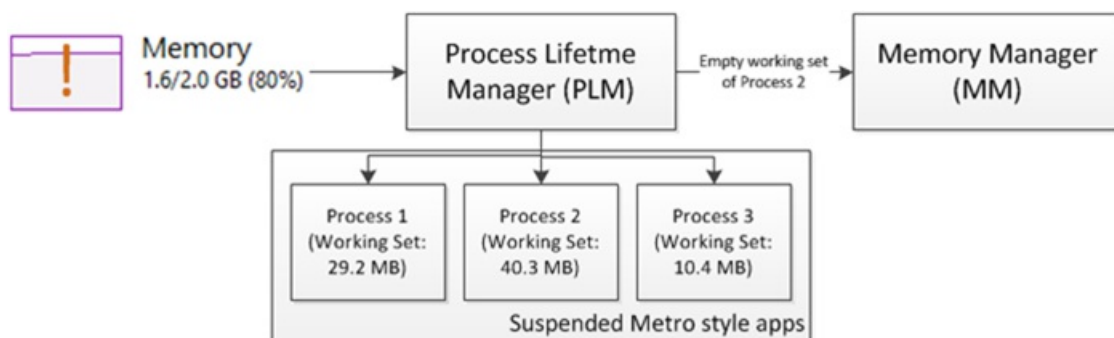
Reclaiming memory from suspended Metro style apps

In Windows 8 Consumer Preview, we can efficiently write the *whole* (private) working set of a suspended Metro style app to disk, in order to gain additional memory when the system detects pressure.

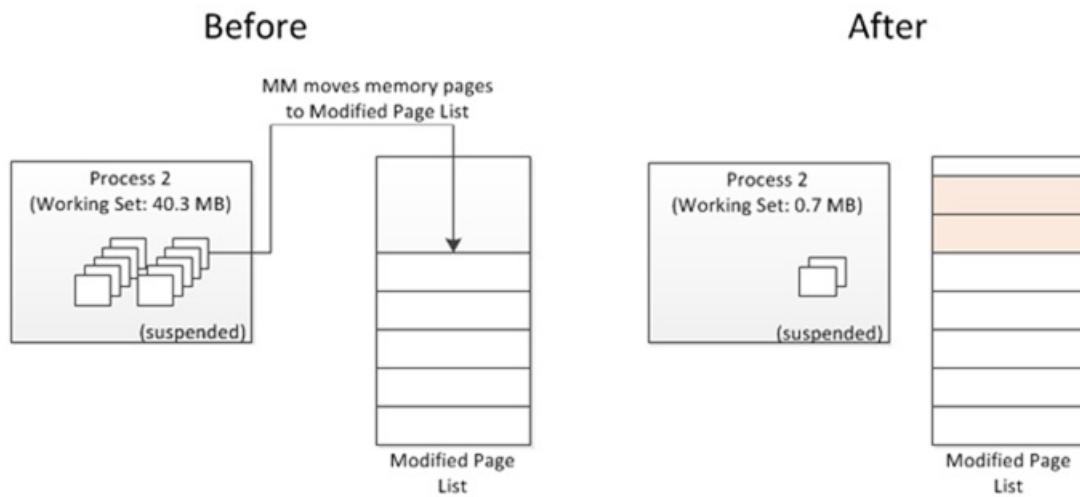
This process is analogous to hibernating a specific app, and then resuming it when the user switches back to the app. We're taking advantage of the suspend/resume mechanism of Metro style apps to empty or re-populate an app's working set.

The sequence of events is described below:

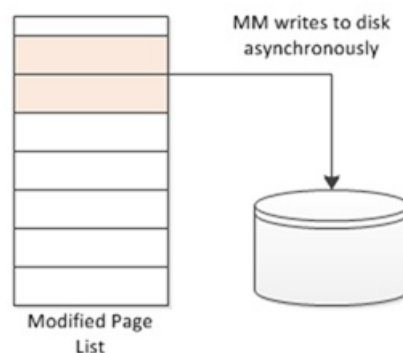
1. The Process Lifetime Manager (PLM) detects memory pressure in the system and asks the Memory Manager (MM) to empty the working set of a specific process that houses a suspended Metro style app.



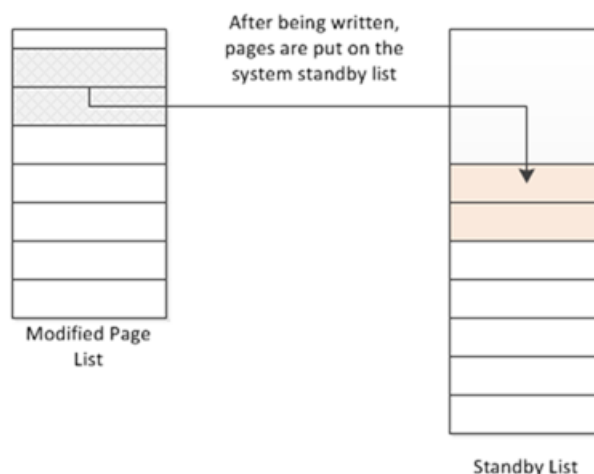
1. MM moves the pages of memory from the working set of the app to the operating system's modified page list (which is a list of memory whose contents are to be written out to disk before being reused).



1. The working set pages on the modified page list are written out asynchronously, as dictated by the usual MM policies (written out opportunistically in the background, writes triggered when under memory pressure).



1. Even after the suspended apps working set is written to disk, the memory pages removed from a process are left intact on the operating system's *standby list*. This is a cache of useful pages of memory that can be repurposed for other apps, if necessary. If these pages are immediately needed again by their original process, they are quickly moved back.



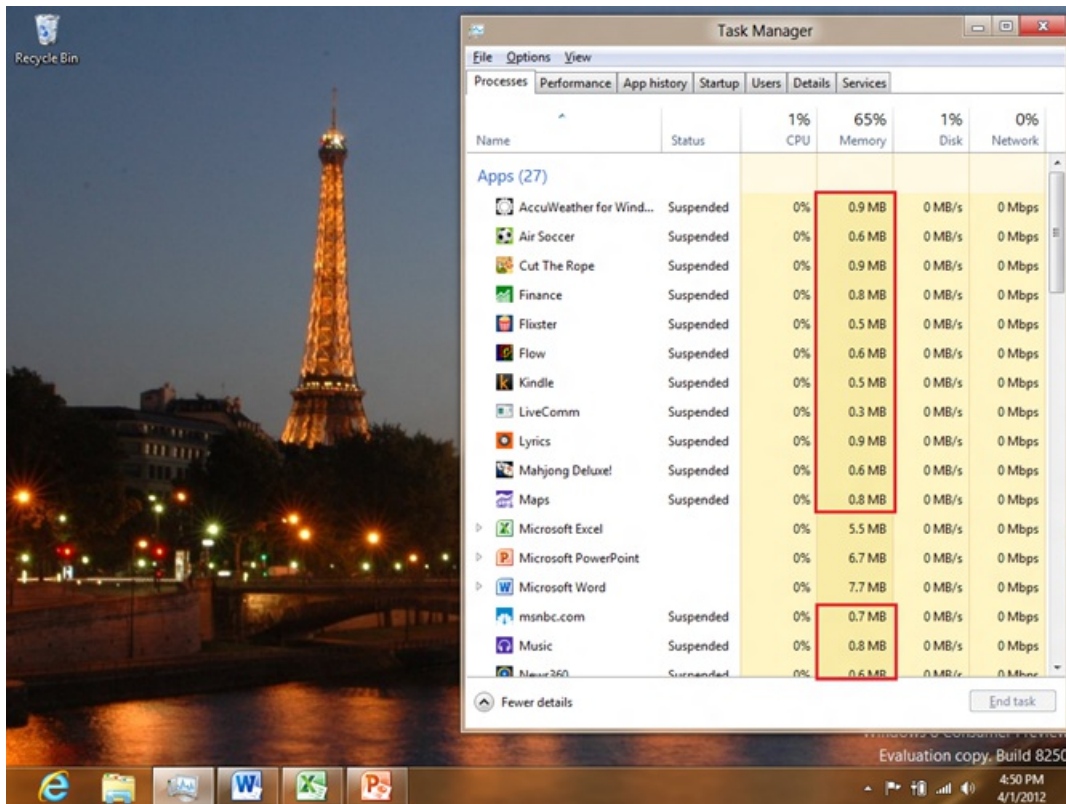
If a user switches back to the app while its working set pages are still in physical memory (on the modified page list or the standby list), it's straightforward: the pages will be added back into the app's process immediately. If they are no longer available, Windows will read in the app's working set from disk in an optimized manner.

“Reclaiming memory” in action

To get a feel for how this works, let's walk through an example with real running code.

The initial state is represented by the screenshot above. I had several Metro style apps running on a PC with 2GB of RAM. The Metro style apps were in the background, and therefore Windows suspended them. I then started opening more apps to drive up memory use on the system and trigger the new functionality.

In the next screenshot, below, you'll notice that I opened some apps in order to introduce additional memory pressure and induce the behavior described above. As you can see, Windows emptied the working sets of the suspended Metro style apps (highlighted).



Working sets of Metro style apps emptied

Let's compare the "before" and "after" working sets of the original Metro style apps (some of the "after" stats cannot be seen because Task Manager ran out of room to show all 27 launched apps):

Metro style app	Working set before (MB)	Working set after (MB)
Flixster	23.5	0.5
Flow	15.2	0.6
Kindle	23.1	0.5
LiveComm	3.8	0.3
Lyrics	65.3	0.9
Maps	28.1	0.8
Remote Desktop	21.0	0.5
SkyDrive	23.1	0.5
Store	26.9	0.6

Weather 42.0 0.8

Windows Reader 9.2 0.4

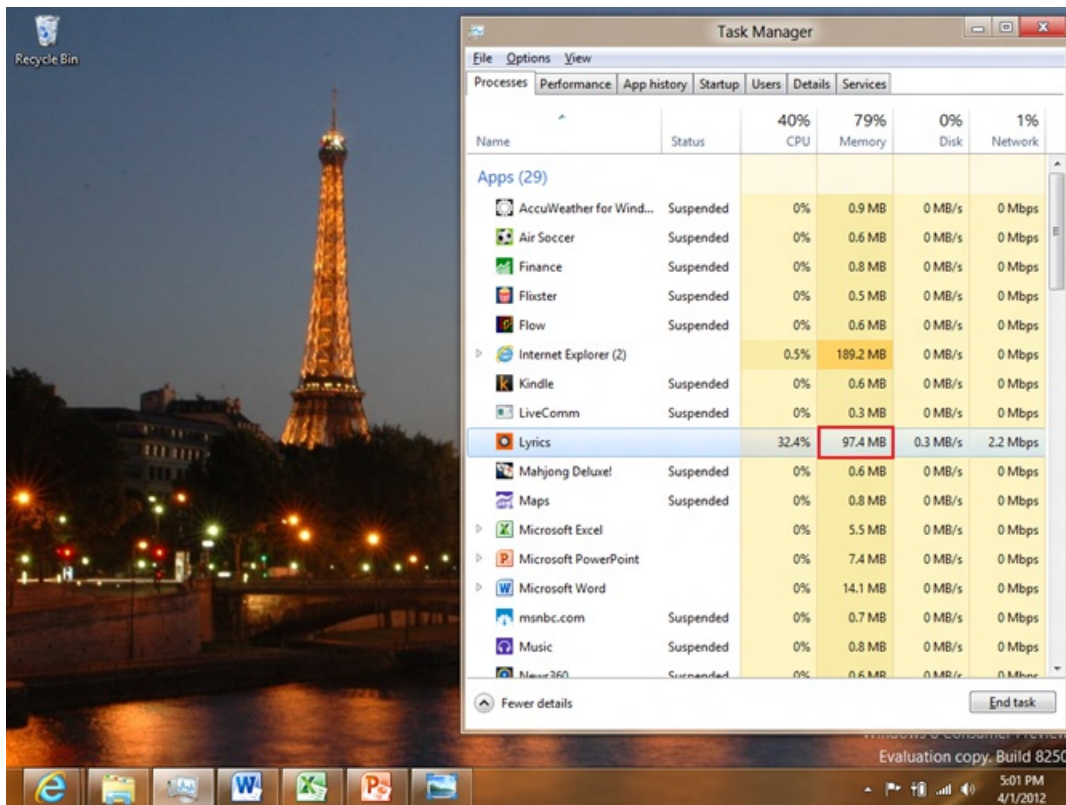
So in this example, we liberated over 250 MB of physical RAM for other apps to use without shutting down the suspended apps.

Responsiveness when reading the working set back in

A test for this new facility is how responsive a suspended app is after its working set contents are emptied and you decide to switch back to the app.

While I was running this test, I used the “Lyrics” app as my indicator of responsiveness. The Lyrics app can display the lyrics of a song as well as play back a music video. When the Lyrics app goes to the background, it gets suspended, which stops the playback.

After driving memory use to the point where working sets were emptied, I opened additional apps and used the system for a while to ensure that a swap back to the app was going to read the working set from disk. I then triggered the switch back to the Lyrics app (below).



Lyrics app's working set is repopulated

The key indicator I was looking for was how long it would take from the time I triggered the switch back to the app to the point where I could hear sound again. On a low-end machine, with a high memory load, it's hard to perceive a difference in responsiveness when switching back to the app whose working set was being read from disk compared to when its working set was still in memory. Your mileage will vary though: the larger the working set, the longer it will take to read-in from disk. We're also continuing to reduce the amount we have to write to disk for Metro style apps and further optimizing this feature.

This functionality is something that everyone with the Consumer Preview release can try for themselves. Just open up a number of Metro style apps and desktop apps to generate some memory pressure, and then switch back to a suspended Metro style app that has had its working set emptied.

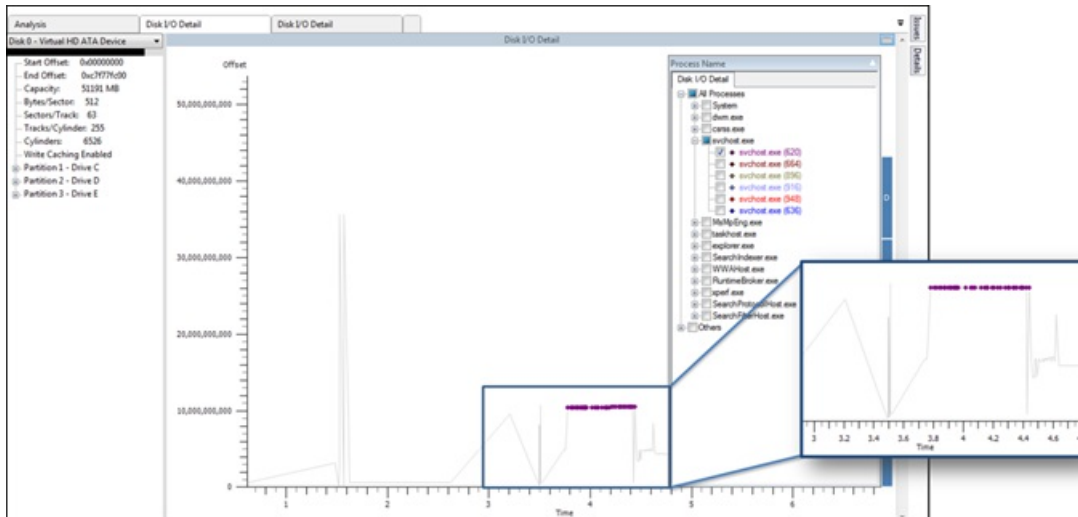
NOTE: Windows will still close Metro style apps if memory gets into the critical range. However, this feature will enable a system to run more applications before getting to that point.

Reading in a working set optimally

To have a responsive experience when you switch back to a suspended app whose working set is on disk, we've optimized how we write the working set in the first place, so that reading it in is as efficient as possible.

When we write out the working set of a suspended Metro style app, the working set pages are being written out to disk sequentially. This allows us to read the data back into a process using a small number of large sequential disk reads. The screenshot below is from our Windows

Performance Analyzer (WPA) tool that comes as part of the [Assessment and Deployment Kit \(ADK\)](#), showing a visual representation of disk reads during this process. I've highlighted the working set "read-in" I/O. You can clearly see the sequential stream as we repopulate an app's working set.



Sequential read I/Os to repopulate an app's working set (WPA Tool)

Reading back sequential data from almost any storage device is really fast. Most rotational disks can achieve between 50-100MB per second. If a storage device is flash-based (like an SSD), you can get upwards of 200MB per second for these reads.

We expect that many apps will take less than a second of I/O to get the working set of a suspended app back into memory.

Conclusion

By definition, all PCs of all form factors have limited physical memory. The challenges of managing memory do not change when you add more physical memory, so long as you are actually running more programs that request more memory.

Several classes of modern software will continue to evolve to use more and more memory – photo editing of giant RAW images, in-memory databases, very large spreadsheets, and so on. Whether these have WinRT-based implementations or desktop implementations, the OS needs to evolve to manage these ever-increasing memory requests. WinRT is an opportunity for programmers to use an API that allows access to all the memory they need while putting the user first in terms of responsiveness and getting work done. I hope you'll try this out this feature in Windows 8 Consumer Preview.

-- Bill Karagounis

Managing "BYO" PCs in the enterprise (including WOA)

Steven Sinofsky | [2012-04-19T10:00:00+00:00](#)

With more and more people providing their own hardware for work, the "bring your own" PC is becoming more commonplace and IT Pros want to have the confidence that they can support their clients who follow this trend. The presence of BYO does not change the need for IT Pros to manage, secure, and remain accountable for the network assets of an organization, and we all know that written policies can only go so far.

*This post focuses on managing WOA PCs, which are designed with this "consumerization of IT" in mind. PCs of all form factors built on x86/64 architecture have the full complement of management tools available to them, especially those supported by third-party code running on the system. Since WOA PCs only support third-party code through the Windows Store and WinRT-based applications, we set out to develop industry-leading management capabilities that support BYO or company-deployed WOA PCs. This post was **authored by Jeffrey Sutherland, a program manager lead in our Management Systems group.***

--Steven

One of the major trends in IT in recent years has been the drive towards "consumerization of IT," which is a term describing how consumer technology, from phones to PCs, is bleeding into business organizations in all forms and fashions. And increasingly, the devices that are showing up are owned by and liable to the employee rather than the organization they work for. We see this most notably in the smartphone device category, but more recently also in tablets or other portable PC form factors that are increasingly showing up in the workplace. As organizations embrace consumerization, IT must consider how much control they can exert over a user's personally-owned device, and how much management is "good enough." These questions were top of mind for us as we began our journey to Windows 8, and particularly, as we built Windows for the ARM processor architecture. Our focus has been on how we can continue to deliver PCs and software that users need, like applications and data-access on any device, with enough IT control to assert that the device is trustworthy, while avoiding any compromise of the user's privacy on their personal device.

In Steven's earlier blog post about [Windows on ARM](#), or WOA, he talked about how the bulk of the Windows experience remains the same on ARM as it is on x86/64, and the products share a significant amount of code. So, while this post will focus primarily on WOA, many of the features discussed are equally applicable to both processor architectures. In addition, this post covers the capabilities on the PC, itself, not the overall management infrastructure and tools used by IT. Also, please keep in mind all the security capabilities built into Windows that come with WOA from the basics of networking all the way through drive encryption.

Line-of-Business applications and the WOA management client

Demand for access to the business apps that users rely on - from email to licensed software from an independent software vendor to home-grown apps developed by IT - is one of the most important use cases for "consumer" devices in the enterprise. We know that developers are going to find it easy and convenient to build elegant Metro style apps that automatically work on any Windows 8 system including WOA, and developers of line-of-business (LOB) apps won't be any different. But many organizations want to directly control and manage access to their internal LOB apps, including the distribution of the app binaries for installation. For these organizations, publishing their LOB apps to the public Windows Store doesn't make sense, since there is no reason to broadcast these applications to others or to have their application deployment managed through the Windows Store process. And access to these resources and the data that they expose requires an assurance to IT that the systems accessing them meet an established bar for security and data protection.

Organizations have been dealing with apps on x86/64 machines for a long time using a variety of tools and methods, including management products like System Center Configuration Manager and Windows Intune. Management of Metro style LOB apps on x86/64 will be able to leverage those same existing tools and methods and only requires that the client be configured to trust the apps that come from a source other than the Windows Store. For more information on the base capabilities of adding and removing Metro style apps on x86/64, see [How to Add and Remove Apps](#). Developing WOA, however, provided us a unique opportunity to architect how LOB apps can be delivered to users in a way that meets the needs of IT while continuing to guarantee a consistent and reliable end-to-end experience over the life of the PC.

For WOA, we have integrated a new management client that can communicate with a **management infrastructure** in the cloud to deliver LOB apps to users. You'll hear more about this management infrastructure at a later date from our friends on the [System Center blog](#), so this post will focus on the benefits and capabilities of the WOA management client itself.

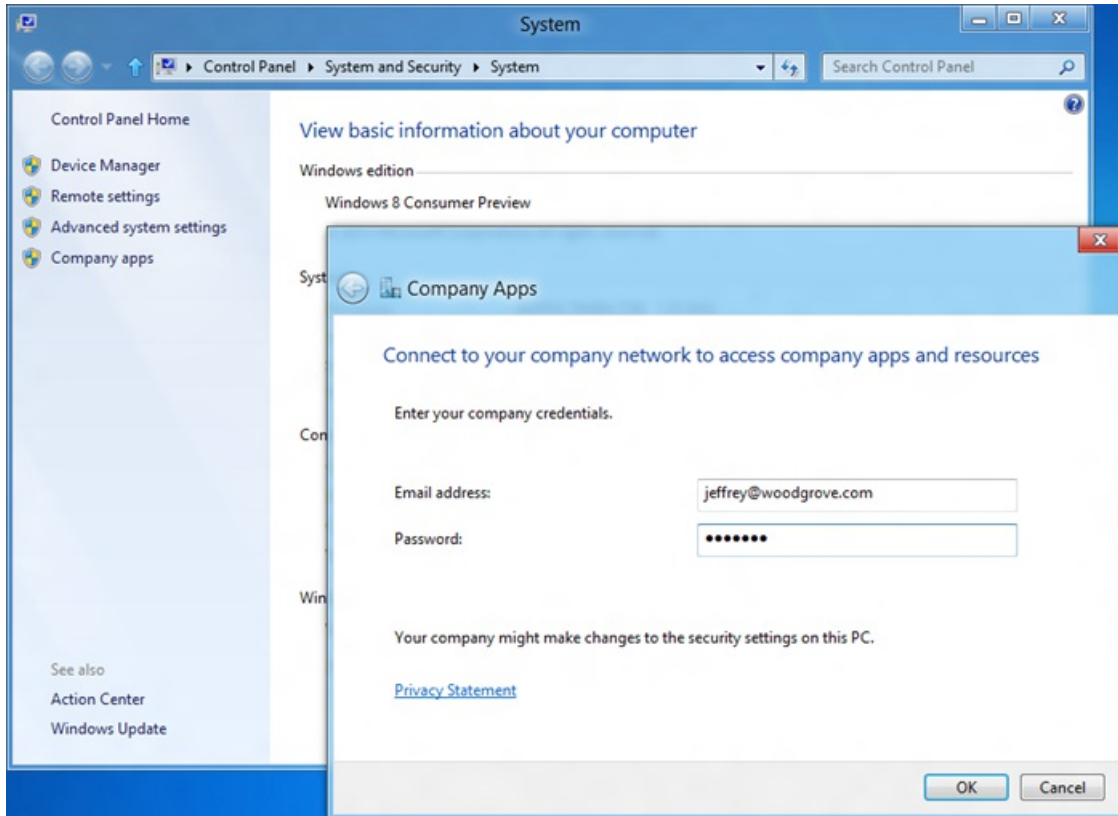
There are actually two parts to the WOA management client: the built-in system component, which we'll call the **agent**; and a Metro-style app, which we'll call the **self-service portal, or SSP**, that the consumer uses to browse for and install LOB apps made available to them. Both parts of the WOA management client are well behaved Windows 8 apps in terms of user experience, power management/battery life, network awareness (for metered networks), and overall functionality.

The agent does most of the heavy lifting on the client. It configures the client to communicate with the organization's management infrastructure; periodically synchronizes with the management infrastructure to check for any updated LOB apps and apply the latest settings policies configured by IT for the device; and handles the actual download and installation of any LOB apps that the user wants to install. Finally, if the user or the administrator chooses to remove the device from the management infrastructure, it clears the configuration of the agent itself and disables any LOB

apps the user installed from the SSP.

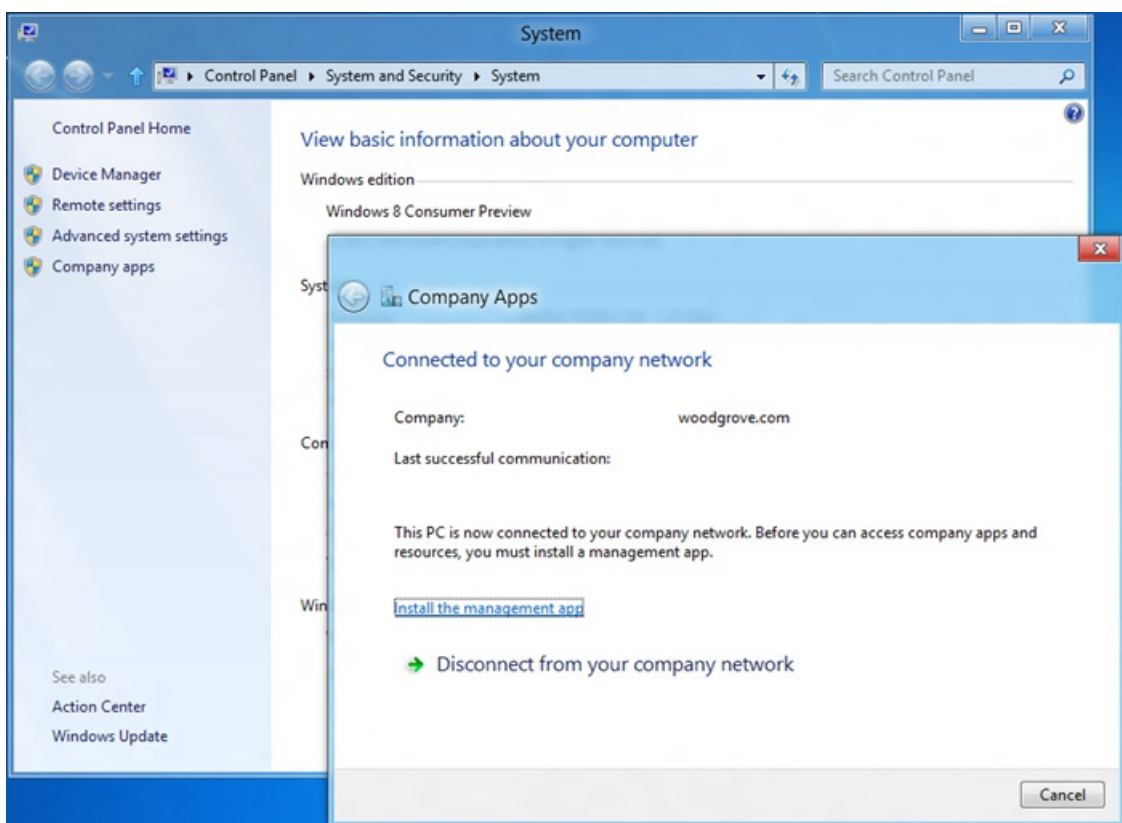
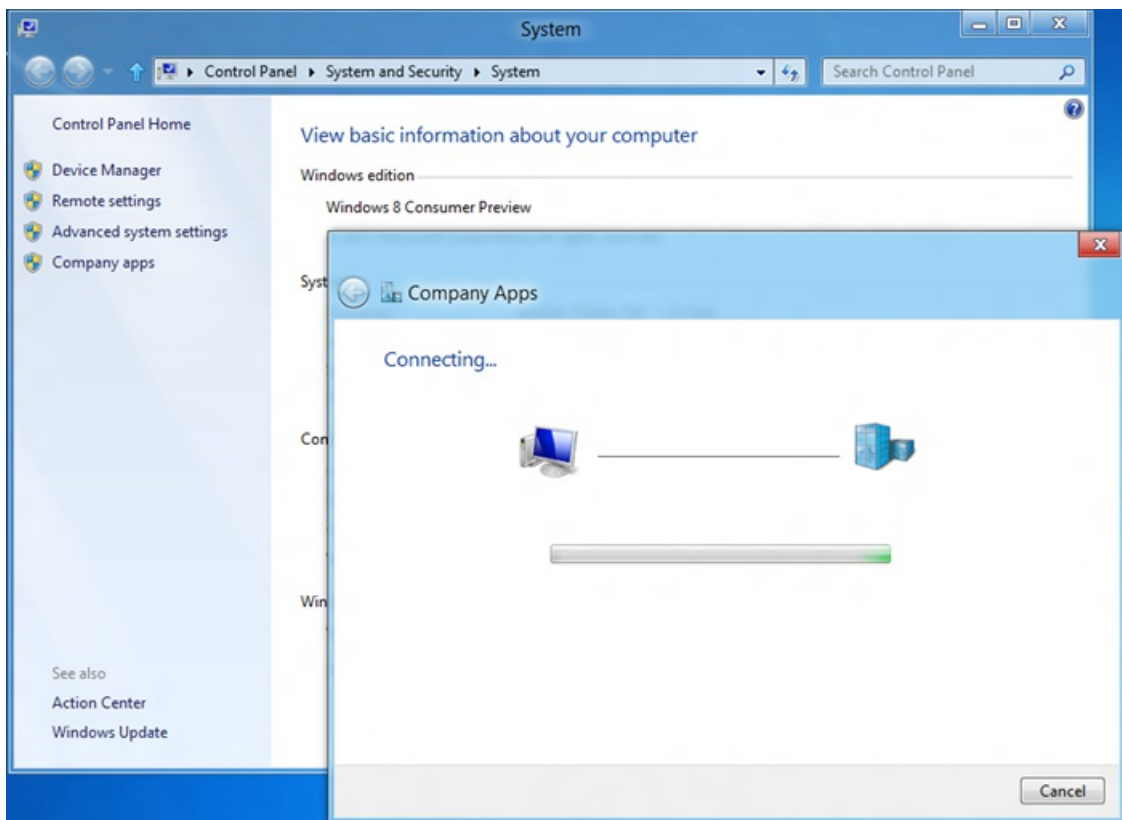
Connecting to the management infrastructure

Let's explore some of these elements in more detail, starting with connecting the client to the management infrastructure. In truth, this step begins with the IT admin who specifies the group of Active Directory (AD) domain users who are authorized to connect devices into the service. The admin also has the option to specify the maximum number of devices allowed per user. For authorized users, the actual steps to connect a device are quite simple. Using a new Control Panel applet on their WOA device, the user supplies their company email address and password, just like they do to set up an Exchange email account. The agent then performs a service lookup to locate the organization's management infrastructure based on the user's email address.



Connecting to your management infrastructure is as easy as entering your company email address and password

Once the agent has found the right address, it establishes a secure connection to the management infrastructure using [SSL Server Authentication](#) and authenticates the user. If the user is successfully authenticated and has been authorized by the admin to connect devices, the service issues a user certificate to the user who initiated the connection. This certificate is sent back to the agent along with the organization root certificate and instructions for the agent, which it uses to configure its ongoing communications with the management infrastructure. All of this happens in a matter of seconds and typically requires no further interaction from the user. Once complete, the user is directed to install the SSP while the agent completes the connection in the background.



Completing the connection

Next, the agent automatically initiates a session with the management infrastructure, using the user certificate to authenticate. This session and any subsequent sessions are performed using SSL Mutual Authentication to ensure the security of the connection. This initial session completes the registration of the device with the service by supplying some basic device information such as the make and model, the OS version, device capabilities, and other hardware information. This allows IT admins to monitor what types of devices are connecting to the organization, so they can improve the apps and services they deliver to users over time.

Following the initial session, the agent initiates communication with the management infrastructure in two circumstances:

- First, as a maintenance task that runs daily at a time that the user can configure on the client. The activities performed during these maintenance sessions focus on reporting updated hardware information to the management infrastructure, applying changes to the settings policies for the device, reporting compliance back to the management infrastructure, and applying app updates to LOB apps, or retrying any

previously failed LOB app installations initiated from the SSP.

- Secondly, the agent will communicate with the management infrastructure anytime the user initiates an app installation from the SSP. These user-initiated sessions are solely focused on app installation and do not perform the maintenance and management activities described in the first case.

Regardless of whether a session is initiated automatically by a scheduled maintenance task or manually by the user, the WOA management client continues to behave well relative to the state of the battery on the device and its current network conditions.

Settings policy management

As already discussed, access to LOB apps typically requires systems to comply with basic security and data protection policies. From the management infrastructure, the IT admin is able to configure a set of policies that we believe are the most critical to give IT the assurances they need without seriously affecting the user's experience with their device, including:

- Allow Convenience Logon
- Maximum Failed Password Attempts
- Maximum Inactivity Time Lock
- Minimum Device Password Complex Characters
- Minimum Password Length
- Password Enabled
- Password Expiration
- Password History

Although our new WOA management client can only connect with a single management infrastructure at a time, we may decide to add other policy sources before we release Windows 8 and so we've architected the policy system to handle this. In the case where more than one policy exists for the same Windows 8 device, the policies will be merged and the most restrictive configuration will be selected for each. This resultant policy will apply to every administrative user on the Windows 8 device and every standard user with an Exchange account configured. Standard users who do not have an Exchange account will not be subject to the policy, but Windows 8 already restricts those users from accessing data in other users' profiles and from privileged locations, thereby automatically protecting your corporate data.

In addition to the configurable policies described above, the agent can also be used to automatically configure a VPN profile for the user, so that WOA devices easily connect to a corporate network without requiring any user action. Finally, the agent can also monitor and report on compliance of WOA devices for the following:

- Drive Encryption Status
- Auto Update Status
- Antivirus Status
- AntiSpyWare Status

Leveraging this compliance information, IT admins can more effectively control access to corporate resources if a device is determined to be at risk. Yet once again, the user's basic experience with the device is left intact and their personal privacy is maintained.

Before we move on, let's consider a couple of the policies listed above and how they practically affect a Windows 8 system. First, we'll look at Allow Convenience Logon. Windows 8 offers users convenience login features, like biometric login or the [picture password feature](#). These options maintain a high level of security for Windows 8 devices, while solving one of the biggest headaches for users and IT alike: forgetting your password. Yet some organizations may require additional time before they are ready to embrace these alternative logon methods, so the Allow Convenience Logon option lets IT manage when to allow convenience logins in their organization.

Secondly, let's look at how drive encryption and Maximum Failed Password Attempts work together. You probably know people who've picked up their smartphone only to find that the device has wiped itself after their young child was playing with it and inadvertently entered the wrong password repeatedly. Nothing so severe will happen with your Windows 8 devices, fortunately. Windows 8 provides strong data protection already out of the box. So, when a user exceeds the password entry threshold, Windows will instead cryptographically lock all encrypted volumes and reboot the device into the Windows 8 recovery console. If your device has been lost or stolen, this effectively renders the device unreadable. But if you're simply the victim of your young son or daughter trying to get to Angry Birds while your device is locked, you can easily recover with the use of a recovery key that Windows 8 can automatically store on your behalf in your SkyDrive account. This way, you are able to get back up and running without enduring a lengthy wait to re-install all of your apps and copy down all of your data.

LOB app management

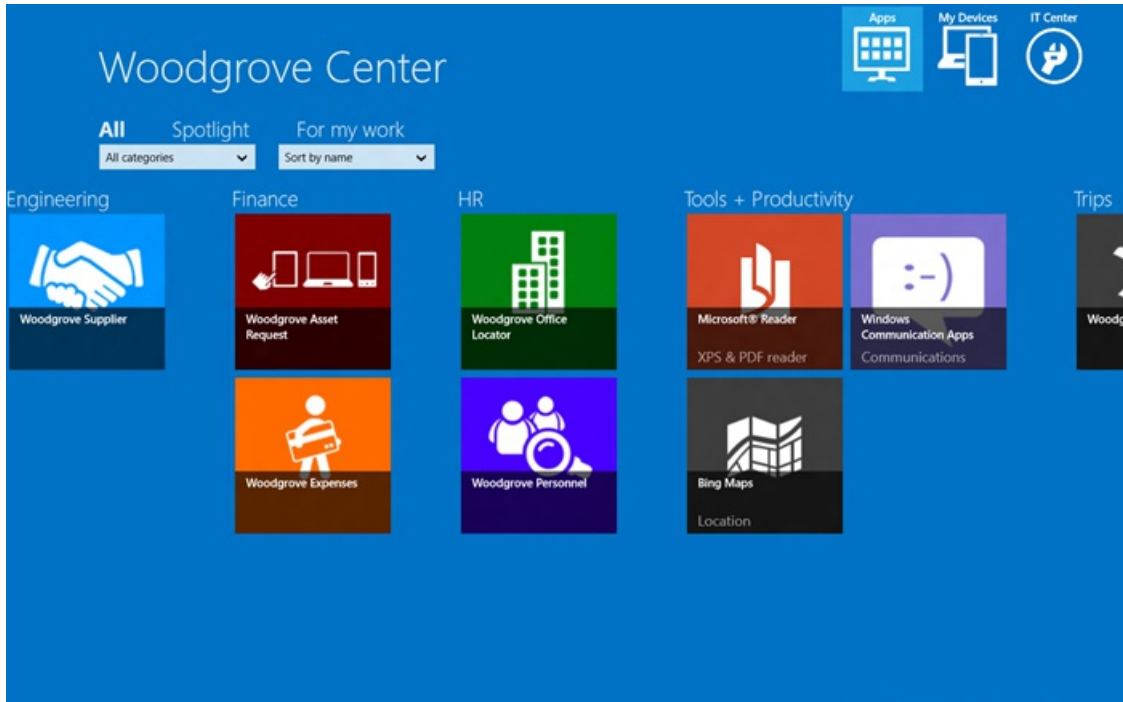
The features we've covered so far are obviously focused more on the mechanics of the management client and infrastructure along with the needs of the IT admin, but ultimately the entire solution exists to benefit the end user by enabling access to their LOB apps. Without such a benefit there's little reason a user would go through the trouble of using the enterprise management infrastructure. So let's dig deeper into LOB app delivery on the WOA platform.

In our previous blog post about WOA, we told you that "consumers obtain all software... through the Windows Store and Microsoft Update or Windows Update." Now, with the addition of the WOA management client, we're adding a fourth trusted source of software for the WOA platform. As mentioned, the Metro style self-service portal app, or SSP, is the day-to-day interface for the corporate user to access their

management infrastructure. Here they can browse to discover LOB apps that have been made available to them by the IT admin. There are actually four different types of apps that IT can publish for users in the SSP:

- Internally-developed Metro style apps that are not published in the Windows Store
- Apps produced by independent software vendors that are licensed to the organization for internal distribution
- Web links that launch websites and web-based apps directly in the browser
- Links to app listings in the Windows Store. This is a convenient way for IT to make users aware of useful business apps that are publicly available.

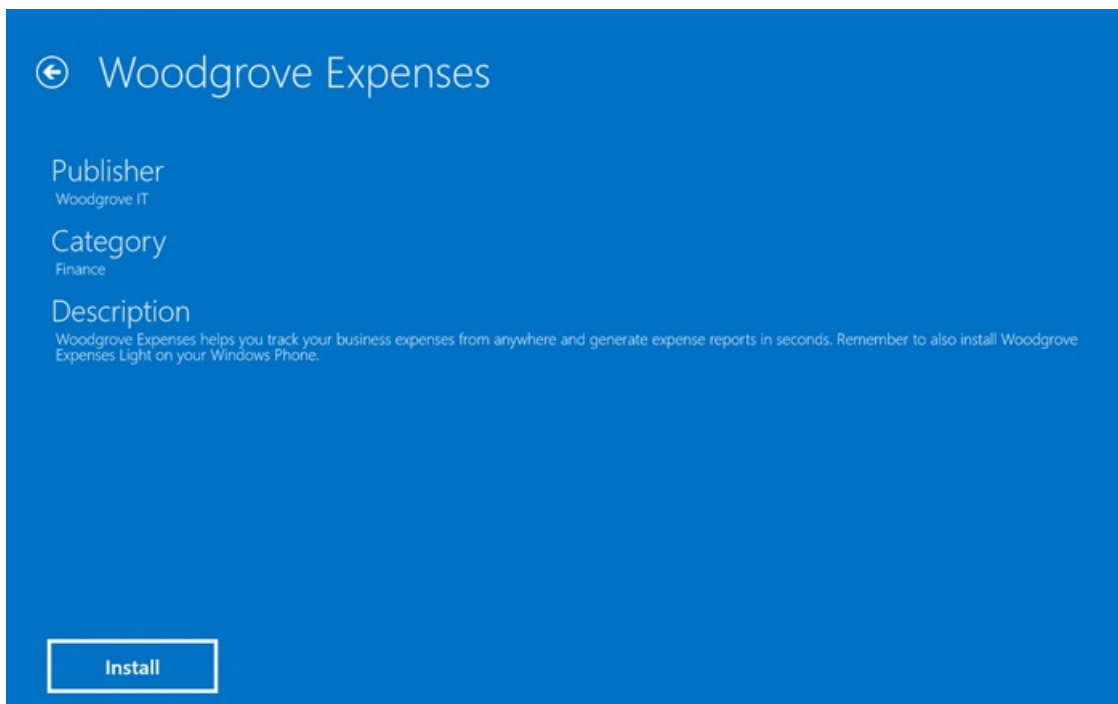
Since the user specified his or her corporate credentials as part of the initial connection with the management infrastructure, the IT admin can then specify which apps are published to each user individually, based on the user's AD domain user account, or as a member of AD user groups. As a result, the user only sees those apps that are applicable to them in the SSP.



*Browsing for LOB apps in the self-service portal (SSP) for a fictional company called Woodgrove
NOTE: This screenshot shows an early prototype of the SSP and may not reflect the final product.*

Before any LOB apps can be delivered through the management infrastructure, there are two things that happen on the client. First, an activation key is issued by the management infrastructure and applied to the WOA device to allow the agent to install apps. Second, any certificates used to sign the LOB apps must be added to the certificate store on the device. In most cases, both the activation key and the root certificates are automatically applied during the first session after establishing the connection with the management infrastructure. Otherwise, they are automatically deployed during a subsequent session after the IT admin has turned on the feature in the management infrastructure.

When the user chooses to install an app from the SSP, the request is sent to the management infrastructure and a download link is provided to the agent. The agent then downloads the app, verifies the validity of the content, checks the signature, and installs the app. All of this typically occurs within seconds and is generally invisible to the user. In the event that an error occurs during any part of this process (e.g. the location of the content is unavailable), the agent queues the app for a retry during its next regularly scheduled maintenance session. In either case, the agent reports the state of the installation back to the management infrastructure.



The details page of an app in the SSP, where the user can initiate installation

NOTE: This screenshot shows an early prototype of the SSP, and may not reflect the final product.

As part of its regular maintenance sessions, the agent will inventory which LOB apps are currently installed and report that information back to the management infrastructure so the IT admin can effectively manage their LOB apps. Only Metro-style apps that were installed via the SSP and the management client are included in this inventory from a WOA device. Apps installed from the Windows Store are never reported as part of the inventory.

Anytime the IT admin publishes an update for an app that has been installed on a WOA device, the agent will automatically download and install the update during its next regular maintenance session.

Disconnecting from the management infrastructure

Finally, let's look at how to disconnect a device from the management infrastructure. Disconnecting may be initiated either locally by the user or remotely by the IT admin. User-initiated disconnection is performed much like the initial connection, and is initiated from the same location in the Control Panel. Users may choose to disconnect for any number of reasons, including leaving the company or getting a new device and no longer needing access to their LOB apps on the old device. When an admin initiates a disconnection, the agent performs the disconnection during its next regular maintenance session. Admins may choose to disconnect a user's device after they've left the company or because the device is regularly failing to comply with the organization's security settings policy.

During disconnection, the agent does the following:

- Removes the activation key that allowed the agent to install LOB apps. Once removed, any Metro style apps that were installed via the SSP and management client are deactivated. Note, however, that the apps are not automatically removed from the device, but they can no longer be launched and the user is no longer able to install additional LOB apps.
- Removes any certificates that the agent has provisioned.
- Ceases enforcement of the settings policies that the management infrastructure has applied.
- Reports successful deactivation to the management infrastructure if the admin initiated the process.
- Removes the agent configuration, including the scheduled maintenance task. Once completed, the agent remains dormant unless the user reconnects it to the management infrastructure.

Summary

Given the trend towards "consumerization" of IT and our introduction of WOA with Windows 8, we wanted to rethink the way systems management is done. We worked to strike a balance between the sometimes competing needs of IT admins and the consumer who uses the device on a day-to-day basis. With the new WOA management client connecting to a management infrastructure in the cloud, we believe we've accomplished those goals, and we hope you'll agree when you see it all in action.

-- Jeffrey Sutherland

Making personal cloud storage for Windows available anywhere, with the new SkyDrive

Steven Sinofsky | [2012-04-23T08:57:00+00:00](#)

In February, we told you about our goals for [connecting your apps, files, PCs, and devices to the cloud with SkyDrive and Windows 8](#). Since then, we have provided the App Preview of a Windows 8 app to access SkyDrive, and we've updated the SkyDrive web experience. Today, we are delivering new capabilities for SkyDrive across the Windows platform.

Mike Torres, and Omar Shahine, group program managers for SkyDrive, co-wrote this post.

--Steven

Over the last year we've been hard at work building SkyDrive alongside Windows 8, setting out [a unique approach](#) to designing personal cloud storage for billions of people by bringing together the best aspects of file, app, and device clouds. Meanwhile, we've made our file cloud more accessible with [HTML5](#) and [mobile apps](#), improved integration with [Office](#) and [3rd party](#) apps, and built a device cloud for [Windows](#) and [Windows Phone](#).

Today, we're excited to take another big step towards our vision by making SkyDrive far more powerful. There are new storage options, apps that connect your devices to SkyDrive, and a more powerful device cloud that lets you "fetch" any file from a Windows PC. Taken together with access from popular mobile phones and a browser, you can now take your SkyDrive with you anywhere, connect it to any app that works with files and folders, and get all the storage you need—making SkyDrive the [most powerful](#) personal cloud storage service available.

Here's what's available for use, starting now:

- **SkyDrive for the Windows desktop (preview available now).** View and manage your personal SkyDrive directly from Windows Explorer on Windows 8, Windows 7, and Windows Vista with this new preview app available in 106 languages worldwide.
- **Fetching files through SkyDrive.com.** Easily access, browse, and stream files from a remote PC running the preview app to just about anywhere by simply fetching them via SkyDrive.com.
- **SkyDrive storage updates.** A new, more flexible approach to personal cloud storage that allows power users to get additional paid storage as their needs grow.
- **SkyDrive for other devices.** We've updated the SkyDrive apps on Windows Phone and iOS devices, bringing better management features and sharing options to those devices. We're also releasing a new preview client for Mac OS X Lion, letting you manage your SkyDrive right from the Finder.

You can [download the new SkyDrive apps](#) now, but you might want to take a look at this video first, which gives you a glimpse of all the things you can do with the new SkyDrive.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

SkyDrive for Windows

In February, we [announced a SkyDrive Metro style app](#) for Windows 8, SkyDrive for the Windows desktop, and a feature called "fetch" that allows you to remotely access files or stream videos from a connected PC. When you combine all of these features, you can seamlessly access any file on your Windows 8 PC from anywhere. The SkyDrive Metro style app was first made available with the [Windows 8 Consumer Preview](#), and today we're releasing a preview of SkyDrive for the Windows desktop including "fetch" support. But first, here's a little background.

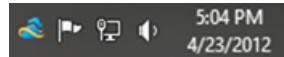
Over the years, we've consistently heard from our most loyal customers that having SkyDrive accessible from Windows Explorer is important, and we're happy to announce that, as of today, when you download the preview of SkyDrive for the Windows desktop, you'll be able to **access your SkyDrive from Windows Explorer on Windows 8, Windows 7, and Windows Vista**. The benefits of SkyDrive integration with Windows are clear: you can now drag-and-drop to and from SkyDrive with files up to 2GB, access all of your files offline, and have the full power of Windows Explorer available to manage your SkyDrive files and folders. Files stored in your SkyDrive are in a plain folder on your PC, which means any app that works with local folders and files can now work with SkyDrive.

As we set upon the path to bring SkyDrive closer to Windows, we had a few goals that drove our plan. First, we wanted you to be able to "**get up and running**" as quickly as possible, with very few steps. Secondly, we wanted to "**be quiet**" on the system and make sure that all processing was entirely in the background, with your needs and your apps as the first priority. And third, we really wanted it all to "**just work**" as you'd expect it to, staying up-to-date automatically, and humming along without confusing dialogs or pop-ups. Here's a bit more about where we're at for each of those.

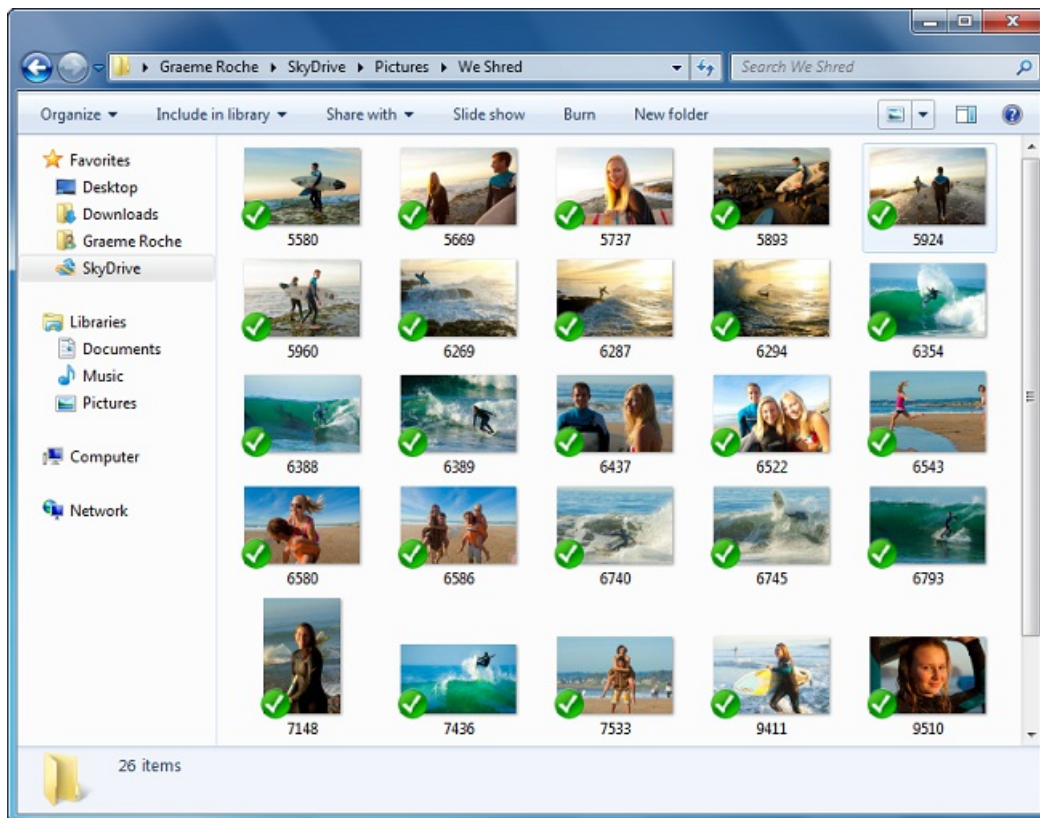
Downloading the preview of SkyDrive for Windows takes just **a few seconds** on most connections (the installer is under 5MB) and installs on most PCs in less than 10 seconds. There are just three simple setup screens and you're finished.



Once it's running, it's **out of the way** in the system tray. A folder is created automatically for you in a default location or one you choose during setup, and your SkyDrive files immediately start to appear.



Once your SkyDrive is available on your PC, this **special folder stays in sync** with your SkyDrive. If you rename a file on your phone, it appears immediately in this folder on your PC. If you delete a file from SkyDrive.com, it is deleted immediately here as well. Or if you create a folder and move files from another PC, Mac, or iPad, those changes immediately sync, too.



Power users can have fun with the SkyDrive folder too

In Windows Live Mesh, which some of you have come to rely on, we allowed arbitrary folders to be synchronized. Our experience has been that this introduced too many unresolvable complexities across different PCs, with the path on one PC synchronizing to entirely different paths on other PCs and the cloud. In order to maintain our goal of “it just works,” we designed SkyDrive to be the same everywhere, and to work well with libraries in Windows.

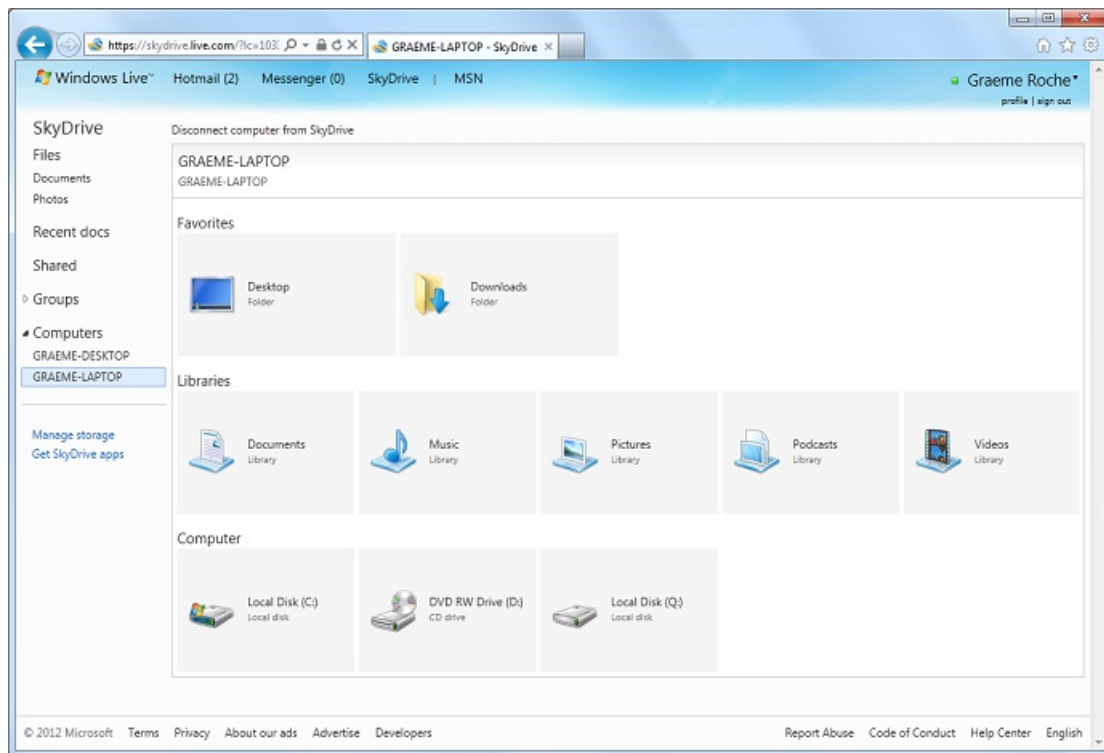
If you'd like your SkyDrive folders to feel less like separate folders, you can add your SkyDrive Documents and Pictures folders to your Documents and Pictures Libraries in Windows 8 and Windows 7.

Alternatively, you could change the target location for special folders like Documents or Pictures (or others) to folders in your SkyDrive, basically treating your SkyDrive as your primary drive (right-click the Documents folder, click Properties, and then Location). You can also customize the default root of the synchronized folder (to use a different drive, for example), and this option is available during setup of the SkyDrive app.

So, as you can see, the simple and straightforward model of having a single folder for your SkyDrive still leaves lots of creative options for personalization.

Fetching files through SkyDrive.com

As we [discussed and demonstrated](#) back in February, with SkyDrive running on a Windows PC, you can also turn that PC into your own private cloud to browse your files and stream videos from anywhere through SkyDrive.com. This feature is great if you forgot something on your home PC and need to fetch it or just copy it quickly to SkyDrive.



Note that, in order to access a remote PC you will have to provide a second factor of authentication beyond your account password. You'll need to enter a code that we send to your mobile phone or alternate email address even if you're already signed in to your SkyDrive account (if you're already on a [trusted PC](#), you won't have to do this every time, and it is easy to do this one-time setup). This means that anyone wanting access to your remote PC would have to have access not only to your account, but also to either an alternate email or your phone (which they would need to physically possess).

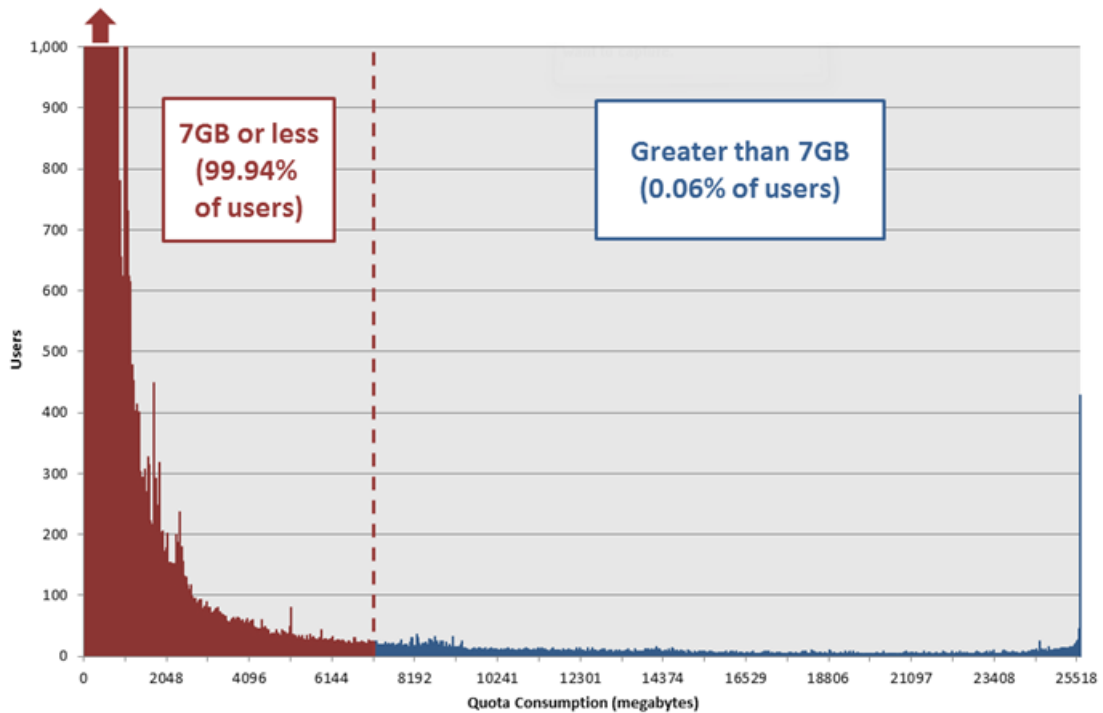
New, more flexible approach to storage

One of the challenges in building personal cloud storage for billions of people is scaling capacity and managing costs, while also meeting the needs of both enthusiasts and mainstream users. Different cloud providers take different approaches. Many promise unlimited storage or big referral incentives to attract enthusiasts – but then have lots of strings attached, which can make the service more confusing and less accessible to mainstream users. Do I really have to [read multiple pages](#) to understand my storage limits? Why do other people's files [count against my storage limit](#)? Why does my upload speed [slow down](#)? Why do I get gobs of free storage but have to [pay to sync](#) my desktop files?

Our model for SkyDrive is friendly and accessible to all, and just as importantly, provides a gimmick-free service that strikes the right balance of being free for the vast majority of customers, and low-priced for those who want more.

Starting today, we are now offering:

- **7GB free for all new SkyDrive users.** We chose 7GB as it provides enough space for over 99% of people to store their entire Office document library and share photos for several years, along with room for growth. To put things in perspective, 99.94% of SkyDrive customers today use 7GB or less – and 7GB is enough for over 20,000 Office documents or 7,000 photos. Since the current base of customers using SkyDrive tilts towards enthusiasts, we are confident that, as we expand the range of people using SkyDrive, this 7GB free limit will prove to be more than enough for even more people.



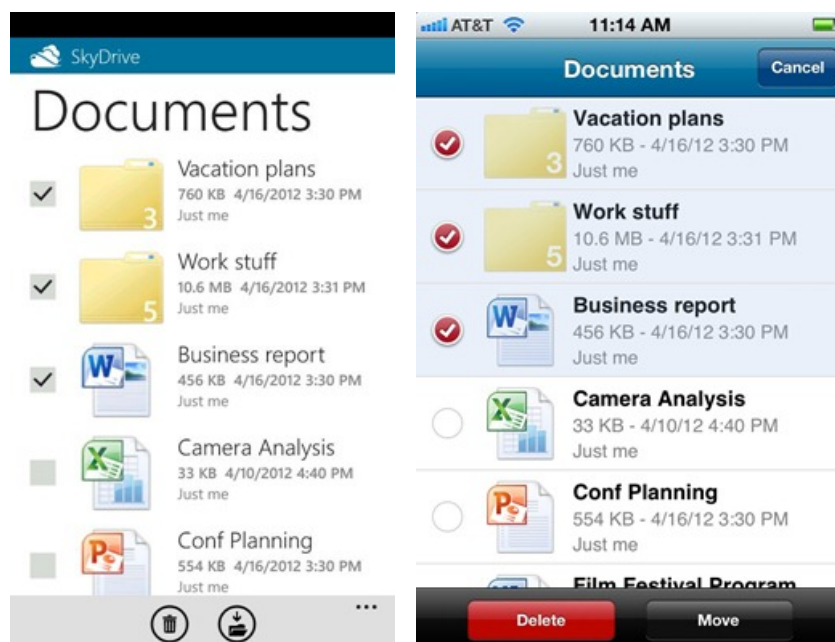
- **Ability to upload large files – up to 2GB – and folders** using SkyDrive for the Windows desktop or SkyDrive for OS X Lion.
- **Paid storage plans (+20GB, +50GB, +100GB)** so that power users who need more storage can easily add more at competitive prices (US\$10/year, US\$25/year, US\$50/year). Please note that paid-for storage requires the ability to pay by credit card (or via PayPal, in some markets) and a Windows Live ID that can be associated with that credit card or PayPal account.

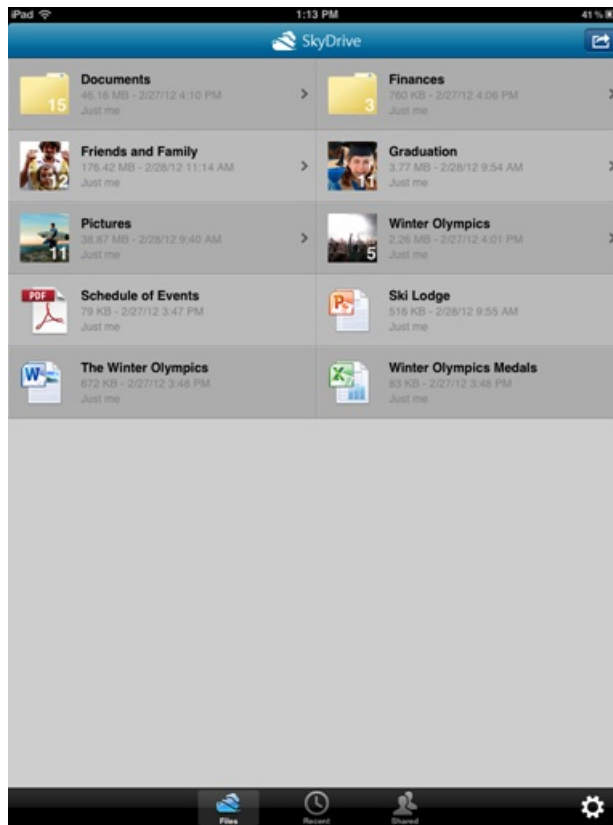
We know that many of you signed up for a service that offered 25GB, and some are already using more than 7GB of storage. So, starting today, for a limited time, **any registered SkyDrive user *who has uploaded files to SkyDrive* as of April 22nd can opt in to keep 25GB of free storage while still getting all of the benefits of the new service.** (For users who are already using more than 4GB as of April 1, we've automatically opted you in to 25GB of free storage to avoid any issues.) Just [sign in here](#) or [view our FAQ](#).

SkyDrive for Windows Phone and other devices

SkyDrive has been available since 2007 from anywhere in the world through SkyDrive.com, but it wasn't until the initial release of Windows Phone and our [dedicated Windows Phone and iPhone apps](#) in December 2011 that people had top-notch SkyDrive experiences from modern smartphones. These apps have been installed on over 2 million phones already by people taking SkyDrive with them wherever they go.

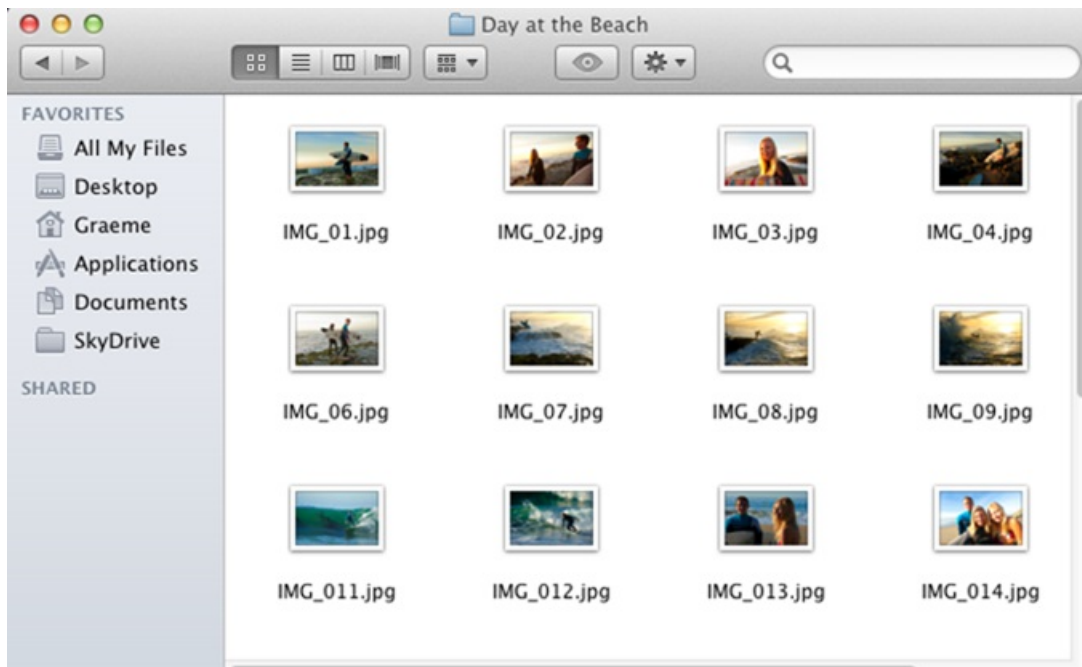
As a Windows Phone or iPhone user, with today's release, you can now delete, rename, and move files in your SkyDrive, and access a full set of sharing options for all files and folders. We're also bringing SkyDrive to the iPad, with all the same capabilities you now have available through the iPhone, plus support for the new iPad retina display.





All of these apps also have dozens of small improvements, including the ability to see your remaining storage space, landscape support, and various performance enhancements and bug fixes.

Almost 70% of Mac users also regularly use a Windows PC. Since we want every customer to be able to rely on SkyDrive to access files anywhere, it's important for SkyDrive be wherever they are. Office for Mac 2011 already [supports SkyDrive](#) files, but starting today, you'll also be able to manage your entire SkyDrive offline using Finder on the Mac. The integration with Finder means that any Mac app that opens from or saves to the file system will be able to take advantage of SkyDrive files as well.



Here's where you go to try SkyDrive today:

- [Get SkyDrive for Windows \(preview\)](#)
- [Get SkyDrive for Windows Phone](#)
- [Get SkyDrive for iPhone and iPad](#)
- [Get SkyDrive for OS X Lion \(preview\)](#)

If you currently use Mesh, we have a few tips for trying [SkyDrive for Windows or Mac \(preview\) side-by-side with Mesh](#). We think you'll find SkyDrive to be increasingly useful over time.

Thanks for supporting SkyDrive and we look forward to your feedback!

Mike (SkyDrive apps)

and

Omar (SkyDrive.com)

Note: Apps and the ability to purchase extra storage are rolling out now, and may take up to a few days to be available in all markets.

*Correction 4/23/12, 11:15 AM PST: Revised wording to clarify that loyalty offer is only for existing users who have uploaded files to SkyDrive before April 22, 2012.

Cloud services for Windows 8 and Windows Phone: Windows Live, reimagined

Steven Sinofsky | [2012-05-02T10:43:00+00:00](#)

We have talked quite a bit about SkyDrive and using your Microsoft account for the sign-in and roaming capabilities of Windows 8. These are just two aspects of a broad service infrastructure that you can tap into when using Windows 8 (and Windows Phone, Xbox LIVE, and a host of other services and apps). We want to talk more about the capabilities and features of cloud services for Windows 8 and Windows Phone. To get started, Chris Jones, the VP of our Windows Live group responsible for the development and operations of all of the services and apps, authored this post to introduce the reimagined role of cloud services in Windows 8.

--Steven

Windows Live was first announced on November 1st, 2005, and in our press release we described it as “a set of personal Internet services and software designed to bring together in one place all of the relationships, information and interests people care about most, with more safety and security features across their PC, devices and the Web.” Since that time, we’ve been hard at work building software and services that deliver that promise, a foundation that we could rely on as we designed new versions of Windows as well as other Microsoft products. We’ve received lots of feedback about features and ways we could improve the software and services. And we’ve also received some feedback about the naming and marketing we have done. Windows 8 is a chance for us to act on that feedback and reintroduce you to the broadest and most widely used collection of services on the Internet.

Today, Windows Live services are used by over 500 million people every month. There has been a lot of discussion recently on what constitutes an “active” user of a service; for the purposes of this post this term refers to people who use Hotmail, SkyDrive, or Messenger at least once a month, meaning they send email, use instant messaging, or upload files to SkyDrive.

These services run at massive scale – Hotmail is the world’s leading web email service, with 350 million active users and 105 petabytes of storage; Messenger is the world’s leading instant messaging service, with 300 million active users, and SkyDrive has over 130 million users with 17 million of these uploading files every month. Windows Live Essentials applications are among the most popular applications in their categories on Windows – including Windows Live Photo Gallery and Windows Live Movie Maker, leading in photo management and video editing, and Windows Live Mail, second only to Microsoft Outlook in mail apps.

While these results are certainly noteworthy, they still did not meet our expectations of a truly connected experience. Windows Live services and apps were built on versions of Windows that were simply not designed to be connected to a cloud service for anything other than updates, and as a result, they felt “bolted on” to the experience. This created some amount of customer confusion, which is noted in several reviews and editorials. The names we used to describe our products added to that complexity: we used “Windows Live” to refer to software for your PC (Windows Live Essentials), a suite of web-based services (Hotmail, SkyDrive, and Messenger), your account relationship with Microsoft (Windows Live ID), and a host of other offers.

Windows 8 provides us with an opportunity to reimagine our approach to services and software and to design them to be a seamless part of the Windows experience, accessible in Windows desktop apps, Windows Metro style apps, standard web browsers, and on mobile devices. Today the expectation is that a modern device comes with services as well as apps for communication and sharing. There is no “separate brand” to think about or a separate service to install – it is all included when you turn on your PC for the first time.

We also believe that you should have a choice and control over what services you use, what information you share (with others and Microsoft), and how you access your services. That’s why using any of these services is optional, and you’re welcome to mix and match them with the software and services you choose.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

Microsoft account is our identity service for individuals who use Microsoft products and services. You can use your Microsoft account to [sign in to your Windows 8 PC](#), and then use the same account to check your billing for services like Xbox LIVE, Zune, and the Windows 8 app store. And your Microsoft account is connected to your Xbox gamer tag so you can track high scores and games. You can sign up for a Microsoft account with any email address, and provide additional verification information including your mobile phone number and a list of your trusted devices. We’ll be rolling out the change in nomenclature from Windows Live ID to Microsoft account over the next several months across our product line. There are still some areas we continue to work on such as migrating your account (credit cards and purchase history) from one market (currency) to another if you’ve connected your account to services such as Xbox LIVE.

When you connect a device or service to your Microsoft account, you’re automatically provisioned with a set of **cloud services**, including a contact list, calendar, inbox, instant messaging, and cloud storage. Of course these services connect to your PC and your Windows Phone, they’re accessible from any web browser, and they’re accessible to different apps if the developer of the app implements our API. Because these

services are a part of your Microsoft account, they are shared across all Microsoft products and services. For example your contact list is shared across Windows Phone, Windows 8, Hotmail, Messenger, and SkyDrive, so when you add a contact in one place, it shows up in the cloud and on all of your other devices and services.

Windows 8 also uses cloud services to roam settings across your PCs so you can log in to a new PC and pick up right where you left off. Along with a Microsoft account, everyone gets a **SkyDrive**, which is cloud storage for documents, photos, your phone's camera roll, and settings from your PC. SkyDrive powers the Windows Phone camera roll, so every picture you take is automatically copied to your cloud photo album. SkyDrive makes it easy for you to share and collaborate on Office documents, either using Office Web Apps or Office client applications. And developers can use the [SkyDrive APIs](#) to provide an even deeper level of roaming and support in their apps if they choose. Through the innovative features of [Contracts](#) and [File Pickers](#) in Windows 8, you can access your SkyDrive data from within any Windows 8 Metro style app.

We recognize that customers will have services from many different companies, particularly for social networking and communications. So we let you **connect your Microsoft account** to other services. Just like our contact list, your connected services are stored in the cloud and roam across your devices. This means that if you connect your Microsoft account to LinkedIn, Facebook, or Twitter, your contacts from these networks show up in your contact list, so you can send them email from your PC or call them from your phone. We also support 3rd party developers through the [Live SDK](#), allowing developers to cloud-power their Metro style apps, or apps and services for other platforms. Our APIs use standard and familiar protocols including OAuth 2.0, JSON, REST, Exchange ActiveSync, and XMPP.

Windows 8 is designed to be cloud-powered, so it comes with **Metro style apps** for communication, sharing, scheduling, photos, and videos. Preview versions of these apps come installed with the Windows 8 Consumer Preview and include **Mail, Calendar, People, Photos, Messaging, and SkyDrive**. They're all powered by cloud services, so when you sign in with your Microsoft account, your email, calendar, contacts, messages, and shared photo albums show up right in your apps. For customers who have shared family PCs, **family safety** is now a feature of Windows accounts and no longer requires a separate download. As we've discussed before, Windows Phone comes with the same set of apps, powered by cloud services, and connected to your Microsoft account. For customers who use Windows 7, we have a set of **Windows desktop apps**, including Photo Gallery, Movie Maker, Mail, Messenger, Family Safety, and our recently released [SkyDrive for the Windows desktop](#).

The chart below breaks down our software and services in the new world of Windows 8:

Service	Windows 8	Windows Phone	Web/HTML 5 (live.com)	API (dev. live.com)	Earlier Versions
Account	Microsoft account	Microsoft account	Account.live.com	OAUTH	Windows Live ID, Passport
Storage/ Docs	SkyDrive app, SkyDrive Desktop	SkyDrive app, Office app	SkyDrive.com	REST, JSON	FolderShare, Live Mesh, Windows Live Mesh
Email	Mail app	Mail app	Hotmail.com	EAS	Windows Live Mail, Outlook Express
Calendar	Calendar app	Calendar app	Calendar.live.com	EAS, REST	Windows Live Mail, Windows Calendar
Contacts	People app	People app	People.live.com	EAS, REST	Windows Contacts
Messaging	Messaging app	Messaging app	Integrated in Hotmail and SkyDrive	XMPP	MSN Messenger
Photos/ Videos	Photos app, Photo Gallery, Movie Maker	Photos app, Camera Roll	Photos.live.com	REST, JSON (via SkyDrive)	Windows Live Photo Gallery, Windows Live Movie Maker

In the coming weeks we will share more information about our software and services for individuals spanning the PC, phone, and web, and go into detail on Microsoft account, cloud services, SkyDrive, Hotmail, Messenger, as well as our upcoming work with Skype. For now you can start

using our apps and services on Windows 8 Consumer Preview, your Windows Phone, on the web with [Hotmail](#) or [SkyDrive](#), or on your other devices.

--Chris Jones

Making Windows Media Center available in Windows 8

Steven Sinofsky | [2012-05-03T15:22:00+00:00](#)

*In this post we wanted to update you on Media Center and Windows 8, specifically how we will make sure Windows 8 fully supports the capabilities of Media Center as it is in Windows 7. We took the feedback about maintaining the functionality very seriously, and we clearly understood what we've heard many of you saying around the value of Media Center for movies, Internet TV, broadcast TV, optical media, music, photos, and all the other scenarios it covers today. Many said in comments and email to us, that so long as the feature is available somehow it is fine. This post is how we will deliver on that and continue to support Media Center for another product lifecycle. **This post was authored by Bernardo Caldas in the Windows Business Group, with help from Linda Averett who leads program management for the Developer Experience team.***

--Steven

If you saw our recent post on the [Windows 8 editions](#), then you know already that Windows Media Center will be available in Windows 8. You might also have noticed Windows Media Center is included in Windows 8 Consumer Preview. Media Center has always been subject of a lot of discussion and feedback in these forums as well as email. Today we would like to share more details about our plan and the motivations behind it.

First let's step back and talk about media experiences in general. Windows 8 will deliver a world-class video and audio entertainment experience. Our focus is on providing a comprehensive video and audio platform for developers to build engaging and differentiated apps. The Windows 8 developer platform will contain a wide variety of industry-standard media formats, along with Internet Explorer 10, which supports the standard HTML5 web platform. It also includes the set of decoders (shown in the table below) and new developer functionality to deliver these modern media experiences.

Metro style apps can use any of the decoders included in Windows. These decoders are optimized for system reliability, battery life, and performance, and cover all key playback scenarios for mainstream content such as YouTube video, Netflix video, Amazon audio/video, H.264 web browsing/streaming, Hulu video, MP4 video, AVCHD video from camcorders, Ultraviolet video, and the HTML5 video tag. Metro style apps can also include additional decoders (such as FLAC, MKV, OGG, etc.) in their apps package for use within the apps.

	Video	Audio
Decoders	H.264	DD+ (non-disk)
	VC-1/WMV	AAC
	MP4 Pt 2	WMA
		MP3
		PCM
Format container	AVI	M4A
	MPEG-2 TS	ASF
	MP4	MP3
	ASF	WAV

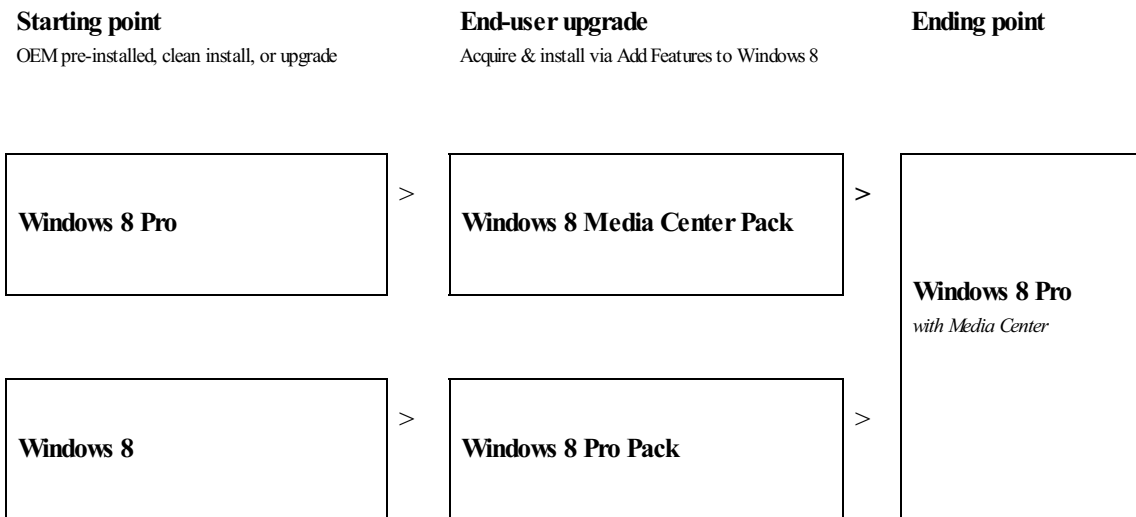
In the process of building a robust platform, we've also evaluated which in-box media playback experiences we want to provide. The media landscape has changed quite significantly since the release of Windows 7. Our telemetry data and user research shows us that the vast majority of video consumption on the PC and other mobile devices is coming from online sources such as YouTube, Hulu, Netflix, or any of the other myriad of online and downloadable video services available. In fact, consumption of movies online in the United States will surpass physical video in 2012, according to this recent [IHS Screen Digest research](#).

On the PC, these online sources are growing much faster than DVD & broadcast TV consumption, which are in sharp decline (no matter how you measure—unique users, minutes, percentage of sources, etc.). Globally, DVD sales have declined significantly year over year and Blu-ray on PCs is losing momentum as well. Watching broadcast TV on PCs, while incredibly important for some of you, has also declined steadily. These traditional media playback scenarios, optical media and broadcast TV, require a specialized set of decoders (and hardware) that cost a significant amount in royalties. With these decoders built into most Windows 7 editions, the industry has faced those costs broadly, regardless of whether or not a given device includes an optical drive or TV tuner.

Our partners have shared clear concerns over the costs associated with codec licensing for traditional media playback, especially as Windows 8 enables an unprecedented variety of form factors. Windows has addressed these concerns in the past by limiting availability of these experiences to specialized “media” or “premium” editions. At the same time, we also heard clear feedback from customers and partners that led to our much simplified [Windows 8 editions lineup](#).

Given the changing landscape, the cost of decoder licensing, and the importance of a straight forward edition plan, we’ve decided to make **Windows Media Center** available to Windows 8 customers via the **Add Features to Windows 8** control panel (formerly known as Windows Anytime Upgrade). This ensures that customers who are interested in Media Center have a convenient way to get it. **Windows Media Player** will continue to be available in all editions, but without DVD playback support. For optical discs playback on new Windows 8 devices, we are going to rely on the many quality solutions on the market, which provide great experiences for both DVD and Blu-ray.

We will offer two ways to acquire Windows Media Center:



Windows 8 Pro is designed to help tech enthusiasts obtain a broader set of Windows 8 technologies. Acquiring either the **Windows 8 Media Center Pack** or the **Windows 8 Pro Pack** gives you Media Center, including DVD playback (in Media Center, not in Media Player), broadcast TV recording and playback (DBV-T/S, ISDB-S/T, DMBH, and ATSC), and VOB file playback. Pricing for these Packs, as well as retail versions of Windows 8, will be announced closer to the release date. To give you some indication of Media Center Pack pricing, it will be in line with marginal costs.

We are incredibly excited about the future of entertainment in Windows. We hope you have had a chance to try some of the new Windows 8 Metro style media applications such as the Video and the Music apps. These apps embody the characteristics that make Windows 8 great for both end users and developers, and are included with the Consumer Preview install, ensuring a great local media playback experience on Windows 8. There is much more to come, as developers embrace the power of the Windows 8 platform to delight media enthusiasts around the world!

--Bernardo and Linda

FAQ – DVD playback and Windows Media Center in Windows 8

Steven Sinofsky | [2012-05-04T21:16:00+00:00](#)

We thought we would follow up the previous post with an FAQ which is based on the comments and discussions, so Bernardo put this together so things are in one place. Some of these might be introductory for some but since the comments covered a lot of topics, it seemed reasonable to start at the beginning. --Steven

What are the codecs needed to play DVD?

A codec is software that is used to compress or decompress a digital media file, such as a song or video. MPEG-2 is widely used as the format of digital television signals that are broadcast by terrestrial (over-the-air), cable, and direct broadcast satellite TV systems, and DVD Video. Dolby Digital is the widely used audio standard for terrestrial (ATSC, over-the-air), cable, direct broadcast satellite TV systems, and DVD Video. Dolby audio is also a mandatory format in Blu-ray.

How has Windows handled DVD related decoder licensing prior to Windows 8?

The issue surrounding the incremental costs of codecs to play DVDs isn't new to Windows. In Windows XP and Windows Vista we addressed it by offering specialized editions, such as Windows Media Center Edition, or codec add-ons to Windows Media Player. DVD playback was not included in Windows Vista Starter, Home Basic, Business, and Windows Vista Enterprise editions. OEMs (PC manufacturers) had the option to license Windows Vista Starter, Home Basic, and Business "with DVD" where we offered a version that includes the Dolby Digital codec to enable the OS to support DVD playback for a nominal price increase. In Windows 7, we decided to make these codecs available broadly in most editions, except Windows 7 Home Basic (available in some emerging markets) and Windows 7 Starter editions (available for netbooks and some emerging markets). That means royalties related to DVD playback in Windows 7 have been paid broadly, regardless of whether or not the PC has an optical drive. Based on sales and usage, we supplied codecs to a very large number of PCs that were not capable of playing DVDs or simply did not ever play DVDs.

Who pays decoder royalties associated with DVD playback on PCs?

Typically, media codecs are based on intellectual property (IP), often patents, held by consumer electronics consortiums or companies. The result is that entities who wish to sell products that include these codecs must pay royalties to the IP owners; sometimes to a single entity (e.g. Dolby Laboratories), and often through a license agency (e.g. MPEG-LA) who administers licensing for a number of IP holders under specific terms. The rules surrounding who pays these royalties vary by licensing program. According to the MPEG-LA program, the company that ships the end product is responsible for paying. In the case of new PCs with Windows pre-installed, that would be the PC OEMs. The Dolby program for Windows 7 was defined based on an agreement between Dolby and Microsoft where Microsoft has paid Dolby directly for the rights to Dolby Technologies built in Windows 7. Royalties are also paid by ISVs that include those technologies in their applications, even if those applications are bundled on new systems. This means that in many cases the same royalties can be paid multiple times over for a single PC (Microsoft pays some, OEM pays some, ISV pays some). In Windows 8, we will continue to include some technologies licensed by MPEG-LA and Dolby that will be paid by OEMs, but only those that relate to online media consumption (e.g. MPEG-2 container for H.264, Dolby Digital Plus audio) and not those related optical media. The costs associated with those codecs are lower, but significant, compared to optical media playback. Also, Windows 8 apps will be able to use these technologies as part of the Windows 8 Media Foundation APIs at no additional cost, as long as they are not providing optical media and broadcast related functionality.

How much does it cost the PC ecosystem to play DVDs?

Playing DVDs generally require MPEG-2 video compression and Dolby Digital (AC-3) audio. Even though it is possible to use other formats, the majority of commercial DVDs are encoded using these formats. In order to decode these formats, the playback device needs to be licensed to use these decoders. MPEG-2 decoder costs \$2.00 per unit under current MPEG-LA terms. Dolby license is an additional cost that varies by the technology licensed, the type of device, and unit volume. While not related to Windows, Blu Ray would be an additional cost on top of these. So when you add all this up and apply to all Windows PCs, it is an ongoing cost of hundreds of millions of dollars per year to the PC ecosystem, well over a billion dollars over the lifecycle of the operating system and yet by most predictions the majority of PCs will not even be capable of playing DVDs.

Why can't I just pay for DVD when I need it?

When we have DVD playback capabilities in software broadly like in Windows 7, there is no way to distinguish whether the PC will ever play a DVD disc but still this cost is carried on every PC. While we might think that the best solution is some sort of "just in time" charge back to Microsoft based on telemetry or an "anytime upgrade" this is not how the third-party licensing programs work as described above. So there isn't an approach where you buy Windows or a PC and only "pay as you go" if DVD playback is provided "in the box". Once it is distributed as a player, a license is required.

Will devices with Windows 8 pre-installed be able to play DVDs out of the box?

This is ultimately an OEM choice for what peripherals and software to include in a given system. If a new device has an optical drive, it will most likely include necessary software and licenses making it a seamless experience to the vast majority of customers. Similarly, an add-on optical drive

(internal or external) will almost certainly come with DVD playback software unless you intentionally purchase a white label drive (which might be a perfectly reasonable choice if the drive is simply for loading software). In all cases, there are numerous complete third-party applications that provide a broad range of support that is properly licensed. On the other hand, the ecosystem won't have to pay for that software and related royalties on devices such as tablets, small form factor desktops, and laptops that are sold without optical drives.

What if I upgrade to Windows 8 on my current Windows 7 PC with a DVD drive?

If there is existing third-party playback software the Windows Upgrade Assistant will help determine if this software is compatible with Windows 8 and you will have the option to keep it during the upgrade to Windows 8. Otherwise, you will need to acquire third-party playback software after the upgrade to play DVDs. Alternatively, you can acquire the Windows 8 Media Center Pack or the Windows 8 Pro Pack post upgrade. Both Packs include Windows Media Center, including the ability to play DVDs.

Why can't I buy a Windows 8 device that includes Windows Media Center pre-installed?

With the evolution of device form factors (tablets, thin and light, etc., none of which have optical drives) and change in media consumption patterns from optical disks and broadcast TV to online (Netflix, Youtube, Hulu, etc.), we concluded that we would no longer make DVD and broadcast TV capabilities available in all Windows editions, simply because the feature applies to a decreasing number of PCs sold. Instead, those capabilities will be available only to customers that want it via Add Windows Feature (aka Windows Anytime Upgrade). This ensures that the costs associated with playing DVDs and watching broadcast TV on PCs only apply to devices that have those capabilities and customers that want it.

Are you adding another Windows 8 edition called "Windows 8 Pro with Media Center"?

The Windows 8 Pro edition that includes Media Center will be named and branded Windows 8 Pro. The only difference is that it will include Media Center and you will also find a different string in the system properties where it will say "Windows 8 Pro with Media Center". This is not a new edition of Windows 8.

Why do I have to upgrade to Windows 8 Pro to get Media Center?

Trends in Media Center usage show a decline in the number of customers that use it on a regular basis, starting from a relatively small base as we previously blogged about. When we look at actual usage, most customers using Media Center and playing DVDs used Windows Ultimate and XP Pro/Media Center. We believe those customers will also be interested in the additional features provided in the Windows 8 Pro edition, such as Boot from VHD, Client Hyper-V, etc., especially if they are using Media Center on a PC used for general tasks. Considering the audience and current usage, we conclude the vast majority of Media Center customers upgrading to Windows 8 will be to the Windows 8 Pro edition. In our efforts to keep the Windows 8 editions plan as simple as possible, Windows Media Center is only available on Windows 8 Pro. If you already have Windows 8 Pro and want to add Media Center, you just need to acquire the additional Media Center Pack as an in-place upgrade available via Add Windows Features (formerly Windows Anytime Upgrade).

What is the Windows 8 Pro Pack and why does it include Media Center?

Windows 8 Pro Pack is an upgrade from Windows 8 to Windows 8 Pro. Like we described above, Media Center is only available on Windows 8 Pro. When you acquire the Pro Pack, we make it a single step that takes you to Windows 8 Pro with Media Center. The cost of the Media Center Pack is essentially built into Pro Pack. Again, this is an attempt to add simplicity to the process of acquiring Media Center.

What version of Windows Media Center will be included in Windows 8?

The version of Media Center included in Windows 8 is what we shipped in the Windows 8 Consumer Preview. It is much consistent with what shipped in Windows 7.

Will CableCard and other devices continue to work with Media Center in Windows 8?

Yes, there is no change in hardware supported between Windows 7 and Windows 8.

Why doesn't Windows Media Player support DVD playback even after installing Media Center?

Based on the above discussion, it should be clear that we cannot enable DVD playback all the time in Windows Media Player. Given the ongoing feedback to avoid feature overlap and to avoid the complexity of behavior changing for a previously installed component, we only enable DVD playback in Media Center once it is installed.

If I upgrade in place, can't I just use the codecs that were already purchased with Windows 7?

The usage rights to these codecs needed to play DVD do not carry forward to a new version of Windows after you upgrade. These terms are defined by the licensors of these technologies, not Microsoft or OEMs.

Update, this FAQ was added to the original post.

Redesigning chkdsk and the new NTFS health model

Steven Sinofsky | [2012-05-09T15:00:00+00:00](#)

*We've written about tons of improvements in the OS kernel, networking, and file system. While for most client PCs, the tried and true chkdsk utility is one we rarely use anymore except in very rare circumstances, we are using Windows 8 as an opportunity to improve this utility. We wanted to focus on rethinking how the utility works to increase availability and reduce downtime due to chkdsk operations. In looking at the real world usage of chkdsk, we note that corruptions are exceedingly rare though running chkdsk is not. While we've worked hard to reduce the manual invocation of disk tools (like defrag) we know many prefer to run them manually "just in case" and so we worked to improve the overall throughput of chkdsk, since running it reduces availability of the machine. With disk capacities becoming extremely large and multi-disk systems more common, we wanted to improve the utility. **Kiran Bangalore, a program manager on our core system team, authored this post.***

--Steven

In this blog post, I'll talk about the new NTFS health model for Windows 8 and our redesigned tool for disk corruption detection and fixing, the chkdsk utility.

We've all experienced the frustration that can be caused by an unexpected chkdsk that pops up while restarting a computer at home or a server at the office. Beyond the surprise, there's the interruption while waiting for the process to complete and Windows to be available. With Windows 8, we provide quick resolution to these problems when they arise, putting the user in control and making systems more available and more scalable.

One of our key design goals for Windows 8 was to increase availability and reduce the overall down-time of systems; this feature, along with other storage features such as [Storage Spaces](#) and the new [ReFS](#) file system, helps reduce the complexity of fixing corruptions and increases the overall availability of the entire system.

The previous chkdsk and NTFS health model

While exceedingly rare, there are a variety of unique causes for disk corruption today. Whether they are caused by media errors from the hard disk or transient memory errors, corruptions can happen in file system metadata (the information used to map physical blocks to that vacation photo you took last year). To maintain access to your data, Windows must isolate and correct these errors, and the way to do this is by running the chkdsk utility.

In past versions, NTFS implemented a simpler health model, where the file system volume was either healthy or not. In that model, the volume was taken *offline* for as long as necessary to fix the file system corruptions and bring the volume back to a healthy state. Downtime was directly proportional to the number of files in the volume.

Reliable telemetry data from systems all over the world have shown us that, although corruptions are quite rare, when chkdsk is needed, it can take between a few seconds to a few hours to run, depending on the number of files in the drive—and even longer for larger storage servers.

In Windows Vista and Windows 7, we made significant optimizations to the speed of chkdsk but, as hard disk capacities have continued to double every 18 months and the number of files per volume is increasing at an equal rate, chkdsk has taken longer and longer to complete (even with speed improvements).

So in Windows 8, we've changed the way we approach the health model of NTFS and changed the way we fix corruptions so as to minimize the downtime due to chkdsk. We've also introduced a new file system for the future, [ReFS](#), which does not require an offline chkdsk to repair corruptions.

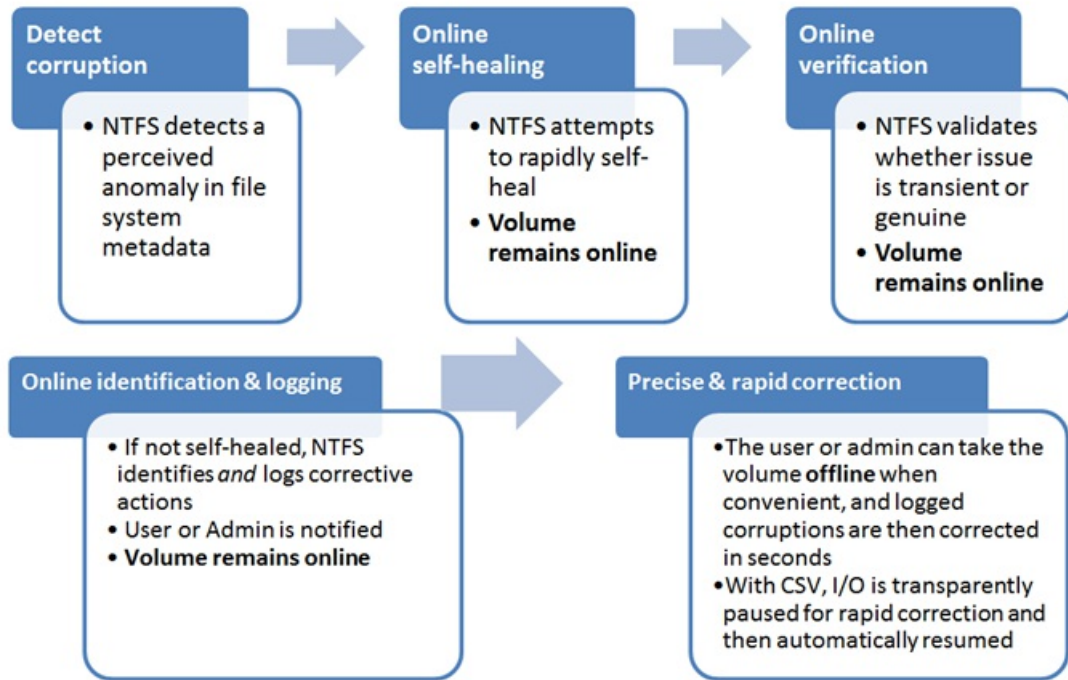
File system health redone

The incredible growth in storage capacity and user data files has necessitated the redesign of the NTFS health model and chkdsk.

There were three important requirements for file system health that our customers made clear:

1. Downtime caused by file system corruptions must be zero in continuously available configurations and nearly zero in all other configurations.
2. A User or Administrator must be made aware of the file system health at all times.
3. A User or Administrator should be able to easily fix their file system when a corruption occurs in a scheduled manner.

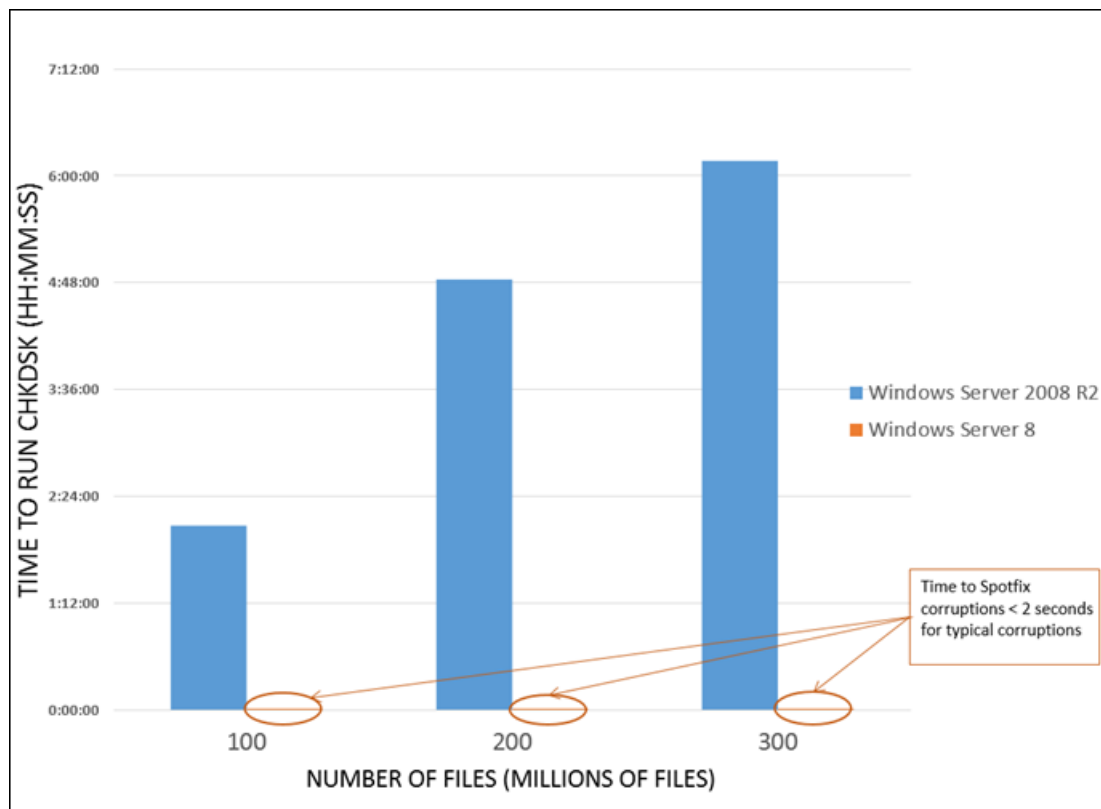
Our design included changes both in the file system and the chkdsk utility to ensure the best availability. The new design splits the process into the following phases to ensure a coordinated, rapid, and transparent resolution to the corruption.



We developed a new method of communication that describes types of corruptions as “verbs” that act upon the key components and points of the design – the file system driver (NTFS), the self-healing module, the spot-verification service, and the chkdsk utility. All file system corruptions are classified as needing one of 18 different “verbs” that we’ve defined in Windows 8. We have also left room for possible new verb definitions that can help us diagnose issues even better in the future.

Key design changes to help improve availability:

1. **Online self-healing:** The NTFS self-healing feature was introduced in Windows Vista (and in Windows Server 2008) to reduce the need to run chkdsk. Self-healing is a feature built into NTFS that fixes certain classes of corruptions encountered during normal operation, and can make these fixes while still online. If all issues that are detected are self-healed online, there is no need for an *offline* repair. In Windows 8 we increased the number of issues that can be handled online and hence reduced any further need for chkdsk.
2. **Online verification:** Some corruptions are intermittent due to memory issues and may not be a result of an actual corruption on the disk; so we added a new service to Windows 8, called the spot verification service. It is triggered by the file system driver and it verifies that there is actual corruption on the disk before moving the file system along in the health model. This new service runs in the background and does not affect the normal functioning of the system; it does nothing unless the file system driver triggers it to verify a corruption.
3. **Online identification and logging:** When an issue is verified, this triggers an online scan of the file system, which runs as a maintenance task in the file system. In Windows 8, scheduled tasks that are for the maintenance of the computer run only when appropriate (during idle time, etc.). This scan can run as a background task while other programs continue to run in the foreground. As the file system is scanned, all issues that are found are logged for later correction.
4. **Precise and rapid correction** – At the user or administrator’s convenience, the volume can be taken offline, and the corruptions logged in the previous step can be fixed. The downtime from this operation, called “Spotfix,” takes only seconds, and on Windows Server 8 systems with cluster shared volumes, we’ve eliminated this downtime completely. With this new model, **chkdsk offline run time is now directly proportional to the number of corruptions**, rather than being proportional to the number of files as in the old model.



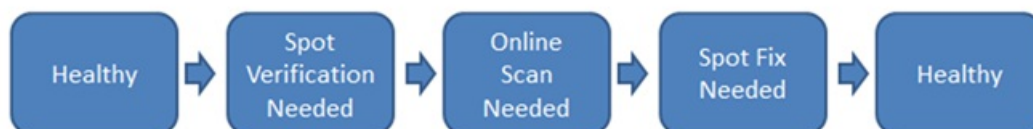
Comparison of Windows Server: *chkdsk /f* vs *chkdsk /spotfix*

- Better manageability** – To enable better transparency into the new health model, Windows now exposes the state of the file system via the following interfaces:
 - Action Center** – The health of the drive is most visible in the Action Center as the “Drive Status” (see figure below), which tells you when you need to take an action to bring the volume to a healthy state.
 - Explorer**: The health state is also exposed in Explorer, under Drive properties.
 - PowerShell**: You can also invoke the *chkdsk* functionality using a new cmdlet in PowerShell, *REPAIR-VOLUME*, which can be helpful for remote management of file system health.
 - Server Manager**: In Windows Server, you can also manage the volume health states directly from the server manager utility.

The new file system health model

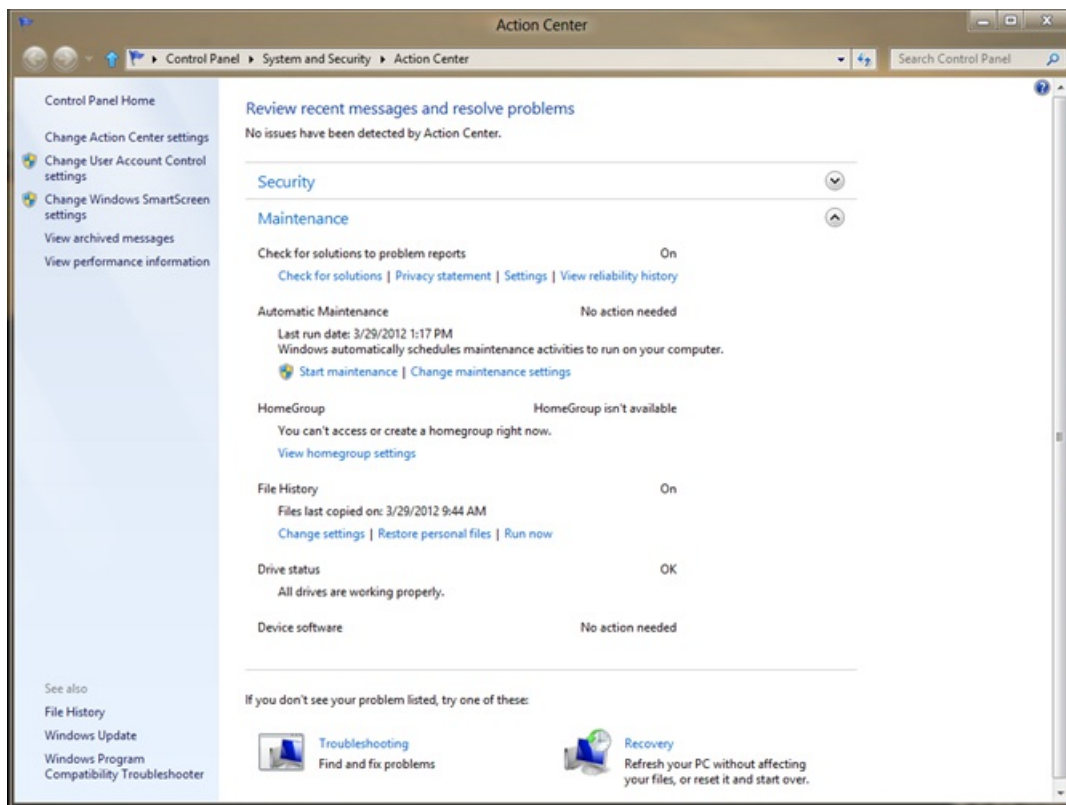
In the new health model, the file system health status transitions through four states – some that are simply informational, and others that require you to act. The health states are:

1. Online and healthy
2. Online spot verification needed
3. Online scan needed
4. Spot fix needed

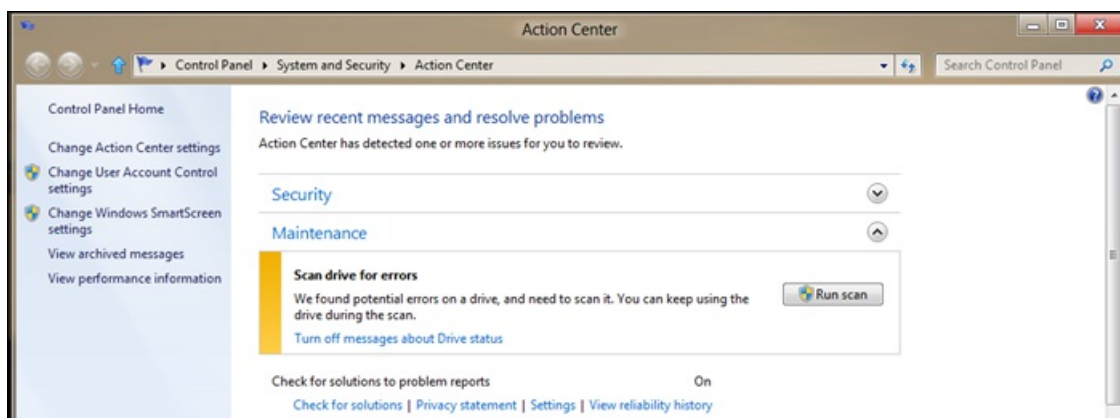
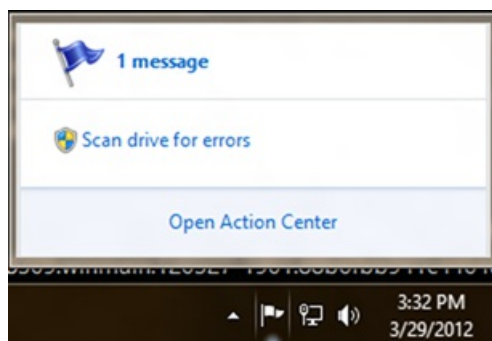


Windows 8 file system health states

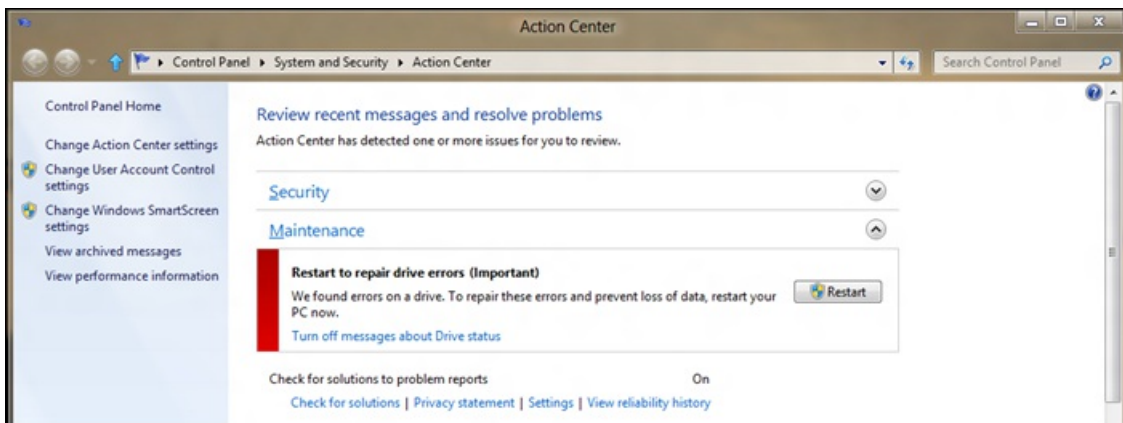
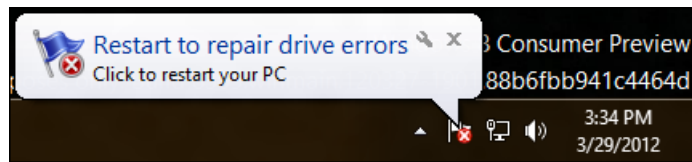
- Online and healthy** – In this state there are no detected file system corruptions and there is no action required of you. The file system remains in this state most of the time.



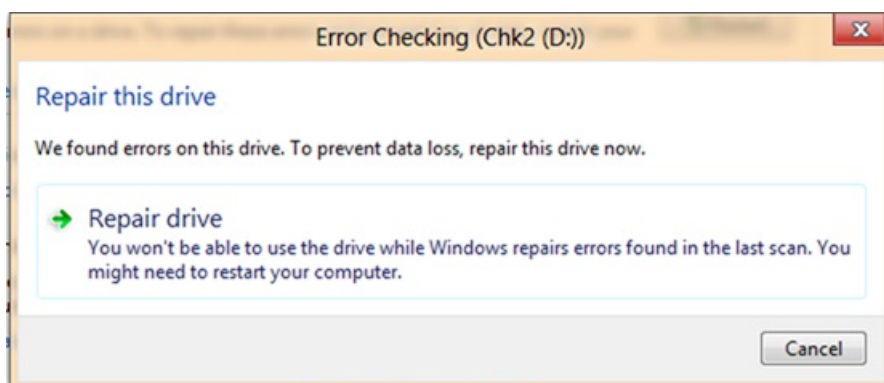
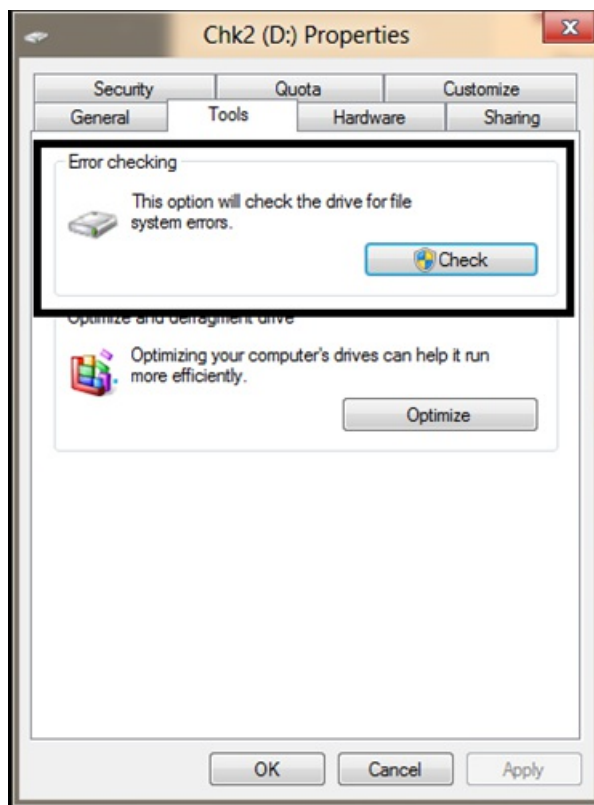
1. **Online spot verification needed** – The file system stays in this transient state only for a brief instant after the file system finds a corruption that it cannot self-heal; it puts the volume in this state until the spot verification service verifies the corruption. Again, there is no user action required.
2. **Online scan needed**– When the spot-verification service confirms the corruption, it puts the file system in the “online scan needed” state. In the next maintenance window, an online scan is performed; there is no user action required. This state is reflected in the Action Center, so you can run the scan manually if you want to do that before the next maintenance window. The scan is run as a background operation, which means that you can continue using the computer while the scan is performed. During this online scan, all verified issues and fixes are logged for later repair. On Windows Server 8 systems, idle time is determined by monitoring the CPU and storage idle times.



1. **Spot fix needed**– The file system puts the volume in this state after the online scan is completed, if required, and this state is reflected in the Action Center. On client systems, you can restart the PC to fix all the file system issues logged in the previous step. The restart is quick (adding just a few additional seconds) and the PC is returned to a healthy state. For Windows Server 8 systems, a restart is unnecessary to fix corruptions on data volumes. Administrators can simply schedule a spot fix during the next maintenance window.



For more advanced users who want to avoid restarting their system to fix a non-system volume corruption, they can open the Properties dialog for the affected volume, and on the **Tools** tab, they'll see an option to check the drive for file system errors. Corruption on drives that are not currently in use can be fixed without needing a full restart of the computer.



Conclusion

In Windows 8, we have made the detection and correction of file system errors more transparent and less intrusive. We believe these changes will

be a welcome enhancement for you and we look forward to hearing your feedback.

-- Kiran Bangalore

Senior Program Manager, Windows Core Storage and File Systems

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

FAQ

Q) Will the new health model work on removable drives?

Yes, this works on removable drives that report fixed media, like most external hard drives.

Q) How do I enable the new file system health model?

You don't need to do a thing—the new file system health model is enabled by default.

Q) Will the new file system health model apply to Windows Server?

Yes, the health model is identical for both server and client. One thing that will be different by default is that the data drives will not be checked or fixed during boot of the system—this maintenance will be left to the administrator when time permits.

Q) Can I move between Windows 8 and Windows 7 and not affect the file system health model?

Yes, the file system health model will adapt to whichever operating system version it is mounted on.

Q) Will ReFS need to run chkdsk?

ReFS follows a different model for resiliency and does not need to run the traditional chkdsk utility.

Q) Will I ever need to run the old chkdsk /f?

There are cases where failing hardware can produce such severe corruption as to make the file system un-mountable; in these cases, you should perform a full, offline chkdsk to fix the file system. If for some reason this fails, we recommend that you restore from a backup.

Q) Is a reboot absolutely required to fix non-system volumes?

No, but the Action Center generally provides the simplest experience. If you're an advanced user, you can fix non-system volumes by opening the properties of the drive, or by running `chkdsk /scan <volume>:` and `chkdsk /spotfix <volume>:` from the command line.

Q) I run chkdsk /f often to check the status of our drives, is that needed anymore?

No, the system will inform you when a corruption is found, and you can then choose to run the `chkdsk /scan` to detect all the issues. An online `chkdsk /scan` will not take away from the availability of the drive or system.

Q) I run read-only chkdsk today to check the status of our drives; do I still need to do this?

No, we recommend you run `chkdsk /scan` instead, since this will also perform all possible online repairs and will also prepare for a `spotfix`, if needed.

Keeping your family safer with Windows 8

Steven Sinofsky | [2012-05-14T11:00:00+00:00](#)

*One of the intrinsic capabilities of Windows 8 is the ability to use multiple accounts on any PC. This makes it much easier for parents to use tools that can help protect their children from content on the Internet as they see fit. It is also a great way for each family member to maintain their own unique online identity while still sharing a single PC. Microsoft has been a leader in creating tools to help maintain a safe computing environment for all users as well as for parents in particular. With Windows 8 we have substantially improved the family safety features and services available. **Phil Sohn, the senior program manager lead for Family Safety, describes how Family Safety features will work in Windows 8.***

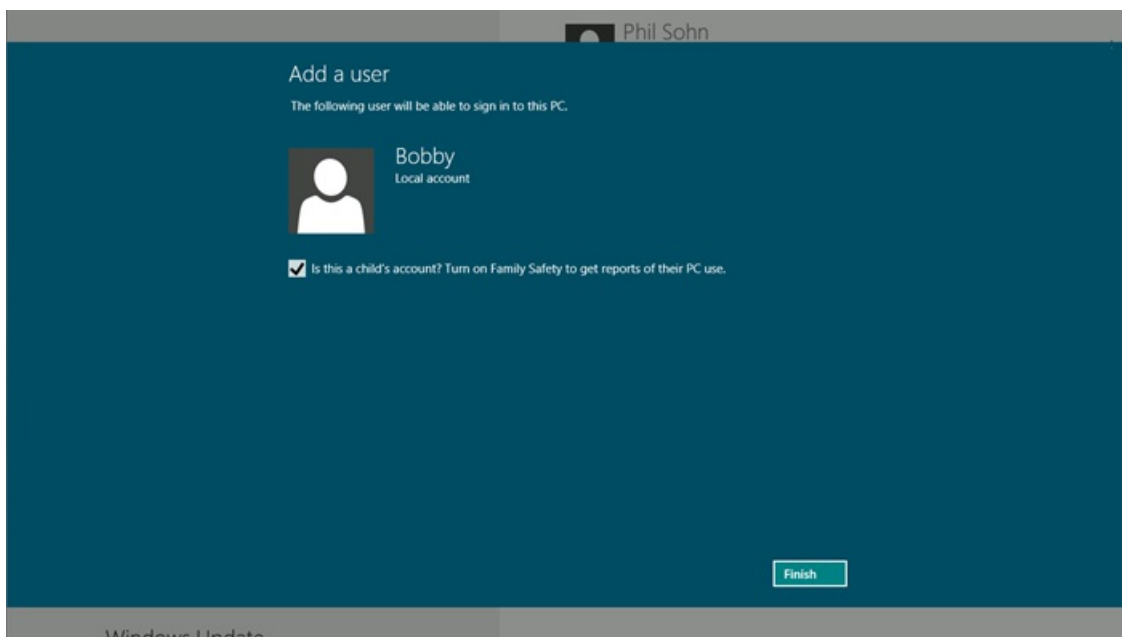
—Steven

Kids today grow up online. They use computers to do their homework, play games, communicate with friends, and access the wealth of information on the web. Computers give children access to many positive experiences; however, parents face challenges in monitoring what their children see online, the people they meet, and the information they share.

At Microsoft, we want to help parents create a healthy computing environment for their kids. We encourage parents to talk to their children about online safety and to set guidelines for their computer use. Microsoft and many safety advocates also recommend moving the family computer to a common room in the house so parents can glance over their kids' shoulders to gain a better understanding of their online activities. Parenting techniques like this are important, but they may be difficult to employ if your household has multiple PCs or if your kids use laptops and tablets. And glancing over a teenager's shoulder can be awkward for both parents and kids.

A safer Internet is just a click away

With Windows 8, you can monitor what your kids are doing, no matter where they use their PC. All you have to do is create a Windows user account for each child, check the box to turn on Family Safety, and then review weekly reports that describe your children's PC use. No additional downloads, installation wizards, or configuration steps are required. Just check the box!

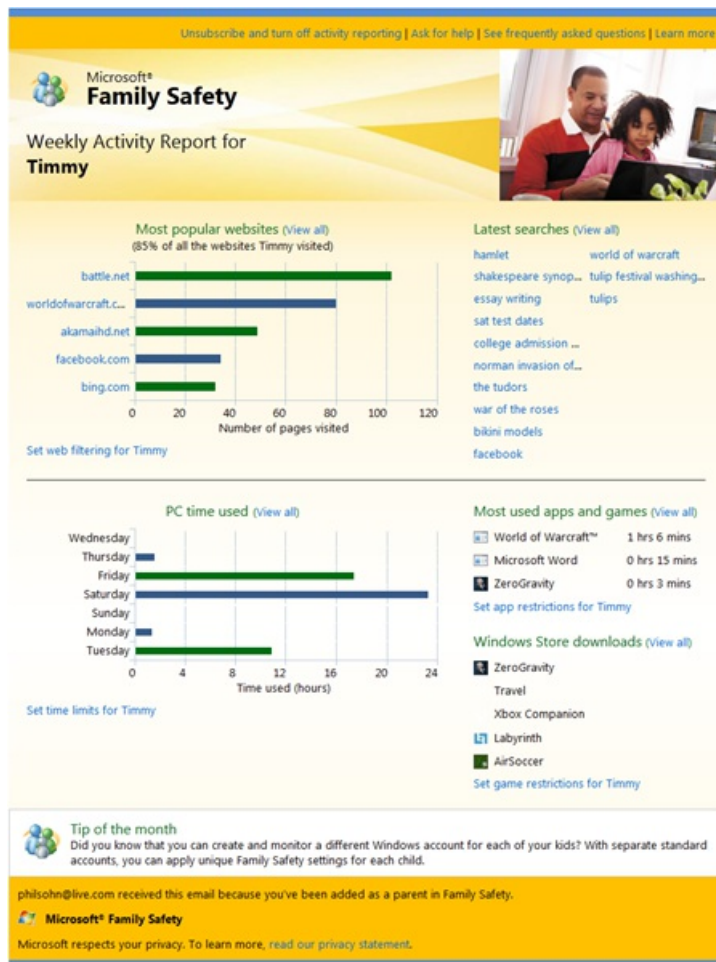


The “monitor first” approach

In the past, many of the industry software solutions for family safety (including Microsoft's) focused on web filtering and other software-based restrictions. This resulted in a more complex setup experience and a constant stream of parental approval requests that could be difficult to manage. The end result was that many parents abandoned family safety products and returned to in-person supervision only—a tactic that has become less effective as computers have gotten more mobile.

Windows 8 gives you a “monitor first” approach, which provides informative activity reports for each child. As previously discussed on this blog, [signing in to Windows 8 with a Microsoft account](#) makes setup much simpler: just create a separate user account for each child and then check the box to turn on Family Safety. As soon as you do, you'll receive a welcome email followed by weekly email reports summarizing your child's computer activities. We expect you'll find activity reports a great tool for teaching your kids about responsible computer use. Of course, you can also easily add restrictions by just clicking a link in the activity report. With the simplicity of activity reports, we believe more parents will adopt Family Safety, resulting in a safer computing environment for children.

Here's what a Family Safety activity report looks like:



With a Microsoft account, you can take action from anywhere, on any device, because the reports are delivered directly to your email inbox. Any changes you make to Family Safety settings are stored in the cloud at familysafety.microsoft.com. These changes are then automatically applied to all Windows PCs where Family Safety is active.

Standard accounts for the kids

We've long recommended that parents log in as the computer administrator and make sure children have separate standard accounts. In Windows 8, accounts that the administrator—or "parent"—creates are automatically created as standard accounts. This approach has several benefits. Children:

- Won't be able to access their parent's email, online accounts, documents, etc.
- Can customize their own account settings without affecting their parent's account
- Won't be able to download malware or other questionable files because the [SmartScreen Application Reputation service](#) automatically prevents it

For parents who want more control

Activity reporting, which is on automatically in the new Family Safety, is the perfect solution for many parents. However, if you like more control, you can set up more powerful and customizable restrictions directly from links in the activity reporting email, or on familysafety.microsoft.com, if needed. In addition to the restrictions currently available in Windows 7, we've added some new ones in Windows 8, including:

- **Web filtering:** You can choose between several web filtering levels.

Web filtering for Bobby

Turn on web filtering Turn off web filtering

Allow list only

Only allows websites that a parent has added to the Allow list.

Child-friendly

Also allows websites in the child-friendly category. Blocks adult sites.

General interest

Also allows websites that are of general interest. Still blocks adult sites.

Online communication (basic)

Also allows social networking, web chat, and web mail. Still blocks adult sites.

Warn on adult

Allows all websites but warns when the sites contain suspected adult content.

- **SafeSearch:** When web filtering is active, SafeSearch is locked into the “Strict” setting for popular search engines such as [Bing](#), [Google](#), and [Yahoo](#). This will filter out adult text, images, and videos from your search results.
- **Time limits:** With Windows 8, you now can restrict the number of hours per day your child can use their PC. For example, you might set a limit of one hour on school nights and two hours on weekends. This is in addition to the bedtime limits currently available in Windows 7.
- **Windows Store:** Activity reports list the most recent Windows Store downloads, and you can set a game-rating level, which prevents your children from seeing apps in the Windows Store above a particular age rating.
- **Application and game restrictions:** As in Windows 7, you can block specific applications and games or set an appropriate game rating level.

Here is a short video showing how Family Safety works in Windows 8:

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

Watch for Family Safety in Windows 8 Release Preview

We are continually striving to help you create a safe, family-friendly computing environment for your kids, but of course, we know that this means different things to different parents. Some parents prefer to simply keep an eye on their children. Others prefer to set up software restrictions on their child's computing activities. We think the simplicity and power of the “monitor first” approach in Microsoft Family Safety addresses either style effectively and will lead to more family conversations about online safety, a safer computing experience for kids, and increased peace of mind for parents. Watch for these Family Safety features in the Release Preview.

--Phil

Delivering reliable and trustworthy Metro style apps

Steven Sinofsky | [2012-05-17T08:30:00+00:00](#)

*As we developed the app model for Windows 8 and the new Metro style apps, a key architectural requirement has been to deliver apps to customers that can be used with confidence—confidence that apps will be well-behaved with respect to resources, that apps will not interfere with other apps, that apps use system resources with your permission, that apps can be installed and uninstalled with ease, and so on. These attributes require a robust platform and strong set of tools for developers. This is an effort that requires a fresh start and cannot be retrofitted on an existing system. Windows 8 is a fresh start in this regard. This post details some of the work we have done at the platform level to deliver reliable and trustworthy Metro style apps. **This post is authored by John Hazen, a program manager on our Developer Experience team.** —Steven*

One of our core principles in the development of the Windows 8 Metro style app platform was to ensure that users would have confidence in their apps. This is a mission we're in together; in this post, I explain our vision for app confidence and reliability and help you build confidence by design into your apps.

Let me start by explaining what we mean by *confidence*. Picture a customer browsing the Windows Store looking at a Metro style app; we want them to be thinking only about the app and whether or not it is right for them. We want them to assume—in fact be confident—that the app will behave the way they expect and thus will perform well on their system, will use only the data and information they authorize, and will harmoniously co-exist with their other applications.

Our goal with the platform is to help us all build great apps that embody this vision of confidence so that we get confidence by default. To that end we made investments throughout the system. Here's how we picture it:



App Confidence: Windows 8 SDK for Metro style apps, Windows App Certification Kit, App Signatures, App Container, Ratings and Reviews, Store Onboarding, Frictionless Install, Telemetry Feedback

This post covers these areas and towards the end goes into depth on our app capabilities. First, a quick overview:

- **Windows Store** – For customers, it starts with the Store, their one-stop-shop for Metro style apps. To get into the Store your app is reviewed for both technical and policy compliance, including security checks. After it's published to the Store, your app will be rated and reviewed by the community. Together, the onboarding process and community reviews help create an environment in which customers can try apps with confidence.
- **App install** – Windows 8 handles all the details of deploying apps on your behalf so your customers don't have to worry that installing, updating, or removing one app will adversely affect other apps.
- **SDK** – The Windows 8 SDK for Metro style apps provides a well-defined set of APIs that help you build reliable apps that conform to the Store onboarding requirements and provide the best experiences for your customers.
- **App container and capabilities** – Windows 8 provides a greater degree of separation between apps than was possible with traditional desktop apps, so you can build apps that interact with each other in more predictable ways, giving customers a more consistent experience.

We of course recognize that any onboarding process can be gamed, API sets can be abused, SDK limits cleverly avoided, and that app containers are not impenetrable. But we are confident that the investments we made in this new ecosystem will help you build apps that delight customers. This multi-dimensional approach is the most effective way to build customer confidence and we will improve each dimension over time as we learn from experience. Now let's talk about confidence.

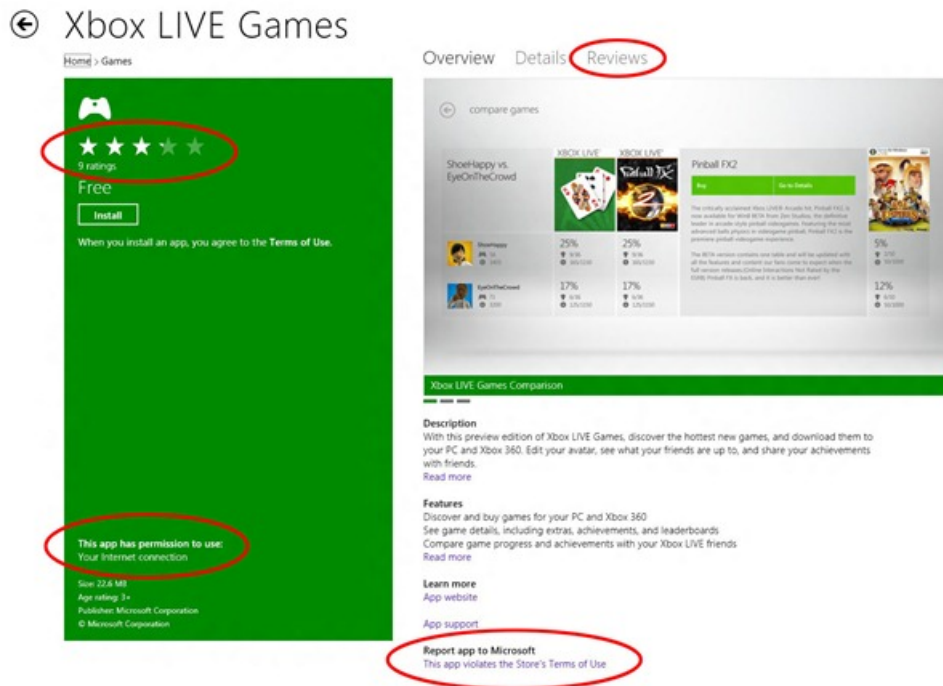
The Windows Store

Several folks on the team have shared quite a bit on the [Windows Store Blog](#) about the overall design and plan for the Store. If you have not had a chance yet, read about the Store, because it plays a central role in helping you connect with your customers. Let's look at a few ways in which the

Store helps build confidence in the app ecosystem.

First, the Store onboarding process establishes a consistent baseline for app quality and reliability. The technical conformance tests, contained in the [Windows App Certification Kit](#), help you know that you meet the expected standards before you submit your app. As you and other devs continue to deliver apps that meet these standards, customers will be excited to discover and try out new apps, creating a stronger app ecosystem for all of us.

The Store also makes it easy for customers to provide ratings and reviews that will help the best apps stand out, enticing even more customers to try out the top rated apps with confidence. In the sample app listing page below, note the highlights not just for ratings and reviews, but also two other important elements of the app listing page. Prior to installing a new app, customers can see what permissions the app has once it is installed (more detail on this later in the blog). After installing an app if the customer has concerns about app content or behavior, they can easily report their concern and we can follow up with you to address any problems identified. Helping customers decide which apps best meet their needs and allowing customers to provide input and feedback is an important way to improve overall confidence in the Windows 8 app ecosystem.



App listing page from the Windows Store

Finally, the Store collects telemetry data that can help you investigate problems your customers see, including the number and types of crashes they suffered. We review this automated telemetry for indications that an app's behavior is unreliable or inconsistent with the expectations of our customers. Our goal is to make effective use not only of the individual feedback that customers provide on apps through ratings and reviews, but also provide insight into how your app is actually behaving on customers' machines and give you the opportunity to improve your app based on this data.

In all these ways, the Windows Store is your partner in connecting you with people who want your apps, and helping them have a great experience with your apps.

Apps are just a click away

When you have your customer's attention on the app listing page, you don't want anything to get in the way of your customer getting your app. On Windows 8, getting an app is a matter of clicking a single button. Customers no longer have to wade through a series of questions or click button after button. After logging into the Store, when they find something they like, they just click the button, confirm the purchase, and go!

The great news for you is that you don't have to write a single line of installation code to make this magic happen; it is all provided for you as part of Windows 8. Not only is installation handled, but Windows uses digital signatures to ensure the integrity of your app all the way from the Store to installation and even when the app is loaded and running on your customer's computer. If Windows detects that the app no longer matches its digital signature, it guides the customer to download a corrected version from the Store.

Because Windows installs each app in a discrete location with separate and private locations for each app's data and settings, customers don't have to worry that installing, or removing, one app will interfere with the behavior of other apps or their computer. Customers will be more willing to try more apps than ever before, knowing that installing and removing apps won't degrade their experience over time (in fact, with Windows 8 contracts, each app you install makes the experience better). You benefit by knowing that there is little another app might do that will damage your customer's experience of your app.

Having a single, verifiable, and consistent mechanism to install apps not only simplifies your work, it provides an easy and positive experience for your customers, giving them confidence to get even more of your great apps. You can learn more about app deployment in [App packages and deployment](#).

Windows 8 SDK for Metro style apps

The better experience customers have with your apps, the more readily they will try new apps and updates as you release them. The [Windows 8 SDK for Metro style apps](#) is a great foundation for you to build apps that customers won't hesitate to install.

We put a lot of thought into the API set we offer for Metro style apps, not only to simplify the Windows programming environment, but also to provide a well-tested platform on which you could confidently build your apps, knowing they will work well with the Windows 8 Metro style app model.

Resist the temptation to find ways to invoke APIs that are not included in the SDK. This ultimately undermines the expectations that customers have for your app. APIs that are outside the SDK are not guaranteed to work with Metro style apps either in this release or in future releases, so you may find that your app doesn't function properly for all customers. These APIs may also not function properly in the async environment that is foundational to Metro style app design. Finally these APIs may undermine customer confidence by accessing resources or data that Metro style apps would not normally interact with. For all these reasons, we have provided checks in the [Windows App Certification Kit](#) to help you catch places where you might have inadvertently called interfaces not exposed by the SDK.

While it is possible to hide or obfuscate calls to APIs that are not included in the SDK, this is still a violation of customer expectations and Store policy. In the end, we have created this platform to help developers like you to build amazing apps that work well with the system and with other apps and devices to delight customers. Working with the Metro style SDK is fundamental to your realizing that goal.

Working well together and apart

A clean installation process is important to your customers, and the SDK helps you build apps that integrate well with the system. Beyond this, customers expect a high degree of reliability from your app and our platform. We help you achieve your reliability goals by providing a greater degree of separation between Metro style apps than is possible for traditional Windows desktop applications. On Windows 8, each Metro style app runs in the context of a unique app container that helps insulate it and its data from other Metro style apps.

App containers provide a few characteristics that are shared by all Metro style apps. They:

- Provide a dedicated environment for your app, including your own store for data and settings. You have little worry that some other Metro style app will change your app's data, settings, or behavior.
- Help ensure that your app doesn't accidentally interfere with the reliability of the Windows platform itself, or accidentally use your customers' data or devices in ways they don't expect.
- Provide a well-defined way to extend the capabilities of your app through declarations you make in the manifest and disclose to your customer in the app listing page.

Having this degree of separation makes it far easier to write apps that are reliable and respectful of the user. At the same time, we all want apps that interact well with one another. Windows 8 provides several mechanisms for Metro style apps to work with each other and with the platform to provide these satisfying experiences, including:

- [App contracts](#), which are the glue that binds Metro style apps together and to the system UI.
- The [FilePicker](#), which allows your app to interact with data the user selects.
- [App capability declarations](#), which allow your app to programmatically interact with devices and data, when appropriate for your functionality.

These are all well-defined ways for your app to engage more deeply with other apps and the system. The app container exists to help you deliver on your customer's expectations of reliability and respectful use of their system and data. The constraints of the app container are designed to help realize customers' expectations for consistent and intuitive app behaviors, and using techniques that allow your app to run code outside of an app container is a violation of user trust and Store policy.

In our discussions with developers during this preview period, we have seen apps that have misunderstood or accidentally misused some of these mechanisms, so let's go into more detail about the app capabilities in particular.

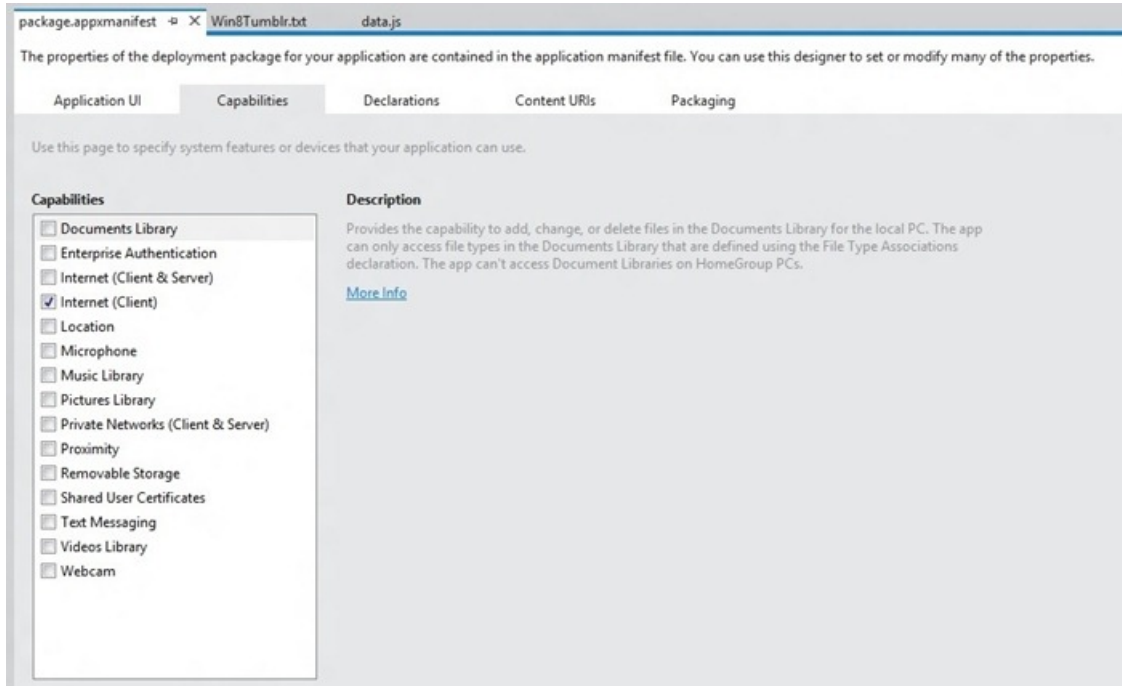
App capability declarations

The app container can be extended in a variety of ways using capability declarations, each of which is designed to enable certain scenarios. Therefore, we recommend that you use them only under certain conditions. These capabilities fall into 4 primary buckets:

- **Data libraries:** By default, apps have no access to the customer's data libraries, like the Music library, or the Documents library. We recommend that you use the FilePicker to interact with these libraries, but in some rare cases it is necessary for your app to be able to directly read and manage data in these locations.
- **Device access:** By default, apps can't use devices that most users consider sensitive for their privacy, including the webcam, microphone, and location. When apps need these devices, they must both declare their intent, and get consent from the user.

- **Network access:** By default, apps have no access to the customer's networks. Because most apps interact with the Internet, we enabled this particular capability in all the Visual Studio templates for Metro style apps. If your app needs more than just simple Internet access, you can read about your options below.
- **User identity:** These capabilities provide direct access to a particular customer's corporate logon info, or to certificates associated with their identity. These capabilities, although rarely needed, are necessary for certain enterprise apps, and you might need to use them in scenarios like banking transactions in which a smartcard might be required for authorization.

Declaring a capability in any of these categories is as simple as checking it off in the Visual Studio manifest designer. But add capabilities only if they are critical to realizing a scenario for an app. During the Consumer Preview, we saw several apps submitted to the Store that declared every capability or a capability that was not essential for the app. So, while the list of possible capabilities is relatively short, it is worth reviewing in more detail each of these capabilities and how to use them.



Visual Studio's manifest designer

Data libraries

These capabilities grant your app access to user data that wasn't necessarily created in your app. Users expect apps to be respectful of access to their private data. One way to honor the trust users place in your app is by declaring only the minimum access necessary for your app. In most cases you can avoid using these capabilities entirely by using the [FilePicker](#), through which the user can browse files anywhere on their hard drive or network. For example, use the FilePicker to provide a File open experience, or to add Save as to your app in order to give your user the opportunity to save content from your app into their library locations or to removable storage.

Manifest declaration	What it provides
musicLibrary	Provides the capability to add, change, or delete files in the Music Library for the local PC and HomeGroup PCs.
videoLibrary	Provides the capability to add, change, or delete files in the Videos Library for the local PC and HomeGroup PCs.
pictureLibrary	Provides the capability to add, change, or delete files in the Pictures Library for the local PC and HomeGroup PCs.
documentsLibrary	Provides the capability to add, change, or delete files in the Documents Library for the local PC. The app can only access file types in the Documents Library that are defined using the File Type Associations declaration. The app can't access Document Libraries on HomeGroup PCs.

removableStorage Provides the capability to add, change, or delete files on removable storage devices. The app can only access file types on removable storage that are defined in the manifest using the File Type Associations declaration. The app can't access removable storage on HomeGroup PCs.

During the Consumer Preview, we saw app submissions that declared these capabilities when they really didn't need to. For example, apps declared `documentsLibrary` for a variety of reasons including:

- Storing app-specific settings in the documents library. The private store is designed to provide this function. You can learn more about app settings and storage [here](#).
- Store a user-generated file. This is more properly accomplished using the `FilePicker` to allow the user to save the file to any location, including the documents library.
- Sharing a document with another app. The Sharing contract is designed for this purpose.

If your app is designed to be the primary handler on the system for a given file type, for example a Fax Viewer that needs to handle all `.TIFF` files in the user's documents library, then declare this capability.

Device access

Many devices, like orientation sensors and accelerometers, are available to any app. But most customers consider certain devices more sensitive than others, given that they are strongly associated with user privacy. There are a lot of great apps you can build on these devices. For example, if you have a casual game that allows shared gameplay, using proximity is a great way to establish a connection between devices.

Because these devices are closely coupled to user privacy, Windows 8 ensures that if you declare the capability, the customer will be prompted to approve this access the first time your app tries to access the particular device. For example, if your app offers mapping, you likely want to access the customer's geolocation data. The first time your app tries to get this info, the customer sees a prompt to approve your app's access. This approval sticks until the customer decides to explicitly remove the access through Settings at a later time. Because the customer can decline your app's access to these devices, even if you have added the capability to the manifest, design your app to handle the lack of access gracefully. For example, if the customer has denied your app access to the built in GPS, you could simply prompt the user to select their location from a map.

Manifest declaration	What it provides
location	Provides access to the current location, which is obtained from dedicated hardware like a GPS sensor in the PC or derived from available network information.
webcam	Provides access to the webcam's video feed, which allows the app to capture snapshots and movies from connected webcams.
microphone	Provides access to the microphone's audio feed, which allows the app to record audio from connected microphones.
proximity	Provides the capability to connect to devices in close proximity to the PC via near field proximity radio. Near field proximity may be used to send files or communicate with an app on a nearby device.

Network access

Most apps need an Internet connection, so the Visual Studio templates for Metro style apps include the `internetClient` capability by default. If your app doesn't need to communicate over the Internet, you should remove this capability. The `internetClientServer` capability is generally used in peer-to-peer (P2P) scenarios like gaming or VOIP, but unless your app must open a port in the firewall, don't use this capability. Use the `privateNetworkClientServer` capability when your app needs to communicate over private networks, for example between devices within a home, or over a corporate network connection.

Manifest declaration	What it provides
-----------------------------	-------------------------

internetClient	Provides outbound access to the Internet and networks in public places like airports and coffee shops. Most apps that require Internet should use this capability.
internetClientServer	Provides inbound and outbound access to the Internet and the networks in Public places like airports and coffee shops. This capability is a superset of internetClient. The internetClient capability doesn't need to be enabled if this capability is also enabled.
privateNetworkClientServer	Provides inbound and outbound access to Intranet networks that have an authenticated domain controller, or that the user has designated as either home or work networks.

User identity

Most developers don't need these capabilities. Use of these capabilities will be highly restricted and subject to additional onboarding policy and review. But there are cases where such capabilities are necessary and appropriate, for example some banks require two-factor authentication and need to allow customers to provide a smartcard that carries a digital certificate that confirms their identity. Other apps that are designed primarily for enterprise customers rather than consumers might need access to corporate resources that cannot be accessed without domain credentials.

Manifest declaration	What it provides
enterpriseAuthentication	Provides the capability to connect to enterprise intranet resources that require domain credentials.
sharedUserCertificates	Provides the capability to access software and hardware certificates, such as smart card certificates, for validating a user's identity. When related APIs are invoked at runtime, the user must take action (insert card, select certificate, etc.). This capability is not necessary if your app includes a private certificate via a Certificates declaration.

Building confidence

Customers want to safely enjoy Windows 8 and the apps you build. The Metro style app experience is designed to make it easy for you to build apps that everyone can try and buy with confidence. This sets up a constructive cycle where people love and buy lots of apps that then generates opportunities for developers to create and deliver even more great Metro style apps.

As I noted at the beginning, we're in this together; we are confident that the collective investments we made in this new ecosystem will help you build apps that people will be delighted with, and we look forward to partnering with you in delivering amazing new experiences to our joint customers.

--John Hazen

Creating the Windows 8 user experience

Steven Sinofsky | [2012-05-18T12:30:00+00:00](#)

*This blog often focuses on the bits and features and less on the “philosophy” or “context” of the product. Given the level of brand new innovations in Windows 8, however, we think it is worth putting Windows 8 in the context in which we approached the design. As with any [significant change](#) to a [broadly used product](#), Windows 8 has generated quite a bit of discussion. With millions of people using the Consumer Preview for their daily work, we’ve seen just as many points of view expressed. Many people—from [David Pogue of the New York Times](#) to [Mat Honan from Gizmodo](#) and many more—have been quite positive, and others less so, most notably in the comments on this blog, where we’ve seen the rich dialog we’d hoped for. Some have asked about design choices we’ve made in the product and the evolution of Windows or suitability of the design to different people. Some bloggers believe it is [critical to further separate the traditional desktop from Metro style elements](#). Other people believe passionately that it is [important to make the desktop more like the Metro style interface](#). There are as many opinions as there are folks who have tried out the [Consumer Preview](#). Designing a new release of a product already used by a billion people in a billion different ways is, as we say, like ordering pizza for a billion people. Doing so out in the open encourages this dialog, and we embrace and value it. **Jensen Harris, Director of Program Management for our User Experience team, authored this post.***

—Steven

At the **D: All Things Digital** conference in June 2011, we demonstrated for the first time [the new user interface](#) that we [developed for Windows 8](#). This new UI is fast and fluid to use, and optimized for mobile form factors such as laptops, tablets, and convertibles, where people spend the vast majority of their time today. Windows 8 works equally well with mouse, keyboard, or your fingers, and has the best pen support of any OS. It supports multiple displays and the widest array of configurations and form factors of any OS. On top of all that, Windows 8 introduces a new kind of app, which we codenamed “Metro style” following the design language that has evolved going back to Windows Media Center and the new Windows Phone. These apps are immersive, full-screen, beautiful, and optimized for the ways that people commonly use devices today.

I thought it would be useful to take a step back and describe a little bit of the background of how the Windows 8 user interface was designed, and discuss some of the decisions we’ve made and the goals of this new experience in more detail.

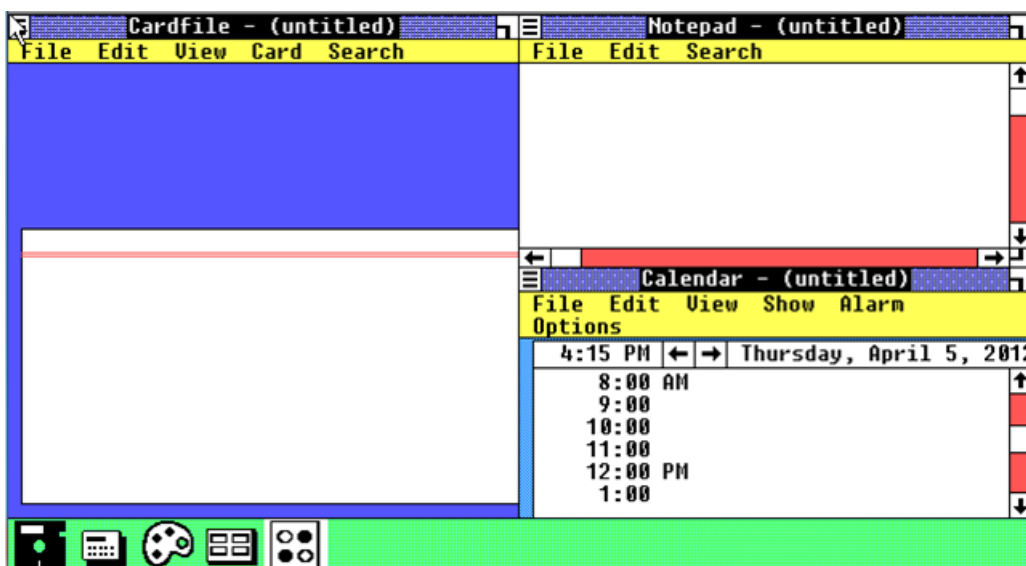
A brief history of the Windows user interface

The user interface of Windows has evolved and been transformed over the course of its entire 27-year history. Although we think about certain aspects of the Windows UI as being static or constant, the reality is that the interface is always changing to keep up with the way people use PCs. It is amazing to reflect back on the history of the Windows UI, and to see the level of dramatic change that has transpired over time.

Since Windows 8 marks a significant evolution of the user experience, I will focus on the releases where the user interface of Windows changed most significantly, and some of the initial perception surrounding those shifts. If you are interested, [a full history of Windows](#) is available to read on the Microsoft website.

Windows 1

Windows 1 was released in 1985, and it was designed for drastically different scenarios than what people use PCs for today.



The first version of Windows was a rough graphical shell around DOS, intended primarily to be used with the keyboard. A mouse was strictly optional and very few PCs had one.

In fact, the mouse was a bit of a curiosity at the time, perceived by many experienced users as inefficient, cumbersome, un-ergonomic, and hard to learn how to use. The mouse was certainly exotic. Do you roll it on the screen? Do you pick it up and [talk into it](#)?

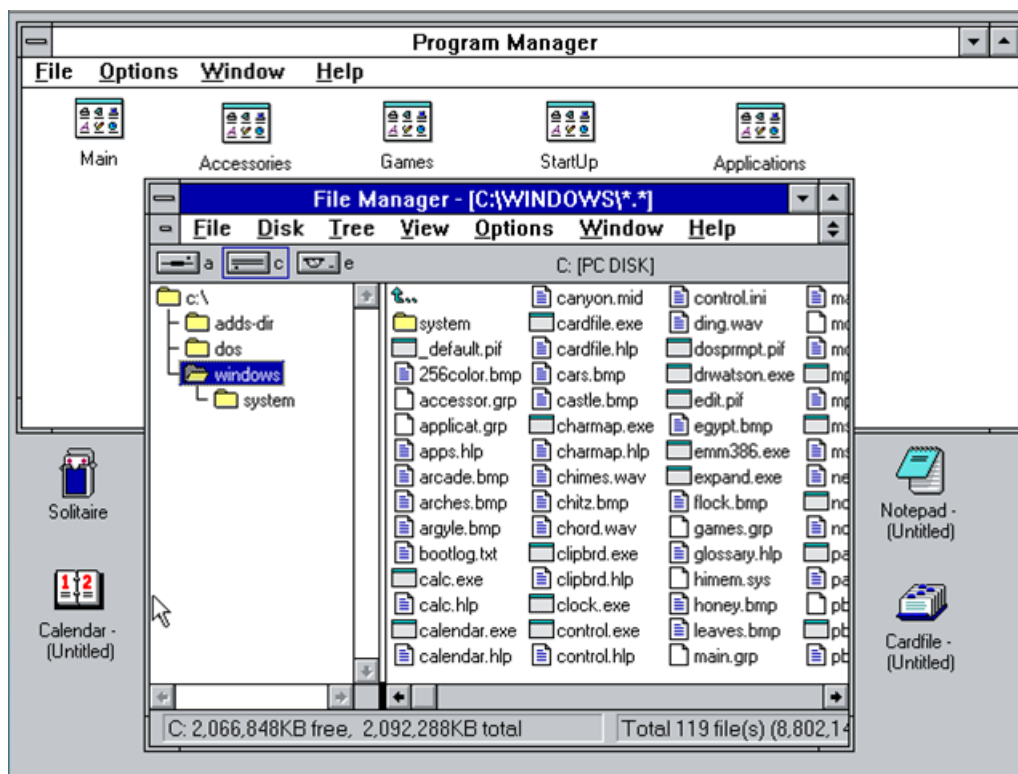
Here are a couple of published expert opinions from early 1980s print publications about whether the mouse would catch on:

- “Mice are nice ideas, but of dubious value for business users” (George Vinall, PC Week, April 24, 1984)
- “There is no evidence that people want to use these things.” (John C. Dvorak, San Francisco Examiner, February 19, 1984)
- “I was having lots of fun, but in the back of my corporate mind, I couldn't help but think about productivity.” (George Vinall, PC Week, April 24, 1984)
- “Does the mouse make the computer more accessible, more friendly, to certain target audiences such as executives? The answer is no.” (Computerworld, October 31, 1983)
- “There is no possibility that this device will feel more comfortable to the executive than the keyboard. Because of its ‘rollability,’ the mouse has the aura of a gimmick...” (Computerworld, October 31, 1983)
- “The mouse and its friends are merely diversions in this process. What sounds revolutionary does not necessarily help anyone with anything, and therein lies the true test of commercial longevity.” (David A. Kay, Datamation, October 1983)

So, as you can see, the mouse was considered gimmicky, unnecessary, and not useful for mainstream use. On the other hand, [some people are now asserting that the mouse is dead](#).

Windows 3 and 3.1

The first commercially successful version of Windows was Windows 3, released in 1990. It featured a totally new interface, centered on a new shell called *Program Manager* for launching, arranging, and switching programs.



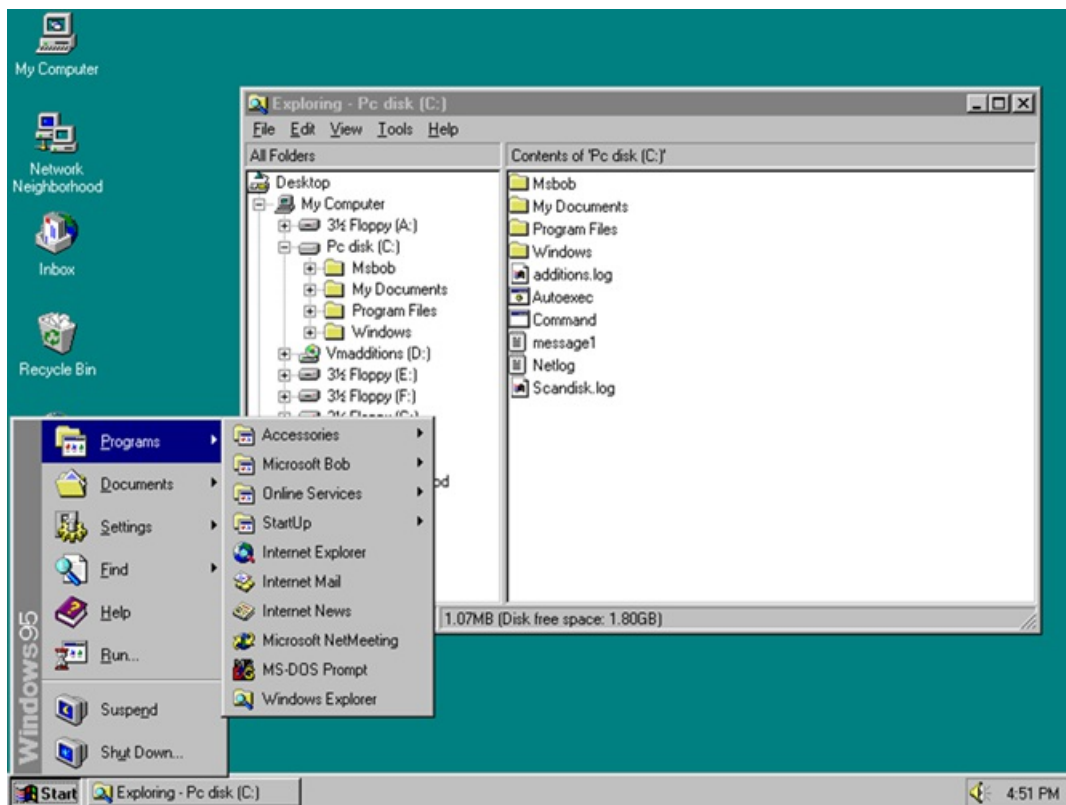
File Manager was the most important new program in Windows 3, used for managing files and drives. This upgrade bet big for the first time on most users having a mouse, and knowing how to use it to click on the colorful, large (for the time) 32x32 icons. Many reviews were critical of the release because to use it effectively required one of those off-criticized mice.

It is worth noting in the screenshot above that File Manager is being used to browse the files in the OS itself—something that was commonplace at the time, but now the modern equivalent of looking under the hood to repair an electronic fuel-injected car.

You could not put links to programs or files on the “desktop” in Windows 3. The area behind the floating windows was where programs went when you minimized them. Because getting to these minimized apps often required moving a bunch of windows out of the way first, the Alt+Tab keyboard shortcut became a very popular way to switch between running programs.

Windows 95

Windows 95, released a few years later in August of 1995, included a substantially reinvented user experience. Many of the constructs that are still present in Windows 7 were introduced in this version—the Start menu, taskbar, Explorer, and the desktop—but in very different forms.



Although we think about these user interface elements as familiar today, at the time, they were radically different from how anyone had used a PC before. The Start button was so undiscoverable that, despite having the word Start right on it, bouncing “<-- Click here to begin” text had to be added to the taskbar after early test releases so that people could figure out how to get started using the programs on their PC.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

Of course, once people figured out the “trick” of the Start button, it stuck with them, and then they were good to use it forever more. And of course, we were able to remove the “Click here to begin” text from subsequent versions.

At the time, PCs were still rather mysterious and the vast majority of homes were yet to have their first PC. Actions that we take for granted today, such as double-clicking and right-clicking, were to many users unknown—yet used extensively in the Windows 95 user interface. These actions proved problematic for many people to discover and master. Here is an historic video of a person in our usability labs trying an early build of Windows 95 which shows an example of the kind of problems many people had:

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

The Windows 95 user interface, designed in 1993, was forward-looking, and drastically simplified the common tasks of the time. Yet, a vocal subset of users continued to criticize it for years, preferring instead the comfort of what they were familiar with: Windows 3.1. Ed Bott from ZDNet [dug up some humorous posts highlighting the frustration some users were having](#) making the transition to Windows 95.

So, while the building blocks of today’s familiar Windows experience were designed in 1993, the world of 2012 is very different. In 1993, the web was still new to everyone, and most Windows 95 users had not tried it yet. The included media player could only play .wav files. Online connectivity was not commonplace, and if it was present at all, it was usually through a service such as AOL, which used a modem to connect to proprietary content and a closed intra-service messaging system.

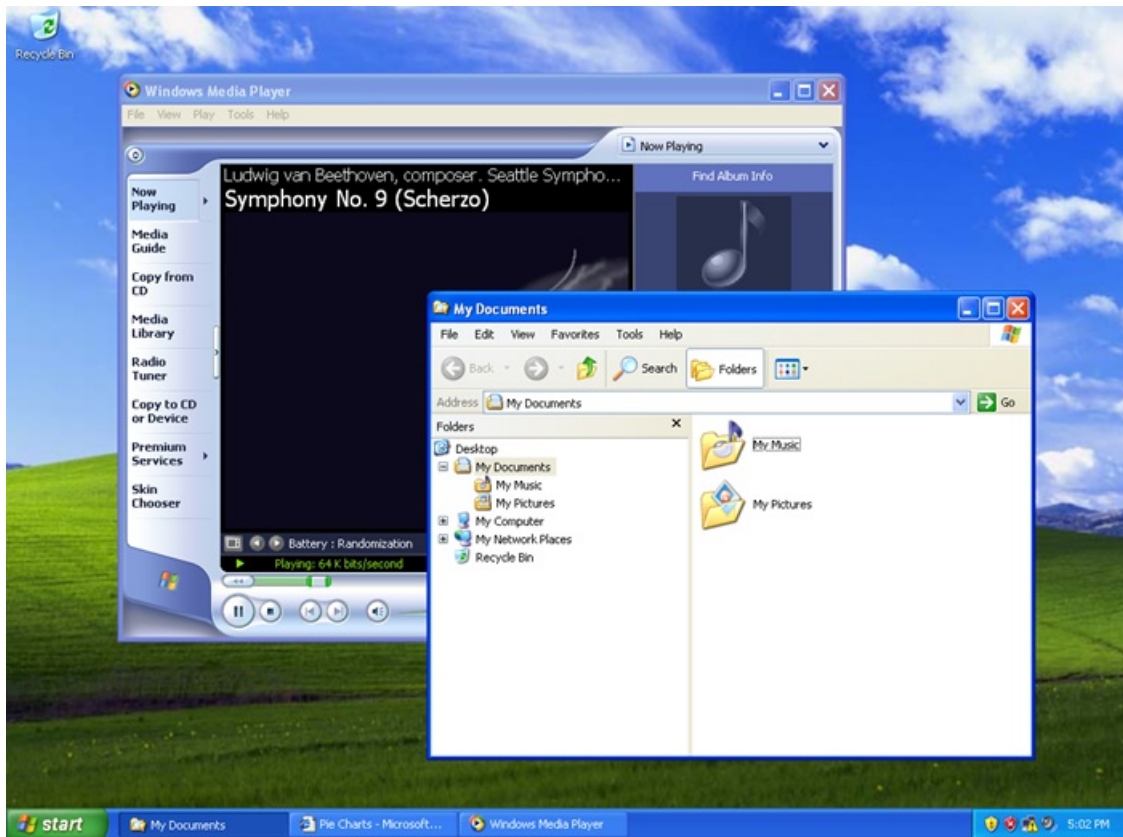
Windows 95 prominently featured a fax service and a terminal client/phone dialer. It did not include any support for commonplace devices such as digital cameras or portable mp3 players, since neither even existed as a consumer device in 1993. The first ever mobile phone with PDA capabilities, the [IBM Simon](#), was introduced around this time. It weighed almost 1.5 pounds, ran DOS, and the [only app ever designed for it sold only two copies](#). This was a very, very different world than the one we live in today.

And yet, the Windows 95 UI was designed in and for this world; the Start menu, the taskbar, the desktop, Explorer, and all of the rest of today’s familiar Windows UI was born in this time. A world in which you spent most of your time disconnected... typing in a word processor and manipulating files on your computer. A world in which the idea of not [starting Windows by typing “win” at the DOS prompt](#) seemed strange and almost unreal (and even the subject of many debates.)

From this distant time in the past, today’s familiar Windows user interface was born.

Windows XP

Windows XP was released to PC manufacturers on August 24, 2001. It represented another important evolution in the Windows user interface.



By 2001, people were using their PCs more every day. Typing and managing files, which doesn't require the web, remained a critical percentage of the time people spent using a PC. Yet collecting and consuming information and media—especially music, photos, and videos—was on the verge of becoming mainstream. (Even then, early digital camera sales were still [just one quarter of film camera sales](#), and would not eclipse them for another three years.) People were spending more time on the PC web browsing and doing mail, in addition to the document-focused productivity scenarios around which Windows 95 was developed.

Although you still invoked the Windows XP Start menu by clicking the word Start in the lower-left corner, the Start menu itself changed considerably. The familiar hierarchical programs list from all versions of Windows since 95, rooted in the old Program Manager, was demoted under an “All Programs” link. Mail, web browsing, pictures, and music were brought to the top level.



Although Windows XP eventually became a major success, some people at the time were frustrated with the changes to the user interface. They found the Windows XP experience [to be garish](#), and users [inquired about how to “downgrade” to previous versions](#).

Windows Vista

In 2006, Windows Vista substantially changed the visual appearance of Windows, introducing the Aero visual style. Aero gave the appearance of highly-rendered glass, light sources, reflections, and other graphically complex textures in the title bars, taskbar, and other system surfaces. These stylistic elements represented the design sensibilities of the time, reflecting the capabilities of the brand-new digital tools used to create and render them. This style of simulating faux-realistic materials (such as glass or aluminum) on the screen looks dated and cheesy now, but at the time, it was very much en vogue.



Aero was designed to help people focus less on the window chrome itself, and more on the content within the window. It draws the eye away from the title bar and window frames, and towards what is valuable and what an app is about.

And of course, the Start menu changed again, most notably by making it possible to press the [Windows key](#) (introduced in Windows 95) and then just start typing to search from anywhere in Windows. (This welcome innovation is one [we’ve kept in Windows 8](#), expanding it to search even within apps.)

Of course, as with every change along the way, [some people](#) expressed reservations [about the changes](#).

Windows 7

Windows 7 was released in the fall of 2009, and a number of the key aspects of the UI were significantly transformed. While many of these

changes centered on an overhaul of the taskbar, significant modifications were also made to the Start menu, [windowing](#), and to [the logical organization of files on the PC](#).



Notably, [launching and switching between programs were brought together in the new taskbar](#). Icons in the taskbar were made bigger and more touchable. The Start menu was changed to focus on launching only the programs you use less frequently, as no program can be pinned to both the taskbar and the Start menu. This marked the start of a transition where we were looking to remove the archaic distinction between starting a program for the first time and returning to a program that was already running. It is interesting to consider how odd it is that we trained ourselves to look one place for a program the first time it is running, and a different place once it is already running.

Windows 7 also was the first mainstream non-phone OS to [introduce multitouch support into the base OS](#). Although tablets on other platforms have followed suit, Windows 7 was the first shipping OS to embrace multitouch in the platform. Along the way, we learned a great deal about the limitations of trying to use touch to navigate Windows when so much of the existing interface, and virtually all of the existing programs, were specifically designed to be used with mouse and keyboard.

Although [some people had critical reactions](#) and [demanded changes](#) to the user interface, Windows 7 [quickly became the most-used OS in the world](#).

Trends that influenced the design of Windows 8

As we started planning the user experience of Windows 8 in mid-2009, just around the time of Windows 7 RTM, we looked around and took note of some of the trends playing out around us.

This was a pre-iPad world, a world before the recent proliferation of new form factors and device types. And [although more than 93% of PCs run some version of Windows](#) today, it was clear even then that the world we lived in and people's expectations of computing devices were rapidly changing.

Here are a few of the trends we noted that influenced the design of the Windows 8 user experience and features:

1. Connected all the time.

Connectivity is becoming ubiquitous. While today's file-centric Windows user interface was designed around assumptions of optional, limited, and sporadic connectivity, today, nearly everything people love to do on their PCs assumes they're connected to the Internet. Wi-Fi is assumed in more and more public locations, and an increasing number of PCs also include the ability to connect to mobile broadband networks as well. Where connectivity was once the exception, it is now the rule.

2. People, not files, are the center of activity.

There has been a marked change in the kinds of activities people spend time doing on the PC. In balance to "traditional" PC activities such as writing and creating, people are increasingly reading and socializing, keeping up with people and their pictures and their thoughts, and communicating with them in short, frequent bursts. Life online is moving faster and faster, and people are progressively using their PCs to keep up with and participate in that. And much of this activity and excitement is happening inside the web browser, in experiences built using HTML and other web technologies.

3. The rise of mobile PCs over desktop PCs.

The kinds of PCs people are buying are rapidly moving towards mobile form factors like laptops and tablets, and away from traditional desktops. While powerful desktops will remain the form factor of choice for people who want to squeeze every ounce of performance out of a highly modular and extensible PC (for example video editors, financial analysts, scientists, gamers, PC enthusiasts...), most people want to have light, portable PCs.

In 2009, desktops were 44% of the worldwide market and laptops were 56%. Just 3 years later, over 61% of the PCs sold are laptops and the trend is accelerating—this is globally, measuring all Windows PCs sold. Among consumers in the United States buying a PC this year, more than 76% will purchase laptops—the absolute number of all US desktops sold will be fewer than the

number of tablets in 2012! That is a fairly stunning change in the role of different form factors. Even in businesses, laptops are now purchased more than half the time.

Videos of the recent [Windows 8 Consumer Preview event we hosted in Barcelona in February](#) were shot, produced, edited, and controlled using only laptops. Many of these were very powerful laptops with secondary monitors plugged in for extra screen space, but even a few years ago we would have hauled around a truckload of desktop PCs for the event. Just because a PC is portable, light, and thin does not mean that it lacks the power or capability to do heavy-duty professional work.

4. Content is on the PC and in the cloud.

Following from ubiquitous connectivity and the popularity of laptops is the fact that people's content now spans the PC and cloud services. This includes not just purpose-built storage services like [SkyDrive](#), but also photos in Facebook and Flickr, videos put up for family to watch in Vimeo, music stored in and streamed from cloud services. All of this is augmented by GBs, or in some cases even TBs, of videos, photos, and music on the PCs in the home. People's content is spreading out everywhere, and as cameras are now high-resolution and always in your pocket (via your phone), the amount of content being generated every day is multiplying rapidly. A service like SkyDrive providing up to 100GB of cloud storage dramatically changes how you think about your PC and the resources you have access to.

These are a few of the key things we took note of in 2009. What all of these trends have in common is that people had started to use their PCs with different expectations and scenarios in mind. Although the PC remains the world's best tool for writing and typing and creating and making things, people increasingly were doing different kinds of things with the time they spent on their PCs. And they had started to expect PCs to behave more like their phones: connected, mobile, long battery life, centered on people and activities and keeping up with what's going on.

At the same time, apps have continued to get richer on mobile devices, as developers have had more time and experience developing apps. Along the way, mobile platforms continue to add APIs and functionalities that already exist in Windows.

We realized that to enable Windows to lead with these trends emerging, we needed to reimagine the Windows experience. Like so many other times in our history, we needed to bring the Windows experience forward: not only to better service what people are doing today, but to anticipate and cultivate the ways they will be using PCs in the future; to modernize the experience of using Windows, and to set the stage for the next decade of platform and developer innovation; to make the PC the most desirable, useful, and loved device in the world.

Windows 8 looks forward towards a new world of capabilities, new hardware, new apps, and new scenarios. Windows 8 is about a billion people doing new things, and the next billion people experiencing Windows for the first time.

Goals of the Windows 8 user experience

As we designed this new experience, a few clear goals emerged for the characteristics of what we wanted to create.

1. Fast and fluid.

Those of you who have followed Windows 8 coverage over the last year have undoubtedly [read or heard the phrase "fast and fluid."](#) This is not some "marketing" tagline we have recently created; these words are part of the [design language](#) we used to define what we intended as the soul of the new user experience in Windows 8. If Windows 8 were to be embodied in a phrase, this is it, and our goal is for this description to fit the product.

Fast and fluid represents a few core things to us. It means that the UI is responsive, performant, beautiful, and animated. That every piece of UI comes in from somewhere and goes somewhere when it exits the screen. It means that the most essential scenarios are efficient, and can be accomplished without extra questions or prompts. It means that things you don't need are out of the way.

It also implies to us a certain feeling of fluidity or weightlessness in using Windows. For instance, swiping from the edge of the screen with your finger to bring up controls feels fluid and natural and pleasing. The human finger is designed for that kind of motion! For example, dragging down from the top of the screen to close an app, or dragging a tile to the bottom of the screen to invoke zoom and then moving it to a distant part of the Start screen feels satisfying to do, in addition to being efficient.

2. Long battery life.

Because most Windows PCs are now battery powered (and soon the vast majority will be), great battery life is just a requirement. When the original Windows programming model was created, literally every PC was plugged in all the time. There was no concept of power management or battery drain. As a result, programs were free to do whatever they wanted. Once running, they ran constantly, regardless of whether you were interacting with them or not. Programs could consume all the memory on the system, or all the CPU, or write to disk every second. Basically they could, in a totally unbridled way, chew through your battery.

Traditionally, the design of PC software was centered on using the CPU as much as possible, whenever possible, because "[MIPS are cheap.](#)" In contrast, now we heavily scrutinize usage of the CPU, and understand the role it plays in preserving or reducing battery life. In a mobile world, this is a new type of engineering tradeoff. Where Microsoft used to primarily focus on reducing memory consumption, now we are also laser-focused on improving battery life while still delivering a fast and fluid user experience.

That means optimizing for memory consumption and CPU and GPU and performance and battery characteristics all at the same time, across a variety of platforms and hardware configurations. Therein lie the real engineering tradeoffs inherent in building a mobile OS, or just a modern OS that happens to be used on a mobile device.

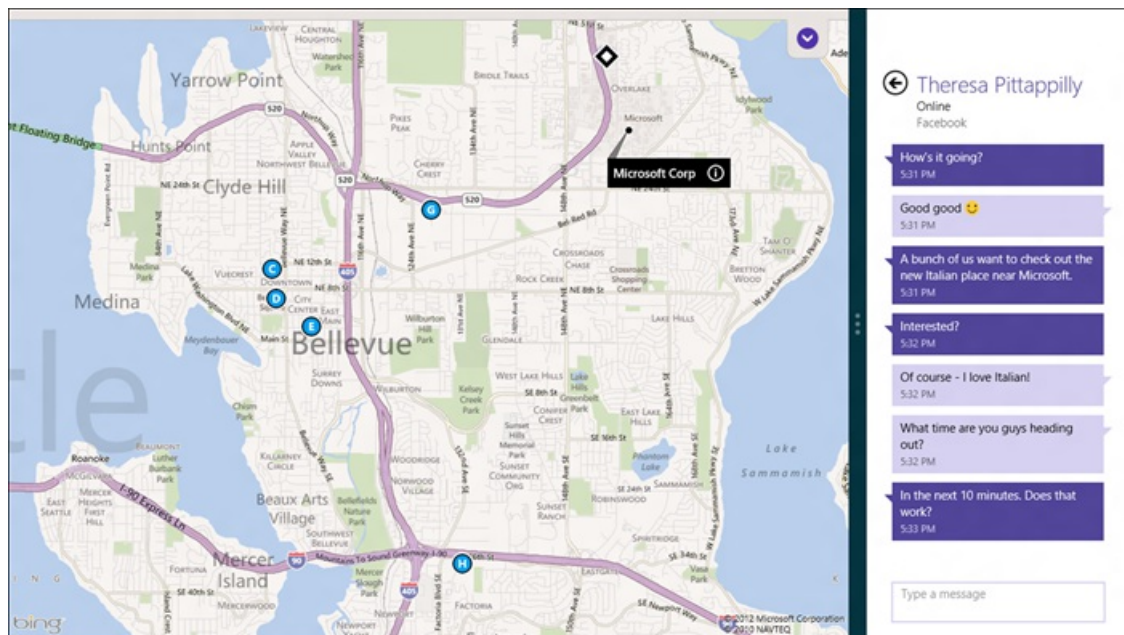
Once we understood how important great battery life was, certain aspects of the new experience became clear. For instance, it became obvious early on in the planning process that to truly reimagine the Windows experience we would need to reimagine apps as well. Thus, [WinRT](#) and a new kind of app were born.

To help extend a device's battery life, WinRT-based apps know how to save their state instantly. [Windows can throttle them down to use no CPU or memory on a moment's notice](#), but without the user losing anything they've been working on in the app. When the app resumes, it resumes in exactly the same place it left off. To the user, it has been running all the time—but technically the program has been suspended or terminated in the background.

There's a reason phones and tablets generally show only one app on the screen at a time. It is not just because of the traditionally small screens on these devices, or because doing one social update is all people do, or because "toy apps take up the whole screen." It is because "one-at-a-time" lets an OS manage the background activity on the device so that only apps you are actively using can drain the battery.

Even with multitasking in the existing desktop still present (and improved), we did feel like only offering "one-at-a-time" in the Metro style experience was a bit of a constraint, and not totally true to the Windows history of multitasking. So we evolved Snap for Windows 8. This feature lets you run any two WinRT-based apps side-by-side, so that you can watch a video while you browse the web, or video chat while checking mail. And we created [facilities for background processing](#) of a wide class of apps, and [background notification capabilities](#) that are unique to Windows as well.

In the below picture, you can see the Windows 8 Messaging app snapped next to the Maps app—two apps at once, even on a tablet.



3. Grace and power: Windows 8 apps.

Windows 8 apps are so much more than just optimizing for battery life though. These apps are beautiful and immersive, using every pixel of the screen to display their content. For years, each release of Windows added more and more chrome around the edges of your screen and windows—buttons and widgets and gadgets. Windows 8 reverses this trend, with Windows itself receding into the background, and putting the content of your apps ahead of the chrome.

Every app has a rich canvas with which to express its soul—when you're using a news app, you're fully immersed in news. When you're checking your social networks, content is presented beautifully and artfully and in ways that draw you in to spend more time enjoying yourself. When you play a game, you're fully and completely immersed in the game. (Although, some types of games have been doing full-screen for years!)

And over time, as apps evolve, when you're editing photos, writing a document, managing your finances, or any other professional productivity task, you'll also be immersed in that. Of course, if you use popular professional tools today, you can see how they are on a path to being full-screen and immersive already. Our unique app constructs such as [contracts](#), [pickers](#), and [many other cross-app capabilities](#) (all accessible globally via the [charms](#)) are about new ways to connect apps together, and are relevant for anything from intense data transfer between apps to sharing a quick link via email. These cross-app capabilities (and more importantly the APIs) are a unique element of Windows 8.

Windows 8 apps are purpose-built and tailored for the specific set of scenarios they are great at. This is different than traditional

desktop programs, which often contain hundreds of loosely-related, powerful, but hard-to-find features. Windows 8 apps focus on being great at something, or a few things, and really delivering a great experience for those targeted scenarios.

The history of development tools and system management tools shows that this approach of “[little languages](#)” (or “tiny tools”) vs. monolithic apps maps better to a world in which obtaining and using apps is easier. The rich capabilities of the Windows Store, contracts, and the searchable Start screen all were designed so that it is easy for people to have, find, and use many apps in Windows 8.

Even though apps express their individual personality and brand and content, they still do so [within the harmony of the overall “Metro style” design experience](#). Great Windows 8 apps [align to a common typographic grid](#) so that the PC feels fluid and harmonious as you switch between apps. We created standardized ways of doing common tasks: with touch, you swipe in from the edges to reveal commands. With a mouse, you move to the corners. The lower-left corner of the screen takes you to Start, no matter where you are. Right-click always reveals off-screen commands for the app you’re using. Within apps, Settings and Search and Share are always in the same location (the charms), no matter what app you are using. There is real value to having the consistent aspects of apps always work the same way. Yes, you do have to learn a few simple things up front, but once you know them, you know how to use the entire system.

WinRT apps scale gracefully from small 7” screens to large desktop monitors, to even larger wall-sized TV screens. Check out the Maps app in Windows 8 being used on an 82” touchscreen during the Consumer Preview launch event in Barcelona! (Or, [watch the video](#), starting approximately at 1:13:00.)



These apps were designed from the beginning to allow developers to target devices with extremely high-density displays. They are designed to work well no matter what input method you choose to use them with—mouse, keyboard, touch, or pen. WinRT apps are designed for the future, and for all of the ways that hardware devices and people’s expectations are evolving.

Because all WinRT apps come from the Windows Store, you can find and install them with confidence. The apps run in a local sandbox called AppContainer, so they can’t mess with or corrupt your PC. And they always uninstall cleanly, without a trace left behind.

4. Live tiles make it personal.

The heart of a new Windows 8 app is its tile. We know that people are increasingly snacking on snippets of live information. Who wrote on my timeline? Did I get any new email? Did anyone post pictures of yesterday’s party? Did anything big happen in the news? Who’s winning the game? Is my expense report approved? Did someone beat my high score? Is it my turn? When is my next meeting? Is a new book by my favorite author available for preorder? Is our inventory running low? What’s the traffic like?

Today, this is increasingly how we see many people use their devices, obsessively switching between different websites and programs on their PC and apps on their phone, checking to see if there’s anything new to see or do.

Tiles are designed so that you can see all of this information together in one place, with a single click, tap, or keypress from anywhere in Windows, without even opening your apps. It takes a bit of imagination right now in the Consumer Preview to fully visualize how this might work, because only a very small number of apps are currently available, and for many of these, developers are still working on building great tiles.

This is also an area where we are bringing together a set of disparate concepts and more strongly connecting them to the apps you actually use. Today, we are all familiar with a row of icons in the Notification Area near the clock, beckoning for our attention (each one using different interfaces, with different methods of control to silence them). These are separate processes running silently (and perhaps secretly) in the background, waiting to update us at inopportune times, and otherwise using system resources like battery

power. Gadgets, introduced in Windows Vista, held the promise of providing a more connected UI surface, but failed to relate to the apps and services we care the most about.

But as we move closer to general availability of Windows 8 and beyond, to a time when all of your favorite apps are available and represented by tiles, suddenly your Start screen will become a personalized dashboard of everything you care about. Your whole computing experience has the potential to be encapsulated in one view. A view that you organize and control.



Even content from within apps can be pinned to Start: people, mail folders, accounts, websites, books, albums, singers, movies, clients, sports teams, cities, etc. Everything you care about is efficiently available and up-to-date at all times. Tiles are the future and fit the way people look for fresh content in apps and websites. Just as yesterday's static highway signs telling you what you already know are being replaced by active and customizable message boards with road conditions, traffic alerts, and flexible lane usage, your PC should convey information that is current and up-to-date. Icons are yesterday's way of representing apps.

5. Apps work together to save you time.

I mentioned before that we have observed that people are increasingly spreading their time and content across an ever-wider cross-section of websites, cloud services, and apps. The result of this is that your stuff is strewn everywhere! Some stuff is stored on your primary PC. Other things are trapped inside apps or cloud services that you can only get to from within those apps or websites.

This leads to many common tasks being more complicated to complete than when, in the past, everything would be saved locally to your PC.

For instance, let's say that you are in a Skype call and you want to send a picture from a show you were at last weekend. Assume you took the picture with your phone and posted it to Yammer.

To do this today, typically, you would open your web browser, log in to Yammer, go to your main page, click on Images, find the photo and click it to select it, right-click it and choose Save As, put it somewhere on the hard drive, then switch back to Skype, choose "Attach", navigate to wherever you put the photo on your hard drive (hope you remember!) and then click it again to attach it. That task took at least eight steps to complete. And now you have two copies of the photo: one on Yammer, and a duplicate somewhere on your PC. It takes expertise and time to find that duplicate file, move it around, or ultimately delete it. What was once a simple photo sharing scenario has become laden with "file management" tasks.

There is a better way, and it is part of what makes Windows 8 apps so powerful. Windows enables any Windows 8 app on the PC to share data with any other Windows 8 app, even if those apps know nothing about one another.

Think about the scenario above again, except this time using Windows 8 apps for Skype and Yammer. From Skype, you click "Attach" and a picker with all of your local photos appears. But because you've installed a Yammer app, you can also instantly switch to pick between photos on Yammer. You click the photo that you want, and it is now attached in Skype. Done! That's only three steps—five fewer than the way it works today.

Although other OSs have attempted to streamline such tasks by hard-coding one or two currently popular services, Windows 8 is more useful, flexible, and future-proof. Our way is not limited to only a small set of specific, known services that are "baked into" the OS.

Any new Windows 8 app can pick from, share with, or save to any other installed app (and of course to the set of services that the app knows how to connect to.) It is a reinvention of how apps work in an OS, with Windows providing the "glue" that binds apps

together. Getting to your stuff, in any service, anywhere in the cloud, is just as easy as getting to that data on your local PC or home network—as long as the service builds a Windows 8 app. And with a reach of over a billion Windows users worldwide, we expect most services will see the value in creating an app for Windows 8.

6. Roam your experience between PCs.

Just like the experience of using most websites, you can sign in to your Windows 8 PC using an online account. The account used to sign in to Windows is called a Microsoft account. It can be an existing Windows Live ID (the email address you use for Xbox Live, Hotmail, and most other Microsoft services), or one can be created using any email address you own.

Once you are signed in, something magical happens—as you personalize and customize your Windows experience, the changes roam to any other PC.

Have a lock screen picture or desktop wallpaper you love? It's there on every Windows 8 PC you sign in to. Configure your settings, colors, and pinned websites just the way you like them? They move with you. Play the first ten levels of a game? You don't have to replay them again on your other PCs. Your saved passwords and favorites and language settings are all just there, whenever you sign in.

After you invest deeply in personalizing Windows, we don't want you to have to redo those steps on every PC you use. Just like if you changed a setting on your favorite website while signed in, you would expect that setting to persist no matter what device you signed in from. We want that same experience in Windows 8. And because roaming is part of the WinRT platform, any app developer can roam the settings for their app just as easily as Windows roams system settings. Roaming is not just for a single app or browser, but part of a platform that every app can easily use and everyone benefits from.

7. Make your PC work like a device, not a computer.

Today most people love their PCs, but it is clear that people's attitudes and expectations are changing for just about any device they carry around with them. People really want a product that just works. They want to sit on the couch and enjoy their favorite apps and games and websites and not worry about the vagaries of the registry or a million control panels or power profiles. They want to pick it up, enjoy using it, and then set it down.

In contrast, today's Windows is almost absurdly configurable. Even the most obscure features are often tweakable through a sometimes impenetrable labyrinth of control panels, group policies, special command-line utilities, undocumented registry keys, etc. Most of these settings are changeable not only by the user, but by any program that happens to be running on the PC that decides to "tweak" something. Much of what has been pejoratively called [winrot](#) over the years is due to overzealous downloaded programs overstepping their bounds and installing system services and updaters and background tasks and all sorts of things that slow down the system.

We recognize that in the proper hands, or in the hands of someone who is willing to tolerate the downsides, these are not features to be critical of, but assets of Windows. Our intention is not to lock down Windows, but to provide a platform that meets consumer expectations for how a device should work. These assets are far too easily abused or accidentally misused—there is a better way.

Our goal in Windows 8 is to redefine people's expectations of their PC. The most commonly used settings (those similar to the ones exposed on most phones or tablets today) are available within the new UI. New Windows 8 apps cannot alter system settings for the most part (with the exception of a few specifically architected capabilities, such as enabling location services or using the webcam, which require user consent.)

Windows updates are applied silently in the background and in the middle-of-the-night "maintenance window" whenever possible. Because Windows 8 apps know how to preserve their state, this is totally seamless to you.

On [SoC-based devices](#), you touch the power button to turn the screen off, and behind the scenes, your PC is immediately moved into a low-power mode. Press it again, and the device instantly wakes up. Windows 8 turns the PC into a device that delivers the kind of experience people expect out of a modern mobile device.

Now if you are an expert who really craves all of the traditional flexibility and customizability of all of the knobs and levers in the system, you can still access them just as easily as you could in Windows 7. These settings are still there, and they still work. The Control Panel and `gpedit.msc` and PowerShell and all of the other places you do expert customization of your PC are still there for you. People who don't have the knowledge to use these advanced settings effectively can just enjoy their devices. And for those who do want that power, it is there for them.

Although these seven goals were certainly not the *only* aspirations we had when designing what became the Windows 8 user interface, they give you some idea of the relationship between the trends we observed and anticipated, and how these observations directly mapped to the goals of the new UI.

Touch as a first-class input method (but not the only one!)

Windows has continually innovated to adapt to and enable new ways of working with the PC.

The earliest versions of Windows were designed to be used with a keyboard. Windows helped transition mainstream users to the mouse by [bundling a mouse with the first version of Microsoft Word, over 25 years ago!](#) This transition took quite a while, as many users were initially very skeptical of the mouse. “Real users only use the keyboard!” (Some might still say this. [The good news is, we have you covered.](#)) Of course now, all these years later, it is hard to imagine using a PC without a pointing device.

In 2001, [Microsoft announced Tablet PC](#), an “experiment” with a new kind of PC form factor, powered by the pen. We developed the best handwriting recognition in the market for certain languages. We pioneered ways to integrate [natural “ink” and drawing](#) into traditional programs like Microsoft Office. We experimented for the first time with slate PCs that had no keyboard. While the technology was not ready ten years ago to build light enough and quiet enough PCs with enough battery life to make this form factor widely compelling, clearly Tablet PC got a lot right in terms of predicting aspects of future computing.

So Windows originally had keyboard support, then added an assumption of mouse, then added the ability to use a pen. At each step of the way, these input devices were integrated into the core Windows UI without forgetting about or degrading the experience of the existing input methods.

Of course, some things had to change about how the user interface worked as each new input method was added (like, once you bet on a 2D grid of icons such as Program Manager did in Windows 3, keyboarding around with the arrows to launch an app becomes more cumbersome vs. traversing a simple list.) But it is fair to say that as of Windows Vista, mouse and keyboard were first-class input methods, with pen as a well-supported but secondary way of interacting with the PC.

Which brings us to the current day. In Windows 7, we introduced multitouch support into the base OS. Touch is an incredibly important long-term bet for us. For an increasingly large number of people over time, it will be the primary way they interact with Windows. And for the vast majority of users, it will eventually be used alongside mouse and keyboard to complete their experience.

In 2009 when we started planning Windows 8, touch was often ridiculed on phones—the rumors of an iPhone with a keyboard were prevalent and often [hopeful](#). It is almost quaint to look back at the speculation from many wishing and hoping for an iPhone with a slide-out keyboard.

Today, you would be hard pressed to find many people who still dream of a phone with a physical keyboard, though in a diverse world with diverse needs, even a small percentage of people represents a large absolute number.

Tablets, of course, don’t come with physical keyboards. But something is different about tablets—people still do desire a physical keyboard. We’ve all seen countless peripherals spring up that provide a keyboard for a tablet as a case or other accessory. Why is that? We see time and time again that it is because people want to use a tablet in place of their PC, and adding a keyboard is the best way to get more work done.

Even in the absence of software like Microsoft Office, the reality is that when you need to write more than a few quick lines of text, you yearn for something better than on-screen typing. Touch typing rates on glass are at best half that of a physical keyboard (and frequently much less), and so the extra time, energy, and thought needed to get the work done is a real issue.

Just as there were always people who could type large amounts of text with [T9](#), there are people who swear by multitouch typing as more than good enough for their work. Looking broadly, however, people benefit from the highly accurate, reliable, and fast user input enabled by a physical keyboard, and we think an OS and its apps should not compromise when one is available.

Beyond phones, touch has become the single most pervasive user model for a vast array of interactions—many of those powered under the hood by Windows PCs! From cash registers, to ATMs, movie rental kiosks, airline check-in, and grocery checkout, touch is literally everywhere. How would you explain to a 5-year-old that when she touches a laptop screen, nothing is supposed to happen?

To think that your PC would remain the single computing device you do not touch seems illogical. It is reminiscent of historic debates over the use of color back when PC displays were generally monochromatic; despite color being everywhere around us, many people believed color would be a distraction to work and should be reserved for play. (The Office team actually had a significant debate about the use of color icons in the first version that introduced toolbars.)

In a decade (or probably less,) we will look back at this transition period and say to each other “Hey, do you remember how PC screens didn’t used to be touchable? Wow, isn’t that weird to think about now?”

Designing for a successful touch experience

Some bloggers have written about how Microsoft invested in developing touch in Windows 7, but ultimately had a poor approach, as evidenced by the touch experience of both phones and tablets surpassing that of Windows-based devices. Going back to even the first public demonstrations of Windows 7, we worked hard on touch, but our approach to implementing touch as just an adjunct to existing Windows desktop software didn’t work very well. Adding touch on top of UI paradigms designed for mouse and keyboard held the experience back.

We took a step back and substantially changed both our approach and our implementation of touch for Windows 8.

Our approach to embracing touch in Windows 8 involved two parts:

1. Improving touch on the desktop.

We knew we needed to improve touch in the existing desktop using the feedback we received from touch users in Windows 7. We created larger touch targets, spread out controls a bit more, and added fuzzy targeting logic to make it easy to grab common controls

such as resizable window borders.

We resisted the temptation to make people choose between using mouse + keyboard OR touch. So many elements of desktop apps just assume people are using both a mouse and a keyboard, and no number of improvements we make to the touch experience on the desktop can fix what has been assumed and designed into these existing app interfaces. (After all, these programs have already been released to the market, in many cases a decade or more ago!)

However, we do believe that touch is a useful adjunct to mouse and keyboard on the desktop. Historically, a new input method is seamlessly integrated as people learn the best use for it. Context menus, keyboard shortcuts, toolbars, and menus are all different ways of doing the same thing, yet everyone makes their own choice about what works best for them.

Touch will evolve the same way. Having used a touch-enabled laptop every day for the last year (a Lenovo x220 tablet), I have a hard time imagining *not* being able to touch the screen for scrolling, or to tap the OK or Cancel buttons in a dialog box. Whenever I use a non-touch laptop, it is as if I've forgotten how to use the PC. Of course touch is not the primary way I use this laptop, but it is a crucial piece of how I interact with it. Even on the large-screen monitor I use at work, I just instinctively touch it—I don't think "because this screen is attached to a desktop PC, I must not be able to touch it."

2. Creating an environment exclusively or primarily suited for touch input.

Within the new UI and WinRT apps, touch is promoted to an equal citizen alongside mouse and keyboard. Just like you can use a PC with mouse and keyboard only (or just keyboard,) you can also have a great experience using the UI with just touch. In other words, we aspired to design a user experience that is new, worked for touch-only devices as a first and only input method, and when a mouse and keyboard are added, these can be used exclusively or with touch. Keyboard shortcuts are there alongside gestures—you pick based on your preference and the capabilities of your PC.

Many have opined that touch can have no role in certain form factors—we're all familiar with those quick to make comments about gorilla arms, fingerprints, poking at a screen, and so on. Many comments with this same tone were at the foundation of initial negative reaction to the mouse—"it makes me move my hand from the home row on the keyboard," "I get sore wrists," "it takes my focus off of my work while looking up at the screen," etc.

While the ergonomics of a [tablet placed in a dock with a keyboard](#) is similar to that of a touch laptop, there is no doubt that touch is new and different in a laptop and desktop. But when you consider that we do not think it has to be used exclusively, it starts to look only like a benefit when it is there. (And our design does not assume it is always there—although we think you will learn to miss it when it is not.)

We designed Windows 8 to take into account the desire to have a PC that works the way you do—whether you want a laptop with a permanent keyboard, a tablet with a keyboard you can attach (wired or wireless), or something in the middle. Touch works across all of these form factors, and you choose which input method to use when. This is what we mean when we say Windows 8 provides a no-compromise experience.

Metro style and the desktop: working together

Most of this post thus far has discussed some of the ways we designed Windows 8 in response to the trends we observed: the popularity of laptops and tablets, and the corollary expectation of excellent battery life; a focus on people and activity as the center of attention more than just files and documents; the ubiquity of cloud services; and the upcoming universal prevalence of touch on every PC.

So what is the role of the desktop in Windows 8?

It is pretty straightforward. The desktop is there to run the millions of existing, powerful, familiar Windows programs that are designed for mouse and keyboard. Office. Visual Studio. Adobe Photoshop. AutoCAD. Lightroom. This software is widely-used, feature-rich, and powers the bulk of the work people do on the PC today. Bringing it forward (along with the metaphors such as manual discrete window sizing and overlapping placement) is a huge benefit when compared to tablets without these features or programs. It is an explicit design goal of Windows 8 to bring this software forward, run it better than in any previous version of Windows, and to provide the best environment possible for these products as they evolve into the future as well.

We see our approach validated time and time again. On one hand, the makers of tablets and phones are in a race to add "PC capabilities" to their devices: support for peripherals like printing, remote access, high-resolution screens, or classes of new APIs for developers that already exist in Windows. At the same time, we also see consumers demanding features in these platforms that have existed for years in Windows—from things as mundane as full support for the keyboard and mouse, to things as complex as support for multiple monitors, background processing, or third-party accessibility tools.

On other tablet platforms, there has been significant customer interest in apps to bring the Windows desktop, running software like Office, to touch devices. These solutions use over-the-network remoting technologies to send pictures of the screen and touch input back and forth between the tablet and a real Windows PC. Of course, because these tablet devices don't natively support Windows software or a mouse, and because they require uninterrupted network connectivity, the experience is suboptimal—subject to frustrating lag, pixilation, and disconnections from the host PC.

We do not view the desktop as a mode, legacy or otherwise—it is simply a paradigm for working that suits some people and specific apps. This is very much like the person who uses a mobile "phone" but really uses it for the mobile browser and mail client and rarely uses apps or the phone. It

is like the person who has a brand new tablet but only uses the web browser.

The desktop is a great way to work with mouse/keyboard and a large monitor or several monitors. It is a powerful and flexible paradigm, allowing for pervasive control over the size and layout of windows on the screen.

If you only want to “live in the desktop,” if you never plan on using a PC with touch or using any apps from the Windows Store whatsoever, [Windows 8 still has a lot to offer](#). The Windows 7 desktop experience has been brought forward and significantly improved, with additions such as the [new Task Manager](#), [new Explorer](#) and [file copy UI](#), [Hyper-V on the client](#), multi-monitor taskbar and wallpaper, etc. And all in a package that uses fewer system resources than Windows 7. The new Start screen is simply a continuation of the Windows 7 trend of unifying disparate elements of the user interface—starting, launching, switching, and notifications.

It is really your choice. You can use only desktop apps if you want. You can use only new apps and never leave them if you want (in which case all of the desktop code is not even loaded.) Or, you can choose to mix and match apps that run in both environments. We think in a short time everyone will mix and match, simply because there is so much creative development energy being put into the new scenarios made possible by new Windows 8 apps.

Two devices, not three

Imagine a tablet. Light and thin. Amazing battery life. Gorgeous screen. You can lounge on the couch enjoying a beautiful, fluid experience, doing the things you love to do on a tablet: playing games, socializing, browsing the web, reading, touching up photos, watching TV. You are just immersed in your experience, doing the things you love to do. You hand it to your daughter and she knows exactly how to use it.

But then, if you want to have a bit more control and efficiency, you can set this same tablet in a stand and attach a keyboard, or just flip a keyboard around, and suddenly you have a complete Windows desktop experience, with full Microsoft Office, multiple monitors, peripherals, and a mouse.

Or, imagine a featherweight laptop with a beautiful large screen and a great keyboard. But in addition to doing everything you use your laptop for today, you can also use your favorite new apps built for today’s tablets.

Windows 8 imagines the convergence of two kinds of devices: a laptop and a tablet. Instead of carrying around three devices (a phone, a tablet, and a laptop) you carry around just a phone and a Windows PC. A PC that is the best tablet or laptop you have ever used, but with the capabilities of the familiar Windows desktop if you need it. You may choose to carry a tablet, or you may choose a laptop/convertible, but you do not need to carry around both along with your phone. You never think about a choice, or fret over your choice of what to carry. Things just work without compromise.

Great hardware like this doesn’t quite exist yet, but it will be commonly available later this year. This is the promise of the Windows 8 experience. With a little imagination, you can start to see why this kind of device will change the way you think of a PC.

Updating the visual appearance of the desktop

Several bloggers have wondered about how much we would be changing the visual appearance of the desktop in Windows 8.

We have appreciated seeing people on various sites post screenshots of their proposed designs for “Metro-izing” the visual appearance of the desktop. It is exciting to see the interest and passion that goes into designing them!

We spent a lot of energy carefully considering how substantially to update the appearance of the desktop in Windows 8. We looked at many, many pictures, and considered hundreds of designs. Our primary goal was to bring visual harmony to Windows, while still preserving much of the familiar feel of the Windows 7 desktop and not sacrificing the compatibility of existing apps.

In the end, we decided to bring the desktop closer to the Metro aesthetic, while preserving the compatibility afforded by not changing the size of window chrome, controls, or system UI. We have moved beyond Aero Glass—flattening surfaces, removing reflections, and scaling back distracting gradients.

A couple of the considerations we thought through:

- While much of the Metro style UI uses white text on a colorful saturated background, the desktop in Windows 8 will continue to use black text on light-colored chrome, as in Windows 7. This choice was made to help preserve maximum compatibility with existing programs.

Since the release of Windows Vista (which introduced Aero Glass), many desktop programs have integrated with glass, making assumptions that they should custom draw dark text with a cloudy “blur” texture behind it to make their text readable.

Some of you may remember the substantial compatibility problems that arose when the system colors changed from light on dark (Windows XP) to dark on light (Windows Vista.) It took many years for these to be fully sorted out. We would prefer not to reintroduce these compatibility issues again in the other direction. So, “color matching” the new design on the desktop is not entirely feasible.

- But at the same time, we want desktop windows to continue to feel light and airy, and we want a chrome style that doesn’t distract from the content of the app. We talk about Metro style apps as being “chromeless,” (that is, no title bar, borders, or Windows UI surrounding them.) Desktop apps, on the other hand, have a lot of chrome. When you add up the cacophony of a bunch of these windows floating on the

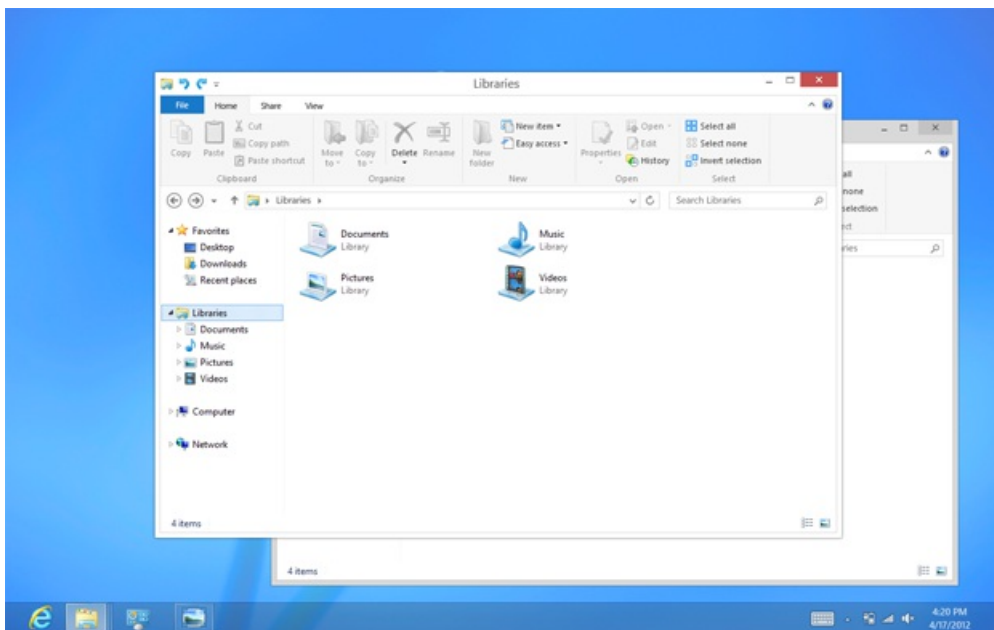
screen, suddenly you have a lot of chrome pleading for your attention. Aero was designed to [help the app's content to be the center of attention](#), and for the Windows system UI to recede into the background. This is still relevant today, and while we are moving beyond Aero, we don't want to lose sight of these goals.

- Visual compatibility with Windows 7. Windows 7 is the most popular and widely-used version of Windows so far. We made a conscious effort to relate the visual appearance of the Windows 8 desktop to the visual appearance of the familiar Windows 7 desktop. This helps people who want to predominantly use the desktop feel comfortable and immediately at home in the new environment.

We have made a number of improvements to the desktop visual appearance in Windows 8. Although we wanted the desktop to feel familiar, we also wanted to take some ideas from our new design language and apply them where we could.

We applied the principles of “clean and crisp” when updating window and taskbar chrome. Gone are the glass and reflections. We squared off the edges of windows and the taskbar. We removed all the glows and gradients found on buttons within the chrome. We made the appearance of windows crisper by removing unnecessary shadows and transparency. The default window chrome is white, creating an airy and premium look. The taskbar continues to blend into the desktop wallpaper, but appears less complicated overall.

To complete the story, we updated the appearance of most common controls, such as buttons, check boxes, sliders, and the Ribbon. We squared off the rounded edges, cleaned away gradients, and flattened the control backgrounds to align with our chrome changes. We also tweaked the colors to make them feel more modern and neutral.



While a few of these visual changes are hinted at in the upcoming Release Preview, most of them will not yet be publicly available. You'll see them all in the final release of Windows 8!

How will people learn to use Windows 8?

As people have tried the Consumer Preview, some folks have publicly asked questions about “learnability.” The new UI introduces a few new concepts to the PC: in particular, swipe from the edge (for touch) and move to the corner (for the mouse.)

Neither gesture works perfectly in the Consumer Preview—it should be expected that some things will not be perfect when we effectively design and test the product in the open like we do. The corners are too fragile to target reliably with the mouse right now, and it is too easy to frustratingly “fall away” from them. We have already significantly improved this in internal builds. And today's touch hardware, which was designed for Windows 7, doesn't always do a great job of interpreting swipes from the edge. The good news is that hardware designed for Windows 8 will have excellent edge detection, and our device manufacturer partners have been working on this for a long time.

So, the gestures themselves will work more consistently, and will be better-tuned than what is in the Consumer Preview. But how will people learn to use them?

We will post more about learnability soon: about how people discover and understand new concepts, and the specific steps we will be taking to make sure that people don't feel lost the first time they sit down with a Windows 8 PC.

But fundamentally, we believe in people and their ability to adapt and move forward. Throughout the history of computing, people have again and again adapted to new paradigms and interaction methods—even just when switching between different websites and apps and phones. We will help people get off on the right foot, and we have confidence that people will quickly find the new paradigms to be second-nature.

Looking forward

The Windows 8 user experience is forward-looking, yet respectful of the past. It reimagines what a PC is capable of, the scenarios for which it is

optimized, and how you interact with it. It enables tablets and laptops that are incredibly light and thin, with excellent battery life, which you can use with touch and keyboard and mouse in any combination you prefer. It is also the most capable, lean, and usable OS ever to power desktop PCs and gaming rigs.

The new Windows 8 user experience is no less than a bet on the future of computing, and stakes a claim to Windows' role in that future. We tried to break new ground in imagining how using a PC might become a fluid and enjoyable experience, how apps might work together to simplify the tasks you do every day, and how a single screen could bring together everything you love and care about into one always up-to-date place.

We believe in convergence—this has happened again and again in technology, and it will continue. We believe that you will want to carry around fewer, more capable devices. In addition to your phone, you want only one device that is equally at home on the couch and on the desk. You want a device that is light enough to hold for hours, but powerful enough to do real work with familiar and full-featured software—and which also allows a mouse or physical keyboard if you want. A device that is deeply personal, that natively understands the cloud, that roams your settings and content wherever you go.

Yes, there are parts of the Windows 8 UI that have generated discussions and even debate, and aspects of the change that will take some people a little time to understand and digest. Any change, particularly a change that doesn't just follow in the footsteps of what everyone else is doing, can be hard to fully grasp at first and will bring forward its share of both deep believers and naysayers.

The full picture of the Windows 8 experience will only emerge when new hardware from our partners becomes available, and when the Store opens up for all developers to start submitting their new apps. At the same time, there's no doubt that all the features of Windows 8 are compelling on today's hardware designed for Windows 7—with or without touch. Since we designed Windows 8 to work great for laptops and desktops, it will work naturally for your Windows 7 hardware. Think of past versions of Windows that worked on existing hardware but were even better with new hardware. That's our approach with Windows 8.

In 1993, when today's familiar Windows 95 user experience was first designed, PCs were beige, heavy, disconnected, and sitting under an office desk plugged in all the time. An average PC cost \$3450 in today's money!

Today, PCs are in the kitchen, in the living room, at the coffee shop, in your purse, on the train, in the passenger seat of your car. Increasingly they are mobile, always connected, affordable, and beautiful. And Windows PCs are in the workplace, no matter where that is or moves to. What would have seemed unrecognizable and "post-PC" 20 years ago is now the very definition of a PC.

The world changes and moves forward. Windows will continue to change too, as it has throughout its 27-year history.

Our vision for Windows 8 was to create a modern, fast and fluid user experience that defines the platform for the next decade of computing. One which upends the way conventional people think about tablets and laptops and the role of the devices they carry.

We wanted to create an experience that works however you want to work, powering a new class of PCs that you are proud to own and love having in your life.

Jensen Harris

Enhancing Windows 8 for multiple monitors

Steven Sinofsky | [2012-05-21T08:00:00+00:00](#)

*This post goes into the details around the multi-monitor experience for Windows 8. From the very first public release and demonstrations of Windows 8 we have shown improvements over Windows 7 for multi-monitor scenarios and have shown how we support new Metro style apps within a multi-monitor environment. We have continued to develop and refine features for multiple monitors and have significantly enhanced the experience as we move to our next milestone, the Release Preview. This post provides a bit of a preview of work that was not yet complete at the Consumer Preview, and serves as a reminder that the Developer Preview and Consumer Preview were works in progress. **Mark Yalovsky, a lead program manager on our User Experience team, authored this post.** (Note: This post is unchanged from last week when it was inadvertently posted as noted on [@buildwindows8](#).)*

--Steven

Connecting multiple monitors to a PC is one of the easiest ways to enhance your Windows experience. Plug in a second monitor and you instantly double your working surface. I've had a multi-monitor setup for the past 10 years; once you start using multiple monitors, you'll never want to go back to your old setup. A multi-monitor setup allows you to be more productive by having more windows across multiple screens. We're very excited about the ease at which tablets in Windows 8 will be able to support large screen and high resolution monitors (often through HDMI connectors), as this opens up a broad range of exciting new scenarios.

When we embarked on planning Windows 8, enhancing multi-monitor functionality was an important area to improve. A multiple monitor setup is certainly more common today than they used to be, and many technical professionals (developers, graphics professionals, architects, etc.) have started using it. Today, support for multiple monitors is standard on virtually all PC hardware, and monitor prices are at an all-time low (as of writing this post, you can purchase a 21" LED display in the \$140 USD range). As a result, we continue to see increased adoption of multi-monitor configurations, both by enthusiasts and technical professionals.

Data collected through the [Windows Feedback Program](#) indicates that approximately 14% of desktop PCs and approximately 5% of laptop PCs have run with multiple monitors. It is important to note that this particular opt-in data set is enthusiast-leaning so represents the high end of usage (relative to previously shared measures that look at the entire universe of PCs), but we thought we would share this data set to reinforce another data source.

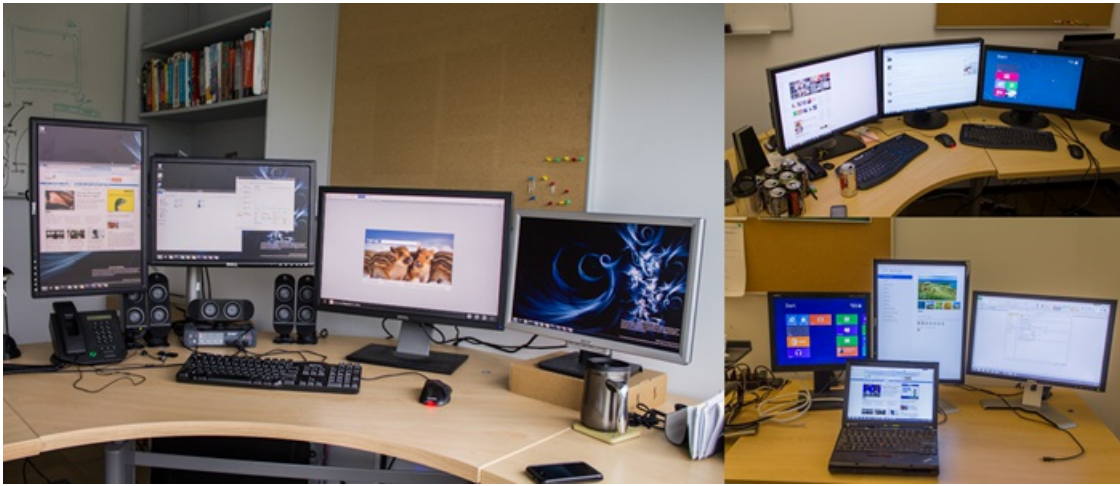
Number of Desktop PC Laptop PC monitors

1	85.32%	95.64%
2	13.48%	4.36%
3	0.85%	0.00%
4	0.34%	0.00%

We recognize that a key value of using multiple monitors lies in the desire to increase multitasking. This is especially true of those of you who spend time arranging your desktop windows to maximize the available real estate across multiple displays. Speaking firsthand, most developers and testers at Microsoft have a multi-monitor setup in their offices, walking through the hallways one sees a wide range of monitor configurations from 2 to 4 or more monitors among the engineering team. This affords two important scenarios. First, developers can use a tool like Visual Studio on one screen and have the running/debugged program on another, or they can add an additional monitor and reserve it for side tasks such as email or web browsing.

With that in mind, we set out to achieve the following goals for those using multiple monitors with Windows 8:

- **Make the desktop a more personal experience.** Perhaps the most personalized feature on the desktop is the ability to customize the desktop background. We set out to make this a great experience on multiple monitors too.
- **Improve the efficiency of accessing apps across monitors.** In Windows 7, the top request from people using multiple monitors was to improve the taskbar efficiency.
- **Improve the efficiency of accessing system UI.** In Windows 7, you could only access the Start menu on one monitor. With the introduction in Windows 8 of new UI that puts controls at the edges of the screen, we wanted to make sure that it's still easy to access Start, the charms, the clock, and your recently used apps from every monitor.
- **Allow side-by-side Metro style and desktop apps.** You can launch or move a Metro style app to any monitor, side-by-side with desktop apps on another screen.



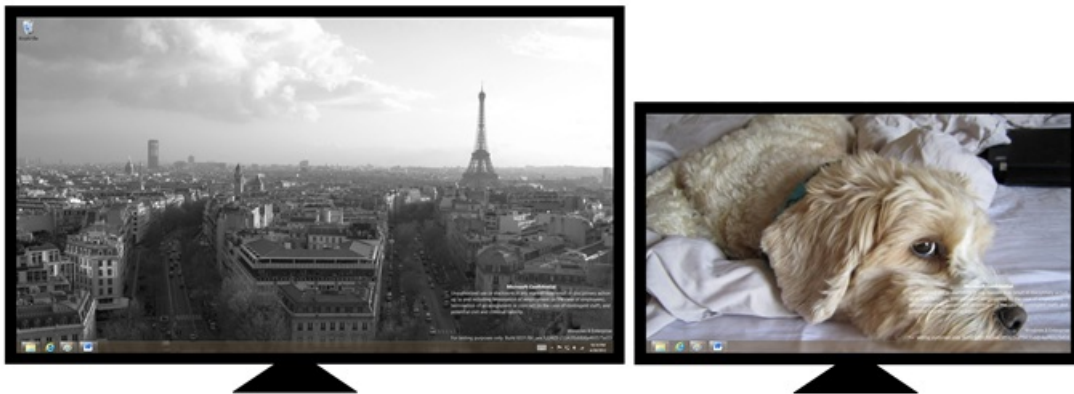
Examples of multi-monitor configurations in Microsoft offices

Multi-monitor desktop background personalization

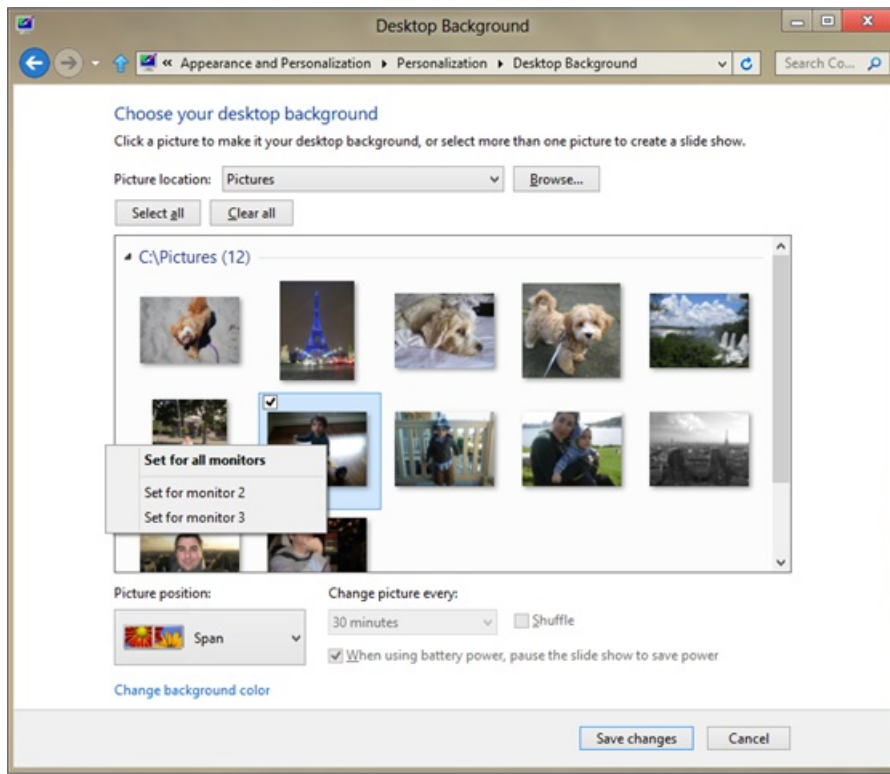
Customizing the desktop background is a very popular feature in Windows 7. In fact, telemetry shows that more than 75% of users customized the desktop background. A limitation in Windows 7 is that in a multi-monitor configuration, you can only select a single background image that is duplicated across your monitors. Not only is this limited from a customization perspective (how many people really want to look at the same picture twice?), but it also looks bad if your monitors have significantly different resolution or are different orientations (portrait vs. landscape).

We know that some of you use some pretty advanced third-party tools for sophisticated background image management. In Windows 8, we made the background customization feature customizable on each monitor you use, and for mainstream customers, we've provided solutions to the common desktop personalization problems encountered with Windows 7:

- **Show a different desktop background on each monitor.** When selecting a personalization theme, Windows 8 automatically puts a different desktop background on each monitor. You can even set a slide show to cycle through pictures across all monitors, or pick specific background pictures for each monitor.

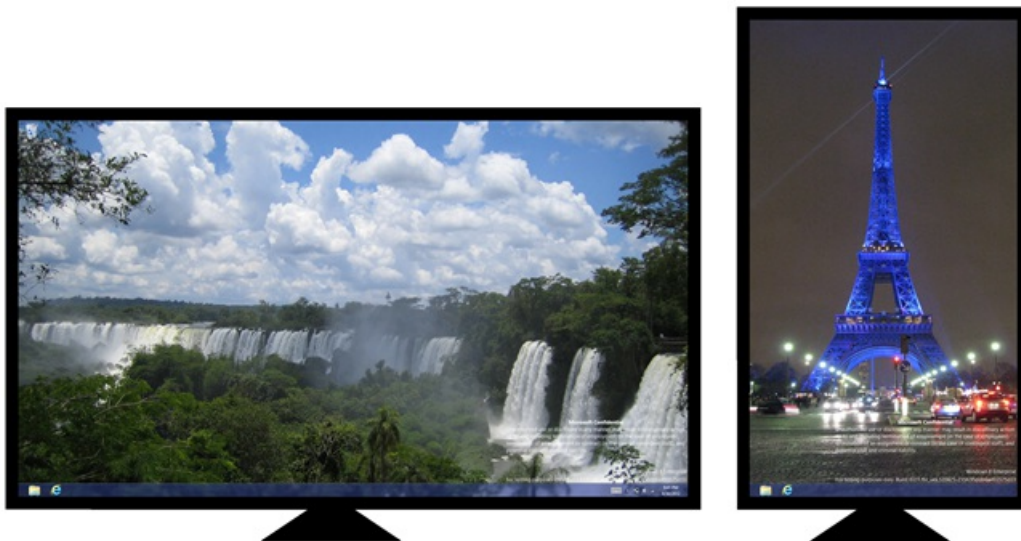


Different backgrounds on each monitor



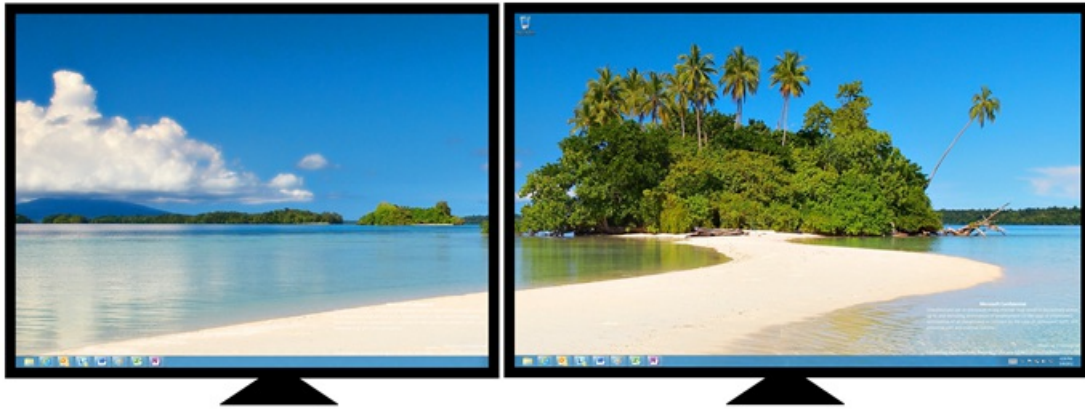
Option to pick different backgrounds on each monitor

- **Multi-monitor slide show.** It is very typical for people to have a multi-monitor setup that consists of different sized and/or oriented monitors. And of course, not all photos look great in both portrait and landscape or on all screen sizes and resolutions. To address this, we've added logic to the slide show code that selects the best suited images for each monitor.

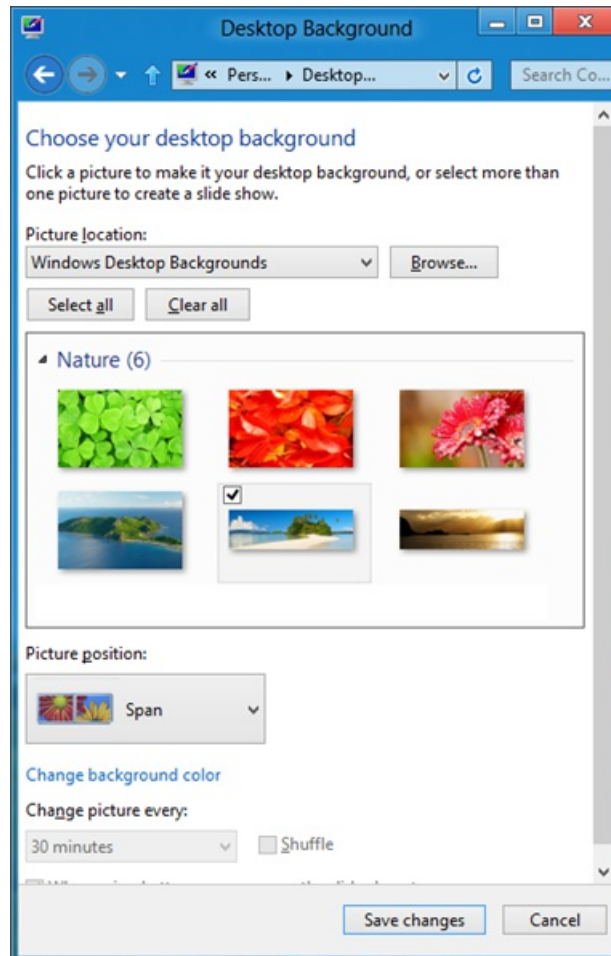


Slideshow with image selection that matches monitor orientation

- **Span desktop background across all monitors.** You can now span a single panoramic picture across multiple monitors. We are also including a new panoramic theme in the personalization options for Windows 8.



Span an image across all monitors



Option to span image across all monitors, including panoramic pictures

Multi-monitor taskbar

Of course the main reason most people use multi-monitor configurations is to be more productive. With the extra screen real estate you are able to see more windows up at the same time. The flip side to having more windows visible is that window management can become more challenging. In the desktop, the taskbar is the primary place for managing windows. As some of you pointed out to us in our Windows 7 blogs, lack of multi-monitor support for the taskbar is a gap. This can be summed up by one comment from the [e7 blog](#):

@AlexJerebtov, “The lack of multi-monitor [Taskbar] support is just about a crime”.

What’s interesting about adding multi-monitor support to the taskbar is that even among a relatively small group of users, there are several opinions as to what the “right” design should be. As you can imagine, this is quite common in designing a new version of Windows—there are many points of view on how even relatively small things should be implemented. These are some observations from a variety of hands-on research methods:

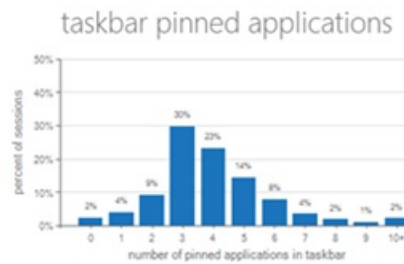
- **People tend to approach window management in either an organized or an *ad-hoc* fashion.** People who manage windows in an *ad-hoc* fashion frequently move windows between monitors as their workflow requires, and do not keep track of what monitor a window is on. People that manage windows in a more organized fashion tend to designate specific monitors for specific apps and tasks (for example, email always on the left, the browser always on the right). There is not always a hard line between these two working styles and most people

move windows in an ad-hoc fashion from time to time.

- **Improved efficiency was consistently cited as a goal for the taskbar.** Nearly all users conveyed the desire for improved taskbar efficiency. When we observed people using multiple monitors in their work, we noticed that the simple act of switching windows would sometimes require them to turn their heads, swivel in their seats, and reposition their mouse cursor as they jumped from a secondary monitor to the main taskbar monitor and all the way back again. Of course we also heard this articulated in term of mouse-efficiency. That is, we want to reduce the distance that you need to move the mouse to find and switch to a window on the taskbar.
- **It is common for people to have a primary monitor.** Many people have one monitor that they run most of their apps on, with a smaller secondary monitor that has a few windows open for peripheral tasks (for example, managing a playlist, sending IMs, playing a video). This is particularly true for users who have kept their old monitor on-hand after upgrading to a newer, bigger, higher-resolution monitor. Ad hoc users still move windows freely between monitors, but tend to prefer one over the other for the tasks that they are currently focusing on, partly because it is comfortable to set up a chair, keyboard, and mouse to face one monitor directly.
- **Taskbar real estate is generally not a problem.** When we designed the taskbar we were fairly confident that most people would find the default setting sufficient even with customization easy to find. Hands-on research confirms the majority of users keep the default setting where windows are grouped by app on the taskbar. Telemetry that looked at hundreds of millions of sessions further confirmed that only 6% of users ungroup taskbar buttons.

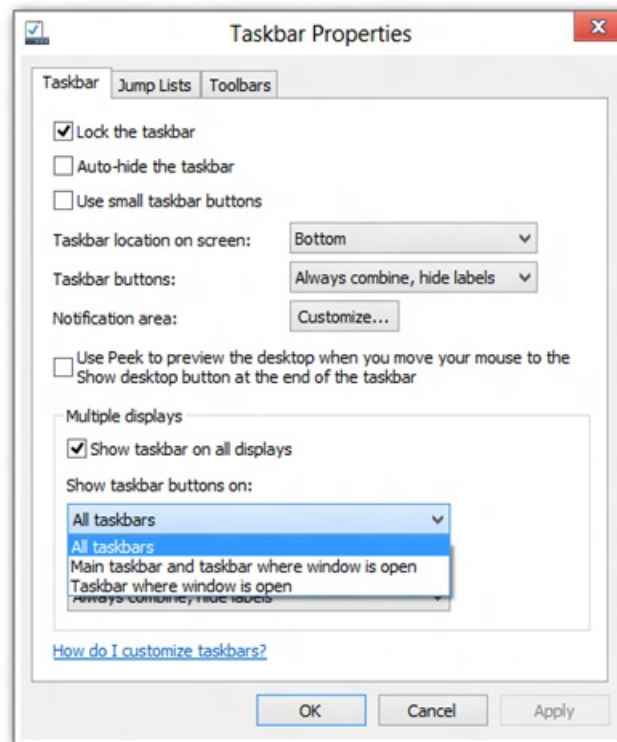
83% of users have the default taskbar appearance settings

6% of users ungroup taskbar buttons



Multi-monitor taskbar options

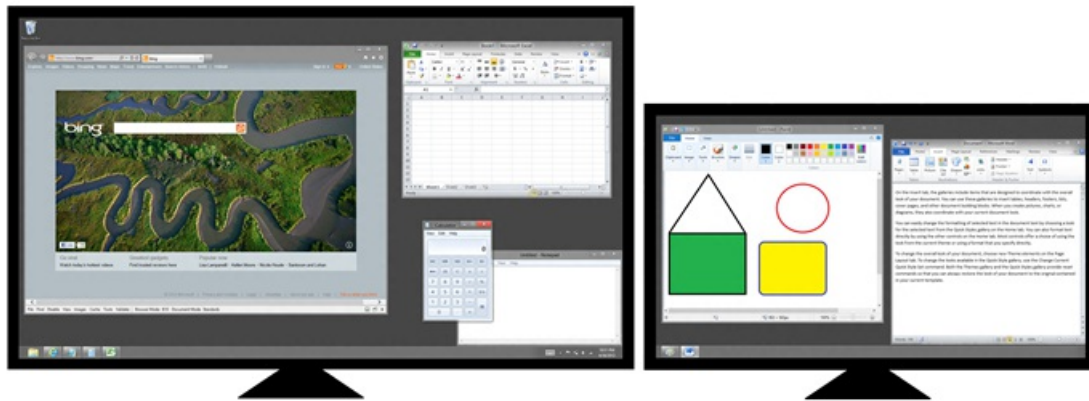
Based on our field and lab observations we understood that people employ different window management techniques (always ad-hoc, always organized, mixed). For this reason, we chose to provide several multi-monitor taskbar options, so that advanced users with multiple monitors can still fine-tune their desktop experience.



Windows 8 taskbar properties

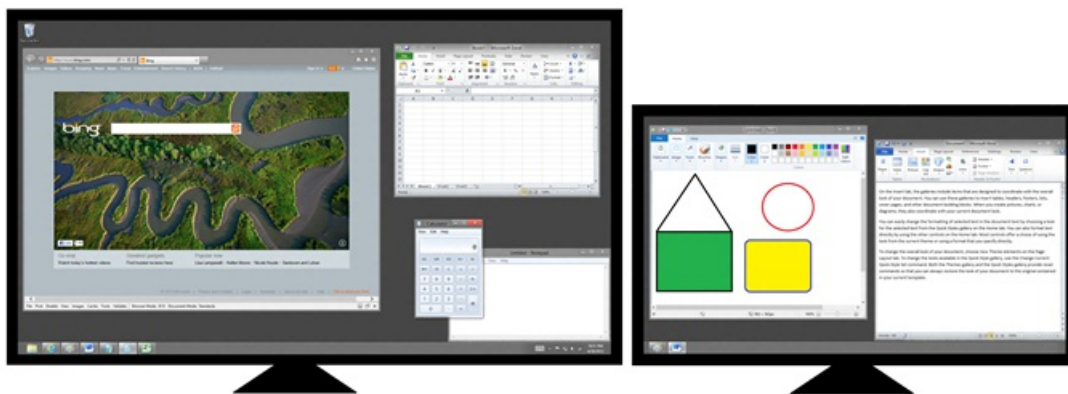
- **Show taskbar buttons on the taskbar where the window is open.** This is the most obvious option that comes to mind when thinking of a

multi-monitor taskbar. In this configuration, each monitor's taskbar contains icons for only the windows that are on that monitor. The advantage of this option is that it is simple and predictable. This tested well with people who were very organized in their placement of windows, or who had dedicated monitors for specific tasks. On the other hand, ad-hoc users found this design to be inefficient, as they needed to remember what monitor a particular window was on.



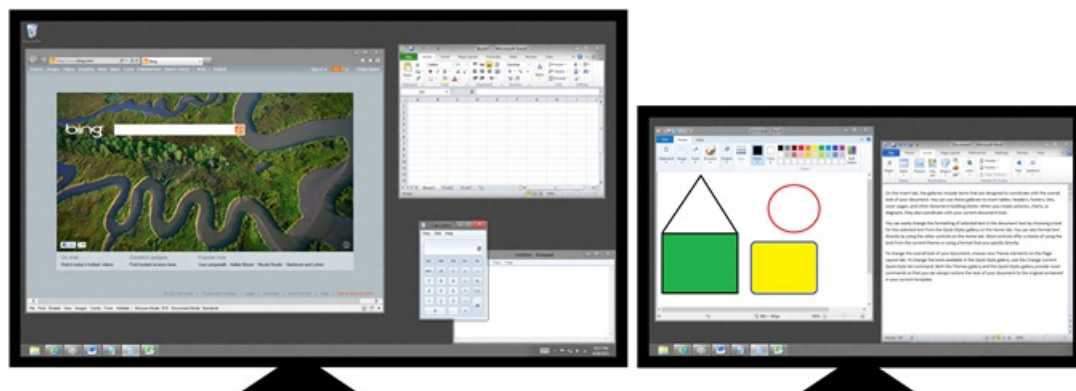
App buttons on the taskbar where the window is open

- **Show taskbar buttons on main taskbar and taskbar where window is open.** In this configuration, the main monitor has a special taskbar that contains all the windows across all monitors. All the other monitors have unique taskbars, as with the first option described above. This option offers some of the cleanliness of the *taskbar where the window is open* model, but also offers a consistent and efficient way to get to any window via the master taskbar. People who think in terms of a primary monitor will probably prefer this option.



App buttons on main taskbar and where window is open

- **Show taskbar buttons on all taskbars (default).** In this configuration, all windows are available on all taskbars. This configuration is designed for maximum mouse efficiency because you can always activate any window from any monitor. Of all the options, this works the best for ad-hoc windows management, as there is no need to keep track of where windows are located. While some users indicated a preference for one of the other options, this was the only option that was efficient for the vast majority of users, which is why this is the default setting.



App buttons on all taskbars (default option)

For those of you who have used the Consumer Preview on multiple monitors, you'll notice that Start, the charms, and the clock are only shown on a single monitor. The feedback has been vocal and clear on this and of course, given the prevalence of multi-monitor setups even in our own hallways, we understood that this feature simply wasn't complete. Looking forward, here's a sneak peak at some of the improvements we're making to multi-monitor usage for the Release Preview.

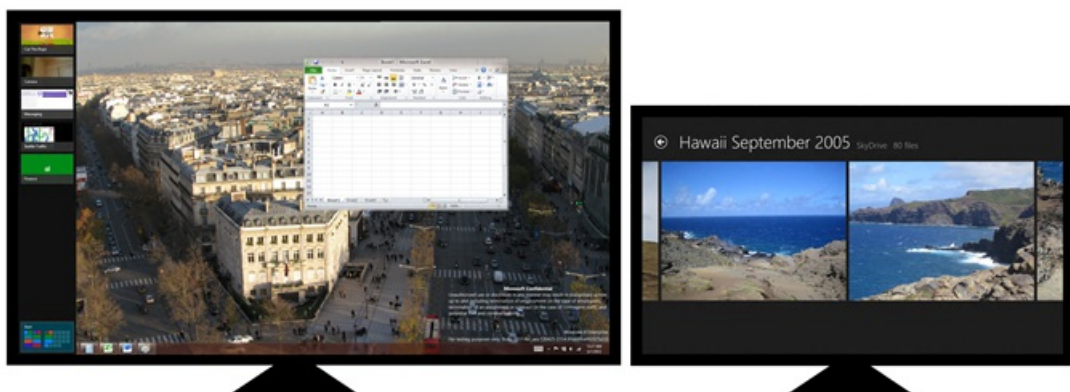
No broken corners and edges

On the Consumer Preview in a multi-monitor setup, it is difficult to find the Start screen and other UI that is invoked from the corners with a mouse, since those activation areas are only available on a single monitor. In the upcoming Release Preview, we are making all the corners and edges alive on all monitors. You can now bring up Start, the charms, and app switching from the corners of any monitor. Want Start on monitor 1? Just go to the bottom-left corner on that monitor. Want it on monitor 2? Go to the bottom-left corner on monitor 2. This not only improves discoverability, it also improves mouse efficiency and multitasking. To launch or move an app to a specific monitor, bring up Start on that monitor and launch the app, or switch to the app using the app switcher at the left edge.

You can launch Start on any monitor:

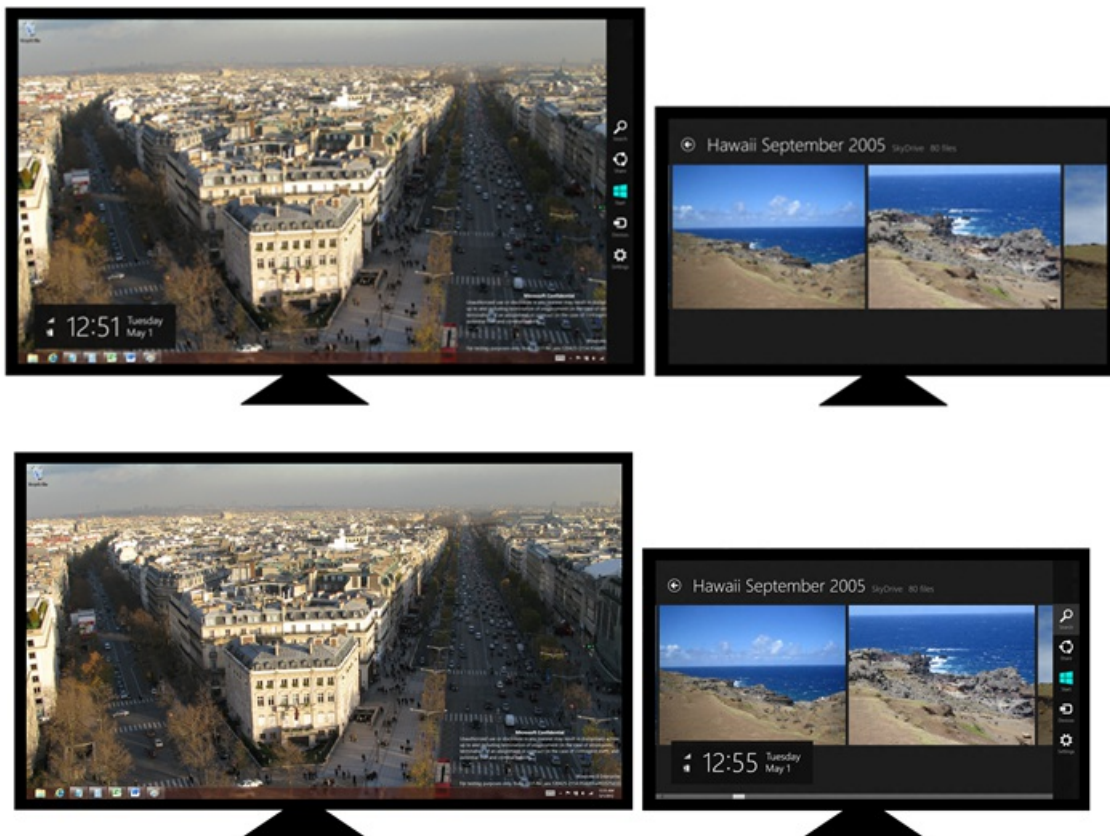


You can switch back to recently used apps from any monitor:





And you can bring up the charms on any monitor:



Launch and move Metro style apps to any monitor

There are several ways that you can launch and move an app:

- **Start.** You can bring up Start on any monitor by moving your mouse to the bottom-left corner, or via the Start charm that you can invoke from the top and bottom-right corners of any monitor. Pressing the Windows key launches Start on the last monitor where Start or a Metro style app appeared.
- **Switch back to an app from any monitor.** You can switch back to an app on any monitor by moving your mouse to the top-left corner. Clicking the app thumbnail switches you back to the app on that monitor.
- **Keyboard shortcuts.** We are introducing new keyboard shortcuts that build on the shortcuts from Windows 7. Win+Pg Up or Win+Pg Dn moves Metro style apps across monitors. Win+Arrow and Win+Shift+Arrow continue to work on desktop apps as they did in Windows 7, by snapping and moving desktop windows across monitors.
- **Drag and drop.** Using the mouse, you can now drag and drop Metro Style apps across monitors. Drag and drop works for both full screen and snapped apps.

Improved mouse targeting on the shared edge

A multi-monitor setup brings the major benefit of more real estate, but it also lacks the [Fitts' Law benefits](#) of hard edges and corners across displays. While it's extremely easy to trigger corner UI such as Start, charms, or recently used apps on a single monitor, it isn't uncommon to overshoot the mouse when the corner appears on a shared edge on a multi-monitor configuration.

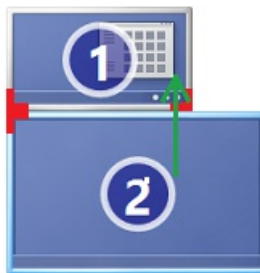
With multiple monitors in fact, targeting the shared edge can be downright difficult. Move a few pixels too far and your cursor is suddenly on the wrong monitor. This has been a common challenge in previous versions of Windows as well, like when you're trying to hit the close button or scroll bars on a maximized window on a shared edge. Many work around this by remembering to move the mouse slowly as it approaches a shared edge or by avoiding window layouts that bump up against those edges. We commonly observe this behavior in our own usage and in field studies.

In the Release Preview, we're introducing an improved model for shared edges that makes it easier to target UI along a shared edge.

Since corners are even more important for Windows 8, we've created real corners along the shared edges to mimic the Fitts' Law advantages of a single monitor. The red corners in the diagram below demonstrate how these corners can help guide your mouse.



We've designed the corners to provide help when you need it and to get out of the way when you don't. The protruding corner target is 6 pixels in height, which means that it is only noticeable when you're trying to target the corner of the screen. Also, we've designed the corner to only work for the monitor your cursor is on. For example, leaving monitor 2 for monitor 1 in the diagram below, the bottom corner in monitor 1 will not interfere as you move your mouse across the shared edge.



Shared corner does not block cross monitor navigation

The shared corner isn't just an improvement for the new Windows 8 UI, but it also makes it easier to target controls on the desktop like *Close* and *Show desktop*. As a result, targeting shared corners is fast and fluid. First-hand experience is a must with this design, as you will notice this improvement right away when using the new Release Preview.

More to come

We have lots of ideas for how we could do even more with Metro style apps on multiple monitors. Our goal for Windows 8 is to deliver a great Metro style app experience alongside desktop apps, improving multitasking efficiency and making it easy to access the controls you need along the edges of every screen. We wanted to make sure your desktop experience was even more efficient, with new functionality such as the spanning taskbar, and we wanted you to also have access to Metro style apps while you're also using the desktop. As we see new apps developed, and as we see how developers might want to take advantage of multi-monitor configurations in new ways with immersive and full screen apps, we will of course enhance this experience (and APIs) even further.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

We hope that you enjoy these new multi-monitor features. Thank you for all of your feedback – it has certainly helped us to improve Windows 8 as we moved from Developer Preview, to Consumer Preview, and soon, to the Release Preview.

--Mark

Designing for PCs that boot faster than ever before

Steven Sinofsky | [2012-05-22T08:30:00+00:00](#)

While we're hard at work making sure you never have to turn off your PC and can run in a connected standby state, we know that there will still be reboots for updating key system components. We've previously talked about [reengineering the Windows boot experience](#) and how we modernized and touch-enabled the core boot loader and choices. We've also made [boot go by very fast](#). In fact, it is now so fast that we had to look at the design to enable the kinds of diagnostic boots required by those who do want to dig into their BIOS or load in alternative ways.

*In this post, **Chris Clark**, a program manager on our User Experience team, talks about the design of an incredibly fast boot experience.*

--Steven

Windows 8 has a problem – it really can boot up too quickly.

So quickly, in fact, that there is no longer time for *anything* to interrupt boot. When you turn on a Windows 8 PC, there's no longer long enough to detect keystrokes like F2 or F8, much less time to read a message such as "Press F2 for Setup." For the first time in decades, you will no longer be able to interrupt boot and tell your PC to do anything different than what it was already expecting to do.

Fast booting is something we definitely want to preserve. Certainly no one would imagine intentionally slowing down boot to allow these functions to work as they did in the past. In this blog I'll walk through how we're addressing this "problem" with new solutions that will keep your PC booting as quickly as possible, while still letting you do all the things you expect.

Too fast to interrupt

It's worth taking a moment to watch (again, if you've already seen it) [the fast boot video](#) posted by Gabe Aul in his previous post about [delivering fast boot times in Windows 8](#). In this video you can see a laptop with a solid state drive (SSD) fully booting in less than 7 seconds. Booting this fast doesn't require special hardware, but it is a feature of new PCs. You'll still see much improved boot times in existing hardware, but in many PCs, the BIOS itself (the BIOS logo and set of messages you see as you boot up) does take significant time. An SSD contributes to the fast boot time as well, as you can imagine.

If the entire length of boot passes in just seven seconds, the individual portions that comprise the boot sequence go by almost too quickly to notice (much less, interrupt). Most of the decisions about what will happen in boot are over in the first 2-3 seconds – after that, booting is just about getting to Windows as quickly as possible. These 2-3 seconds include the time allowed for firmware initialization and POST (< 2 seconds), and the time allowed for the Windows boot manager to detect an alternate boot path (< 200 milliseconds on some systems). These times will continue to shrink, and even now they no longer allow enough time to interrupt boot as you could in the past.

On the Windows team, we felt the impact of this change first, and perhaps most painfully, with our own F8 behavior. In previous versions of Windows (as far back as Windows 95), you could press F8 at the beginning of boot to access an advanced boot options menu. This is where you'd find useful options such as Safe Mode and "Disable driver signing." I personally remember using them when I upgraded my first PC from Windows 3.1 to Windows 95. F8 helped me quickly resolve an upgrade issue and get started using Windows 95.

However, the hardware and software improvements in Windows 8 have collapsed the slice of time that remains for Windows to read and respond to the F8 keystroke. We have SSD-based UEFI systems where the "F8 window" is always less than 200 milliseconds. No matter how fast your fingers are, there is no way to reliably catch a 200 millisecond event. So you tap. I remember walking the halls and hearing people frantically trying to catch the F8 window – "tap-tap-tap-tap-tap-tap" – only to watch them reboot several times until they managed to finally get a tap inside the F8 window. We did an informal study and determined that top performers could, at best, sustain repeated tapping at about a 250ms frequency. Even in this best case, catching a 200 millisecond window still depends somewhat on randomness. And even if you eventually manage to catch this short window of time, you still have to contend with sore fingers, wasted time, and just how ridiculous people look when they are frantically jamming on their keyboard.

The problem we saw with our F8 key extends to any other key you may want to press during boot. For example, in the Windows 8 Developer Preview release, the F8 key led to a full set of repair, recovery, and advanced boot options. A different key allowed developer-focused options, such as enabling debugging or disabling driver signing. And on most PCs, there are additional keystrokes used by the firmware and advertised by messages during POST: "Press F2 for Setup" or "Press F12 for Network Boot." Now, POST is almost over by the time these instructions could be displayed. And in many cases, the keyboard wouldn't be functional until so late in POST that it's almost not worth the time it would take the firmware to look for these keystrokes. Some devices won't even try.

Even so, every one of these keystrokes plays an important role, and we have historically counted on them to provide important interrupt functions in boot. However, now, there is no longer time to do any of them.

Defining the problem space

We looked at these problems from many angles, and took a holistic approach to solving them. This effort spanned across developers, testers, and

program managers, examining everything from the deepest parts of the kernel to the overall user experience. Approaching this first as an engineering problem, we identified the situations and scenarios that depended on keystrokes in boot and considered literally dozens of ways to restore functionality to each scenario in Windows 8.

Here are some of the key scenarios pulled from this list:

- Even when Windows is booting up correctly, you may want to do something different – for example, you may want to boot from an alternate device such as a USB drive, go to the firmware’s BIOS setup options, or run tools from within the protected Windows Recovery Environment image on a separate partition. In general, these scenarios were accomplished in the past mainly without the involvement of Windows, using firmware-specific keys such as F2 or F12 (or some other key that you couldn’t quite remember!).
- You may need to troubleshoot a problem after something goes wrong, or want to undo something that just happened. Windows has many tools that assist with situations like these, such as allowing you to refresh or reset your PC, go back to a restore point using System Restore, or perform manual troubleshooting via the always-popular Command Prompt. In the past, these troubleshooting options were accessed primarily via the Windows boot manager, by pressing F8 at the beginning of boot.
- Some error cases in startup are difficult to automatically detect. For example, the Windows boot process may have succeeded, but errors in components that are loaded later actually make Windows unusable. These cases are rare, but an example of where this might happen is a corrupt driver installation causing the login screen to crash whenever it loads. On previous-era hardware, you could interrupt boot with a keystroke (F8, for example) and reach a suitable repair option before the crashing component was even loaded. Over time, it has gotten harder to interrupt boot in this way, and in Windows 8, it’s virtually impossible.
- We needed to enable certain startup options that are mainly used by developers – both inside and outside of Windows. Previously you could access these by pressing a key like F8 at the beginning of boot. These developer-targeted options are still important and include disabling driver signature enforcement, turning off “early launch anti-malware,” as well as other options.

One key design principle we focused on was how our solutions would fit in with the rest of Windows 8. We believed that these various boot options were more alike than they were different, and shouldn’t be located in different places within Windows. To look at this from the opposite direction, no one should need to learn how Windows is built, under the hood, to know where to go for a certain task. In the purest sense, we wanted it to “just work.”

Three solutions – one experience

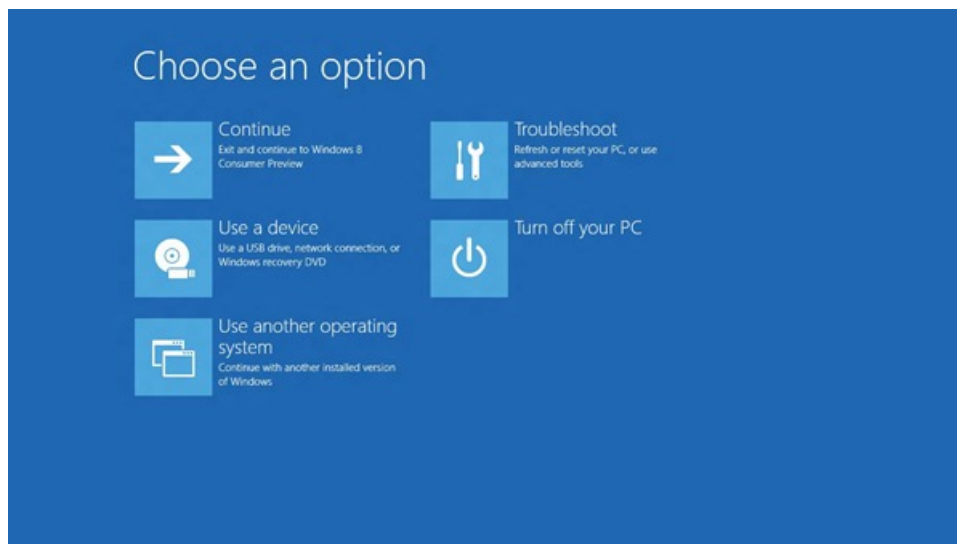
We ultimately solved these problems with a combination of three different solutions. Together they create a unified experience and solve the scenarios without needing to interrupt boot with a keystroke:

1. We pulled together all the options into a single menu – the boot options menu – that has all the troubleshooting tools, the developer-focused options for Windows startup, methods for accessing the firmware’s BIOS setup, and a straightforward method for booting to alternate devices such as USB drives.
2. We created failover behaviors that automatically bring up the boot options menu (in a highly robust and validated environment) whenever there is a problem that would keep the PC from booting successfully into Windows.
3. Finally, we created several straightforward methods to easily reach the boot options menu, even when nothing is wrong with Windows or boot. Instead of these menus and options being “interrupt-driven,” they are triggered in an intentional way that is much easier to accomplish successfully.

Each of these solutions addresses a different aspect of the core problem, and together they create a single, cohesive end-to-end experience.

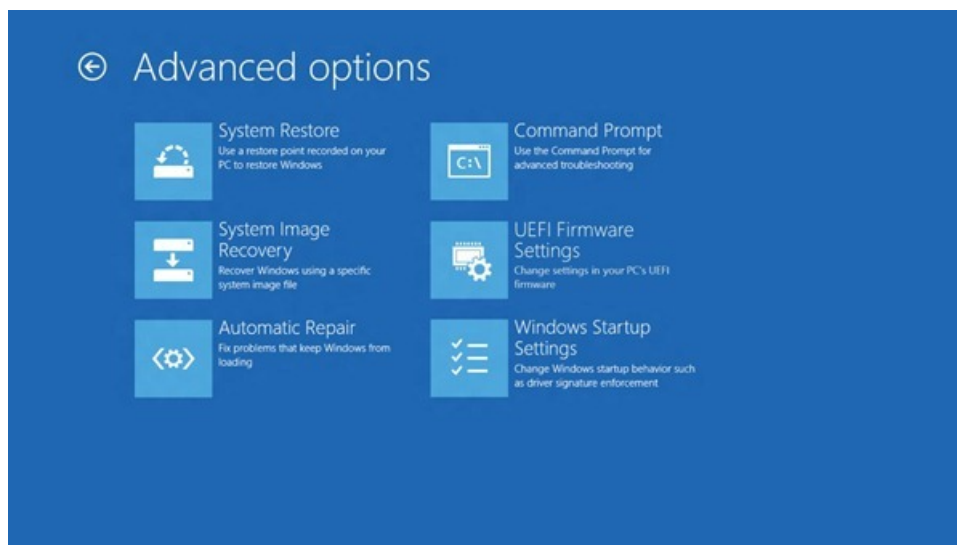
A single menu for every boot option

The core vision behind the boot options menu is to create a single place for every option that affects the startup behavior of the Windows 8 PC. Portions of this menu were discussed in detail in our previous blog post titled [Reengineering the Windows boot experience](#). That post has the complete details and describes the fundamental changes made within the boot menus to enable touch interaction, Windows 8 visuals, and a cohesive user experience across the many surfaces that make up boot. Here is a screenshot of the boot options menu on one of my UEFI-based PCs:



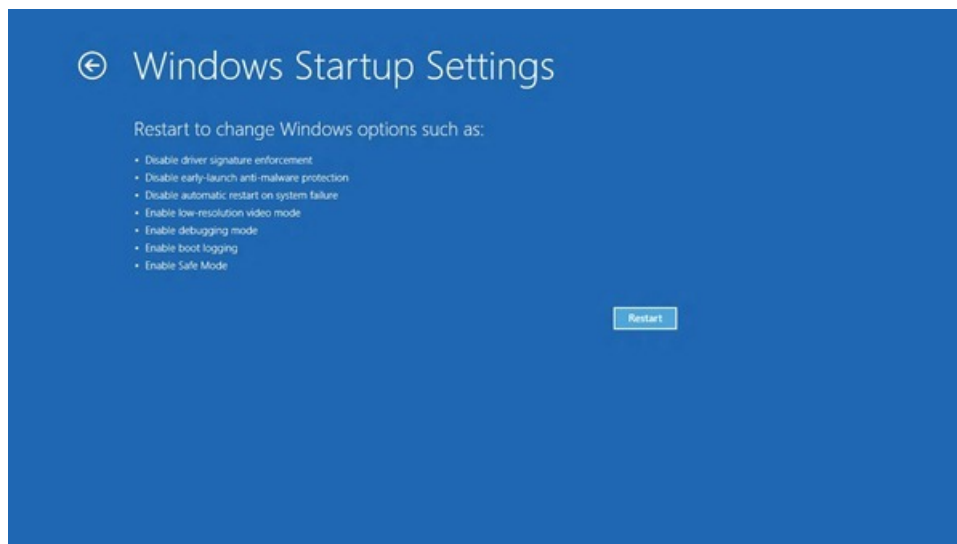
Booting to an alternate device (such as a USB drive or network) is one of the most common scenarios that previously required interrupting boot with a keystroke. With Windows 8 UEFI-based firmware, we can now use software to trigger this. On these devices, you'll now see the "Use a device" button in the boot options menu, which provides this functionality directly. As you can see in the above image, this functionality sits side-by-side with the other boot options. Windows no longer requires a keystroke interruption to boot from an alternate device, (assuming, for the moment, that you can reach the boot options menu itself without requiring a keystroke in boot. More on this in a minute.)

Into this same menu, we've added new functionality that allows you to reboot directly into the UEFI firmware's BIOS setup (on Windows 8 UEFI hardware that supports this). On previous-era hardware, instructions for entering BIOS setup appeared at POST in messages like "Press F2 for setup." (These messages have been around on PCs longer than perhaps any other type of UI.) They will still occur on systems that were made prior to Windows 8, where they will continue to work (primarily because these devices take several seconds to POST.) However, a Windows 8 UEFI-based PC won't stay in POST long enough for keystrokes like this to be used, so the new UEFI-based functionality allows this option to live on in the boot options menu. After looking at the other items in this menu, we decided to place the button that reboots the PC into the UEFI firmware's BIOS setup under the "Troubleshooting" node, within the "Advanced options" group:



A quick note about older, non-UEFI devices: legacy hardware that was made before Windows 8 will not have these new UEFI-provided menu features (booting to firmware settings and booting directly to a device). The firmware on these devices will continue to support this functionality from the POST screen as it did in the past (using messages such as "Press F2 for Setup"). There is still time for keystrokes like this to work in POST on these legacy devices, since they won't have the improvements that enable a Windows 8 PC to POST in less than 2 seconds.

The next item appears on all Windows 8 devices – UEFI and non-UEFI alike. In the image above, you can see that we've added **Windows Startup Settings**. This new addition brings the entry point for the developer-focused Windows startup options into the unified boot options menu, and allows us to satisfy the scenarios that previously required the separate key during boot. These include items such as "disable driver signing" and "debugging mode," as well as Safe Mode and several other options. Here is a close-up view of the informational page for these options:



The boot options menu creates a single place for every option that affects the startup behavior of the Windows 8 PC. By bringing these together into a single place, the boot options menu has become a familiar, unified, and highly usable place for these related items. Tasks such as changing Windows Startup settings, entering the UEFI firmware's BIOS setup, or booting to a USB drive no longer require interrupting boot with a keystroke – assuming you can get to the boot options menu itself. So let's look at how you get there.

Getting to the boot options menu (automatically) when there is a problem

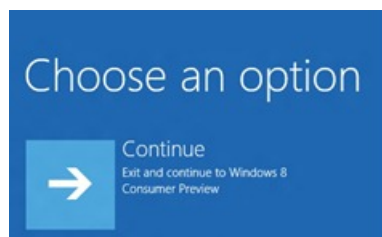
There are two main situations where you'll need to get to the boot options menu on a Windows 8 PC. The first case is when something has gone wrong and a repair action is necessary to restore the PC to full functionality. The second case (which I'll cover in the next section) is when nothing is wrong, but you want to change some aspect of startup behavior or firmware configuration, or boot from a different device than usual.

In the first case, something has gone wrong and repairs are needed. The previous model of PC hardware required you (or someone you trust) to begin this troubleshooting process by pressing one of the several possible keystrokes during boot. For example, the options in the Windows Developer Preview release were split between Shift+F8, F8, and firmware-dependent keys such as F2 or F12, (which often varied across different PCs).

Each of these keystrokes represents the first step in troubleshooting that will lead to eventual repair. Unifying all of these in a single boot options menu removes the need to use multiple keys for the many available options. And to take this even further, we've removed even this one remaining keystroke by automatically loading the boot options menu when there is no way to successfully complete Windows startup.

In Windows 8, this automatic failover behavior will take you directly to the boot options menu whenever there is a problem that would otherwise keep your PC from loading Windows. This even includes cases where it appears (to Windows) that boot has succeeded, but in actuality the PC is unusable. An example of how this could occur would be a faulty driver installation that is causing the main logon screen to appear completely blank. Windows may not be aware that the screen is blank, but anyone looking at the screen knows this immediately. We now algorithmically detect when this has occurred across multiple boots, and automatically boot directly into the boot options menu inside the Windows Recovery Environment (WinRE). Since the source image for WinRE contains drivers and files that are kept separate from the main Windows installation, it's not affected by any software changes and is a reliable environment to begin troubleshooting from the boot options menu.

Could this behavior ever result in Windows going to the boot options menu in Windows RE when nothing is actually wrong? Requiring two consecutive occurrences certainly reduces this chance, but it's definitely possible. With this in mind, we designed the boot options menu to have a prominent **Continue** button in the first position, as a clear escape path for anyone not actually experiencing problems with their Windows 8 PC. We studied this in our usability lab to see what people would do when this boot options menu appeared unexpectedly. We were happy to find that the **Continue** button served its purpose and provided an important escape hatch against false positives.



In certain situations, Windows 8 can be even more specific about taking appropriate action to a specific problem. For example, if the core boot sequence itself fails to complete, we automatically try a second time. If this also doesn't succeed, then Windows RE is automatically loaded and launches the specialized Startup Repair Tool. Even though this tool is tailor-made to fix many errors in the boot process, we still provide a pathway to all the other troubleshooting tools within the boot options menu for cases when the Startup Repair Tool is unsuccessful.

These automatic detection behaviors ensure the repair and recovery tools within Windows are always available, even when Windows itself is

unable to load properly. Without needing to press a key or take any action, Windows RE is automatically loaded when it's needed, allowing repair and recovery using the troubleshooting tools from the boot options menu itself.

Getting to the boot options menu whenever you want (even when nothing is wrong)

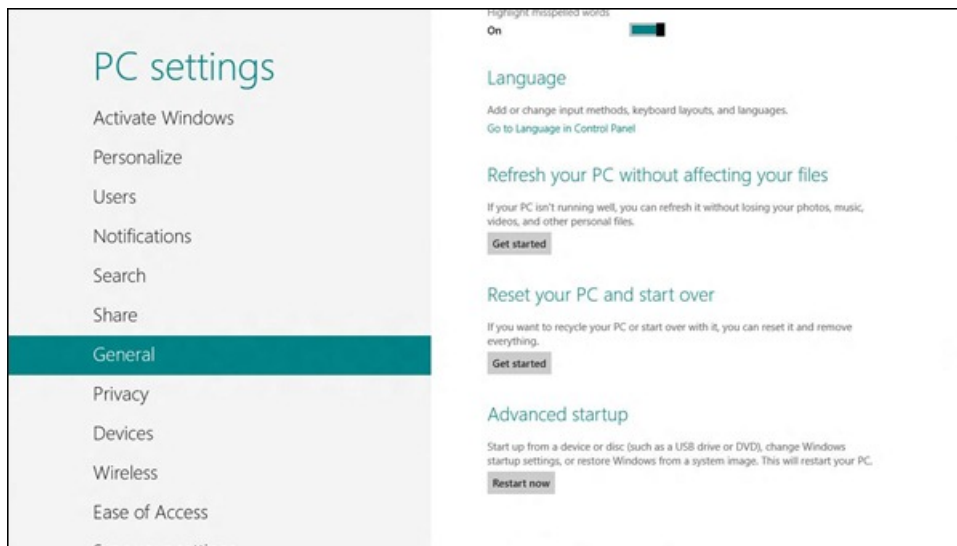
Even in non-error situations, we wanted an easy pathway to the boot options from within Windows. Many of the items in the menu are necessary even when everything is fully functional: booting to an alternate device, changing firmware configuration, and changing the developer-focused Windows Startup Settings, for example.

We wanted to make it easy to get to the boot options menu whenever you needed it, in a way that would logically fit within a fully-functional Windows 8.

In general, our preference is to create one method to do a certain thing, and make this one method the best possible. Even when there are multiple ways to do something, there is always a primary method – usually the most commonly used one, which covers the majority of cases. By choosing one way to do a certain thing, this way can be designed for a specific set of usage scenarios, and we can reasonably expect it to remain useful, usable, and desirable across these scenarios. Sometimes there are other cases that are not covered by the primary method. If these cases are not compelling enough to address, the primary method may truly be the only way.

However, in our case, we built a primary method and then added two more pathways: one to ensure we covered all the necessary scenarios, and a second to maintain a consistent pattern with existing Windows components.

The primary method of reaching the boot options is from **Advanced startup** on the **General** tab of **PC settings**. You can get to **PC settings** from the Settings charm, or by searching from the Start screen using specific search terms, such as boot, startup, safe mode, firmware, BIOS, or several others. On the General tab, you'll see a short description of the options that will be available in the boot options menu, as well as a **Restart now** button. The descriptions shown on this screen are fully dynamic, and will change based on the hardware, firmware, and software available on your specific Windows 8 PC.



Pressing the **Restart now** button under **Advanced startup** begins the primary pathway to reach the boot options on a fully functional system. The system begins the normal restart process. Then, just before Windows has finished shutting down and is about to fully restart and enter POST, the entire process is paused and the boot options menu fades into view. This is the latest point that UI can even appear during the shutdown/restart sequence. We decided to pause the restart process at this middle point, so that you can choose your destination before the PC goes through another POST. By choosing the desired boot option before POST occurs, we can jump directly to the firmware setup or device-boot (when these are chosen) without needing to go through a second restart and a second POST. You can even use this menu to quickly boot into a second Windows installation if you want to. Since Windows pauses the restart sequence to show the boot options menu, this is one of the fastest ways to boot to a second OS.

For even quicker access, there's another way of reaching the boot options menu: from within the shutdown menu. If you hold down the Shift key while clicking **Restart**, Windows 8 will go through the same sequence of events as if you had clicked **Advanced startup** from within **PC settings**. Since you can open the shutdown menu from any part of Windows 8 using the Settings charm, this is an especially quick way to directly reach the boot options menu. As you watch the video at the end of this post, you will notice that we've moved this command so there is a straight linear flow with your mouse to reach these options — a flow that is less demanding than in Windows 7.

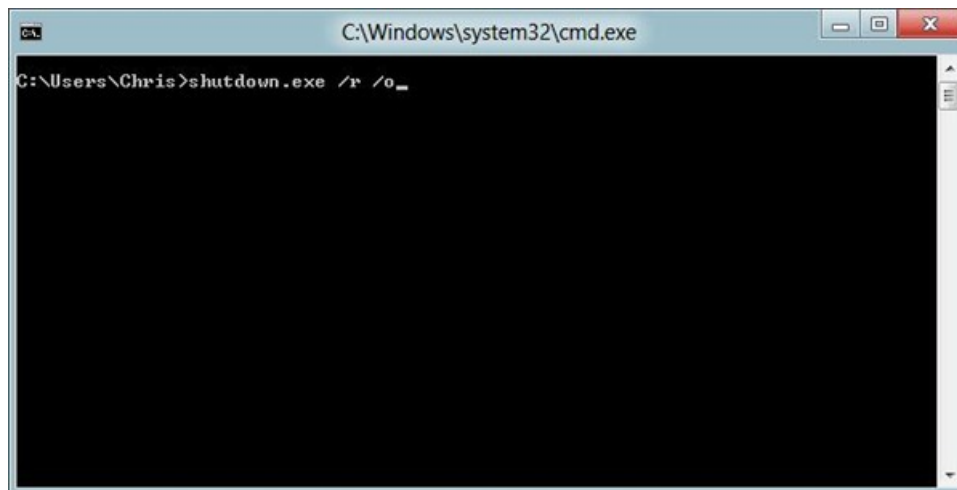


The reason that we added this Shift+Restart option to the shutdown menu was because the boot options need to be available even when no one has signed in to the PC. In the old hardware model that allowed keystrokes in boot, anyone with physical access to the PC could press a key to interrupt boot and use the available boot options. To preserve those scenarios, we needed a way for someone who hasn't signed in (but is still physically using the PC) to use the boot options menu.

The shutdown menu fits these requirements perfectly – it's always available from the login screen, even when no one is signed in. Also, the use of the Shift modifier on Restart fits with the pattern of using Shift on other items in that same menu. You may notice that the shutdown menu appears in many other places as well, for users who are signed in as well as users who aren't. In all of these places, the same Shift+Restart behavior still works – we felt it was important for the shutdown menu to behave consistently and predictably, wherever it appears.

There's one other way to trigger the boot options menu during shutdown, and this way has the added bonus of working from Command Prompt. We've added a new flag to shutdown.exe: /o. The /o flag only works in conjunction with /r (for restart), so the full syntax is:

```
Shutdown.exe /r /o
```



We added this new flag to shutdown.exe because we wanted to keep this part of Windows consistent and predictable. Not everyone uses Shutdown.exe, but those who do, depend on it for the full set of shutdown-related tasks.

- Chris Clark

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

Delivering the Windows 8 Release Preview

Steven Sinofsky | [2012-05-31T12:01:00+00:00](#)

Today, [Windows 8 Release Preview](#) is available for download in 14 languages. This is our final pre-release, and includes Windows 8, Internet Explorer 10, new Windows 8 apps for connecting to Hotmail, SkyDrive, and Messenger (and many more), and hundreds of new and updated apps in the Windows Store. Since our first preview release last September, millions of people now use the pre-release product on a daily basis and millions more have been taking it through its paces, totaling hundreds of millions of hours of testing. We genuinely appreciate the effort that so many have put into pre-release testing, and of course, we appreciate the feedback too. Direct feedback and feedback through usage contributed to hundreds of visible changes in the product and tens of thousands of under-the-hood changes.

Just nine months ago, we kicked off this blog as a dialog about the design and development of Windows 8. We've talked in depth about building Windows 8, including the features, the designs, and the background behind these. We've done so in over 70 posts totaling over 500 pages if printed out and 34 videos totaling over 90 minutes, all coming directly from engineers of the product. We've had about 18,000 comments from approximately 7,000 people. Over 170 Windows engineers contributed to the dialog, including over 200 comments I posted (though I was out-commented by one other pretty active reader!). Of course, we've been carefully watching the telemetry of the millions of tech enthusiasts using the product at each milestone.

Windows is unique in this way. No other product used by so many provides such an inside view of the choices and development of the product as it evolves—and sometimes we forget that we are talking about a product still under active development even while we are discussing the designs and actively using it. The affirmations, debates, and even disagreements play a crucial role in the development of Windows. This has never been truer, as we reimagine Windows from the chipset to the experience—new hardware support, new user interaction models, new scenarios, new APIs and more, are all enabled with Windows 8, while we bring forward **and improve** the way Windows 7 has been used on over 550 million PCs around the world. Coming soon, we will see a new wave of PCs designed for Windows 8, along with new apps powered by the new Windows 8 platform.

The team has the deepest respect for, and is always humbled by the responses on the blog and in the stories about the posts. Thank you!

Our next milestone is traditionally called RTM, Release to Manufacturing, and from today until RTM, we will still be changing Windows 8, as we have done in past releases of Windows. We thought it would be a good idea to outline the kinds of feedback we are acting on as millions download and use the Windows 8 Release Preview.

Our focus from now until RTM is on continuing to maintain a quality level *higher* than Windows 7 in all the measures we focus on, including reliability over time; security to the core; PC, software, and peripheral compatibility; and resource utilization. We will rely heavily on the telemetry built into the product from setup through usage to inform us of the real world experience over time of the Release Preview. In addition, we carefully monitor our [forums](#) for reproducible reports relative to PC, software, and peripheral compatibility. We'll be looking hard at every aspect of Windows 8 as we complete the work on the product, but we want to highlight the following:

- **Installation** – We have significant telemetry in the setup process and also significant logging. Of course, if you can't set up Windows 8 at all, that is something we are interested in, and the same holds for upgrades from Windows 7. Please note the specifics regarding installation requirements and cautions found on the download page.
- **Security and privacy** – Obviously, any vulnerability is a something we would want to address. We will use the same criteria to address these issues as we would for any in-market product.
- **Reliability and responsiveness** – We are monitoring the “crash” reports for issues that impact broad sets of people. These could be caused by Windows code, Microsoft or third-party drivers, or third-party apps. Information about crashes streams in “real time” to Microsoft, and we watch it very carefully. We also have a lot of new data coming on the hundreds of new apps in the Windows Store.
- **Device installation and compatibility** – When you download a driver from Windows Update or install a driver via a manufacturer's setup program, we collect data about that download via the Plug and Play (PnP) ID program. We've seen millions of unique PnP IDs through the Consumer Preview. We also receive the IDs for devices that failed to locate drivers. We are constantly updating the Plug and Play web service with pointers to information about each device (driver availability, instructions, etc.) We actively monitor the use of the compatibility modes required when the first installation of a Windows 7 based product does not succeed.
- **Software compatibility** – Similar to device compatibility, we are also monitoring the installation process for software, and noting programs that do not install successfully. Again, we have the mechanism to help move that forward, and/or introduce compatibility work in the RTM milestone. Here too, we actively monitor the use of compatibility modes required when the first installation of a Windows 7-based product does not succeed. We have tested thousands of complex commercial products from around the world in preparation for the Release Preview.
- **Servicing** – We will continue to test the servicing of Windows 8 so everyone should expect updates to be made available via Windows Update. This will include new drivers and updates to Windows 8, some arriving very soon as part of a planned rollout. Test updates will be labeled as such. We might also fix any significant issue with new code. All of this effort serves to validate the servicing pipeline, and to maintain the quality of the Release Preview.
- **New hardware** – Perhaps the most important category for potential fixes comes from making sure that we work with all the new hardware being made as we all use build 8400. Our PC manufacturing partners and hardware partners are engineering new PCs, and these include hardware combinations that are new to the market and new to the OS. We're working together to make sure Windows 8 has great support for these new PCs and hardware.

In fact, as some have noted, the RP itself was compiled over a week ago (build 8400). It takes time to complete the localized builds, validate the download images and process, as well as gear up all along the network edge for a fairly significant download event.

The path to RTM is well defined and critical to the careful and high quality landing of Windows 8 for our PC manufacturing partners. The changes we make to the product from RP to RTM are all carefully considered and deliberate, including some specific feature changes we plan on making to the user experience (as we talked about in previous posts). This is a routine part of the late stages of bringing a complex product like Windows to market. Throughout this process, every change to the code is looked at by many people across development and test, and across many different teams. We have a lot of engineers changing a very little bit of code. We often say that shipping a major product means “slowing everything down.” Right now we’re being very deliberate with every change we make and ensuring our quality is higher than ever as we progress towards RTM. The product is final when it is loaded on new PCs or broadly available for purchase.

RTM itself is a product development phase, rather than a moment in time. We continue to roll out Windows 8 in over 100 different languages and we are preparing final products for different markets around the world. As that process concludes, we are done changing the code and are officially “servicing” Windows 8. That means any subsequent changes are delivered as fixes (KB articles) or subsequent servicing via Windows Update. Obviously, our ability to deliver fixes via Windows Update has substantially changed the way we release to manufacturing, and so it is not unreasonable to expect updates soon after the product is complete, as occurred for Windows 7. There are no surprises here, but we’re making sure readers of this blog know what is coming down the road.

Once we have entered the RTM stage, our partners will begin making their final images and manufacturing PCs, and hardware and software vendors will ready their Windows 8 support and new products. We will also begin to manufacture retail boxes for shipment around the world. We will continue to work with our enterprise customers as well, as we ensure availability of the volume license tools and products.

Remember, if you buy a new PC running Windows 7 today, with the great support from our PC partners, you will be ready for Windows 8.

Delivering the highest quality Windows 8 is the most important criteria for us at this point—quality in every dimension. The RTM process is designed to be deliberate and maintain the overall engineering *integrity* of the system.

Ultimately, our partners will determine when their PCs are available in market. If the feedback and telemetry on Windows 8 and Windows RT match our expectations, then we will enter the final phases of the RTM process in about 2 months. If we are successful in that, then we are tracking to our shared goal of having PCs with Windows 8 and Windows RT available for the holidays.

On behalf of the Windows team,

Steven Sinofsky

PS: Please be sure to check the download page for system requirements, release notes, upgrade instructions, and other details on how to install and use the Release Preview.

Web browsing in Windows 8 Release Preview with IE10

Steven Sinofsky | [2012-06-01T10:01:00+00:00](#)

In the Windows 8 Release Preview, we continue to deliver the re-imagined experience of the web browser, incorporating your feedback to provide the best browsing across all Windows 8 devices, including more of the web you browse every day. Rob Mauceri, the group program manager for Internet Explorer, authored this post.

--Steven

We built a new browsing experience in lockstep with Windows 8 to give you all the advantages that Metro style apps offer. We built that experience by extending IE's underlying architecture to provide a fast, fully hardware-accelerated browsing engine with strong security and support for HTML5 and other web standards.

Internet Explorer 10 is designed to make website interaction fast and fluid for touch as well as for heavy mouse and keyboard use. With IE10, websites participate in the Metro style experience in Windows 8, including the Start screen, charms, snap, and more. IE10 also provides the best protection from malicious software on the web while providing convenient control over your online privacy.

The Metro style browsing experience is a better way to browse on a desktop computer with a big screen, mouse and keyboard, or on a touch-enabled mobile device. As people browse more "chromelessly" on their phones, they've become accustomed to a more immersive and less manual browsing experience compared with the desktop. Metro style browsing offers you a full-screen, immersive site experience with every pixel of the screen for your favorite sites. With IE and Windows 8 you can always use the charms to accomplish what you want to do next with a website (e.g. share, print, search. . .). We've heard from many people – even those with the most enthusiastic and intense browsing patterns – prefer Metro style browsing because it's fast, fluid and more focused on what you browse than on how you browse.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

Note: On June 7, 2012, we updated the video above to provide a more illustrative demo of interacting with videos in IE10.

Browsing more of the web

The Windows 8 Release Preview includes a new power-optimized, touch-friendly Adobe Flash Player for IE10 that is updated through Windows Update. Adobe Flash content on compatible websites will now play in the new Metro style web browser. This optimized Flash Player is integrated with IE 10 in Windows 8 to ensure that our customers have a great experience browsing the web on Windows 8. We believe that having more sites "just work" in the Metro style browser improves the experience for consumers and businesses alike.

As a practical matter, the primary device you walk around with should play the web content on sites you rely on. Otherwise, the device is just a companion to a PC. Some popular websites require Adobe Flash and do not offer HTML5 alternatives, and this change to the product reflects the feedback that we've heard from customers about their experience with sites that do not offer an HTML5-only experience for Metro style IE. For example, try pbskids.org on an iPad. Some workforce solutions, like [Beeline](#), require Flash. Some financial management sites, like [this one](#), require Flash. And some sites still deliver their best experience with Flash, such as youtube.com.

You can read more about the technical details and architectural improvements to the underlying HTML5 "Trident" browser engine and Chakra JavaScript engine, including support for integrating the Flash Player on the [IE Blog](#).

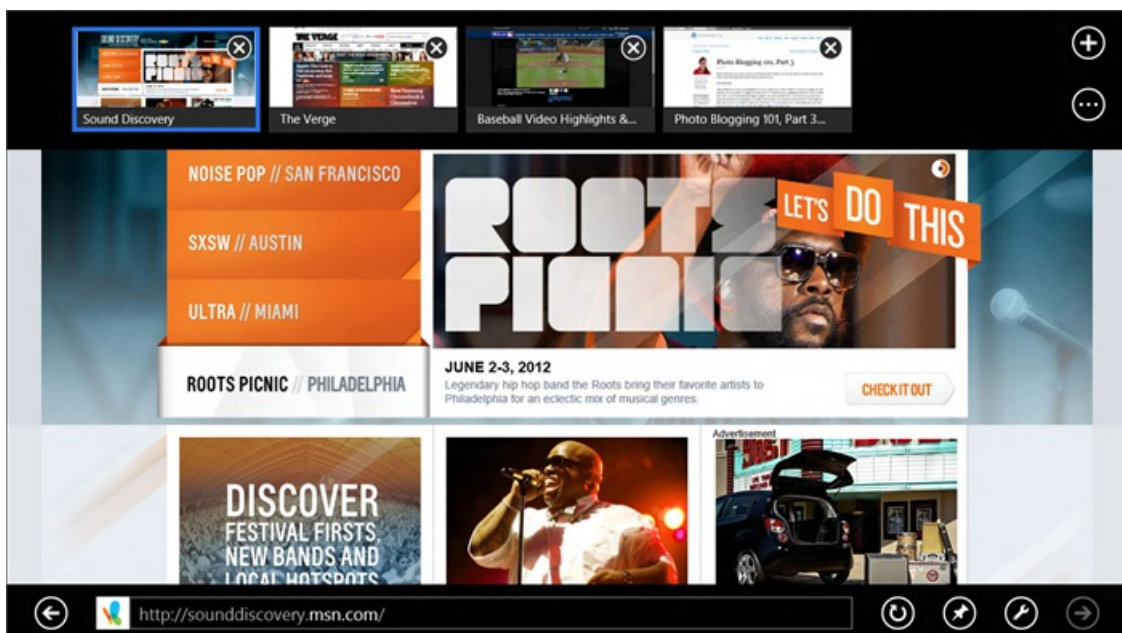
The Metro style browser for Windows 8

We built IE10's user experience exclusively around all the Metro style design patterns to be fast and fluid for even the most intense everyday browsing. We listened to [your feedback](#) from the Consumer Preview and acted, adding more conveniences like saving images from web pages, "paste and go" for faster navigation, and integrated network diagnostics.

We designed the interface and controls to be there when you need them and out of view when you don't. We also designed in the comprehensive functionality that people need for everyday heavy-duty web browsing: great touch keyboard support for forms, integrated spell checking with AutoCorrect, finding text on the page, etc. The user experience follows Metro style patterns and conventions for personality, animations, and command activation, and support for Windows 8 charms, snap, and more.



IE10 puts the focus on your sites, providing a full screen edge-to-edge experience that uses every pixel for the web.

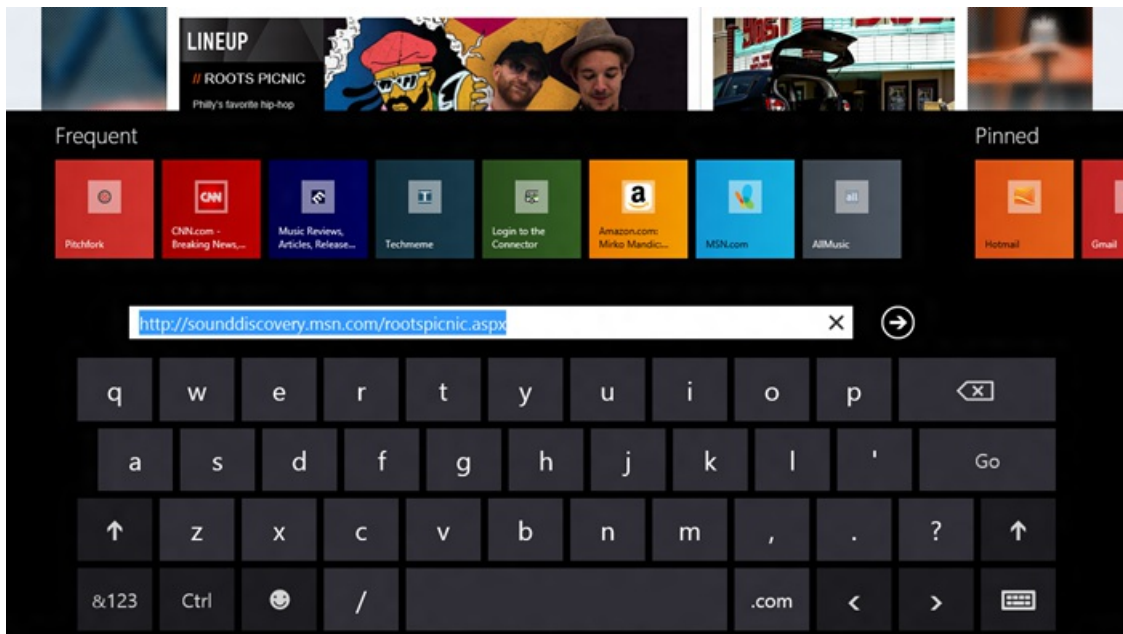


Tabs are available, and stay out of your way until you need them.

IE10 is fast and fluid for the full web, not just the mobile versions of sites. We made IE super responsive to touch, mouse, and keyboard. The Metro style browser delivers on *touch browsing*, not just browsing on a touch device. You can feel it in the stick-to-your-finger responsiveness of the touch support for panning and zooming, swiping back and forward for page navigation, and double tapping to zoom in and out of content. The Release Preview includes improved visual feedback when following links with touch, for higher confidence even when the site isn't coded for touch. Context menus and form controls are optimized for touch, and the browser responds fluidly to device orientation (scaling smoothly to landscape and portrait screen layouts) and "snapping" Windows 8 applications next to it. IE10 also improves on the experience of browsing the Web with mouse and keyboard with support for the keyboard shortcuts you expect, and convenient mouse affordances for back and forward navigation. These are also improved in the Release Preview; for example, you can just slam your mouse to the left edge of screen and click on the back button, which is now smaller.

Metro style IE10 takes a different, more modern approach to browsing. It puts the focus squarely on the websites you browse rather than the tab and window management intensive activity that has defined browsing for the last decade. For example, in the Release Preview, you can double-tap to focus on HTML5 video with full-screen playback. On our hallways, we've been using it as our primary browser on laptops and desktop workstations, with touch screens as well as with keyboards and mice. From tiles on the Start screen for websites to the immersive full screen web experience, we designed IE in Windows 8 to be your daily browser for the real web.

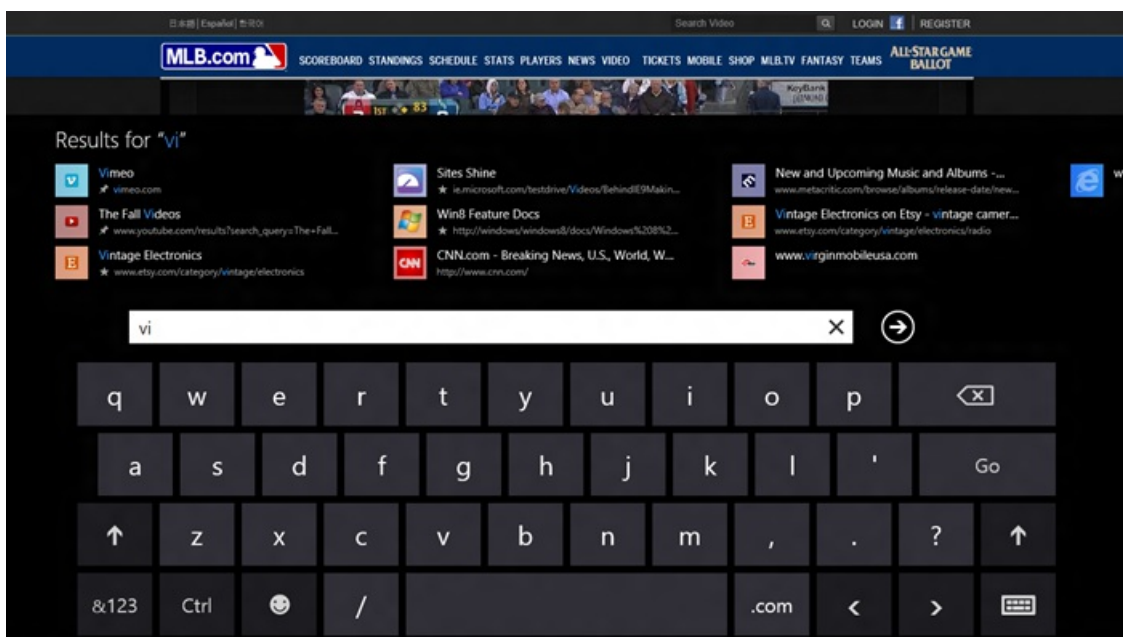
Navigation: Navigation tiles are designed to help you find and navigate to sites immediately using the site's icon and color while minimizing your typing. In the Release Preview, we improved the layout of these tiles for efficiency and speed, optimized for visual recognition of sites you visit most often, and with clearer consistency with the Start screen.



Get to your most important sites quickly with navigation tiles.

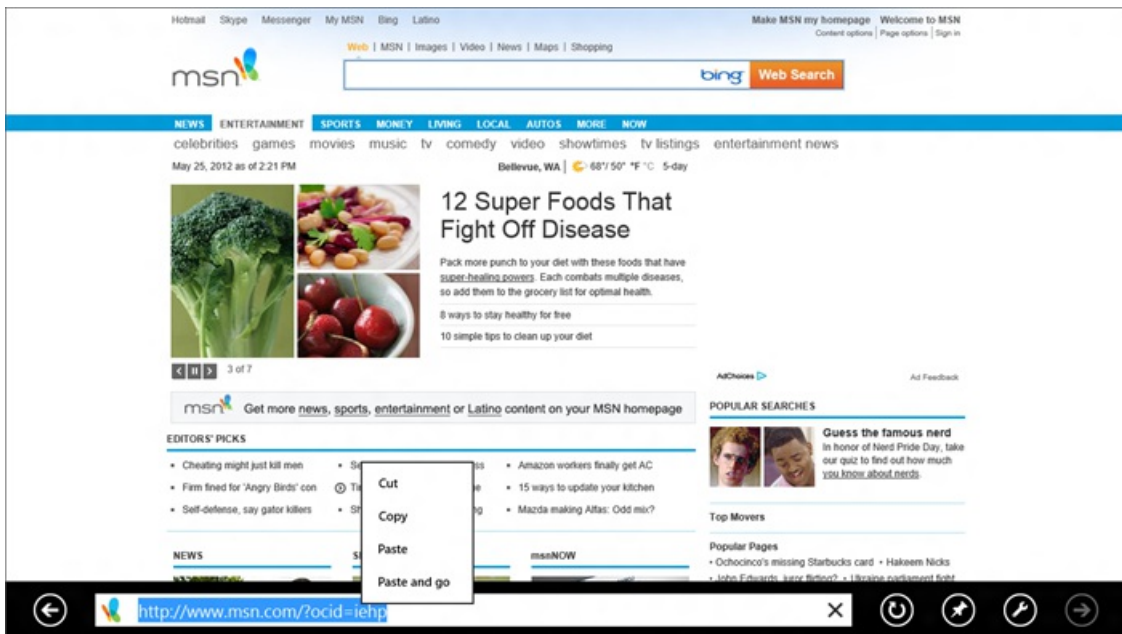
You can quickly access your favorites by typing the first few letters of the name and in the Release Preview, favorite and pinned sites are marked with a badge for quick recognition. IE shows you frequently visited sites as well as sites that you've pinned to the Start screen. As you type in the address bar, the navigation tiles filter to show you sites from your history, favorites and even popular URLs.

With Windows 8 roaming and connected accounts, your browsing history and favorites roam with you so that you can easily access recent webpages across all of your PCs.



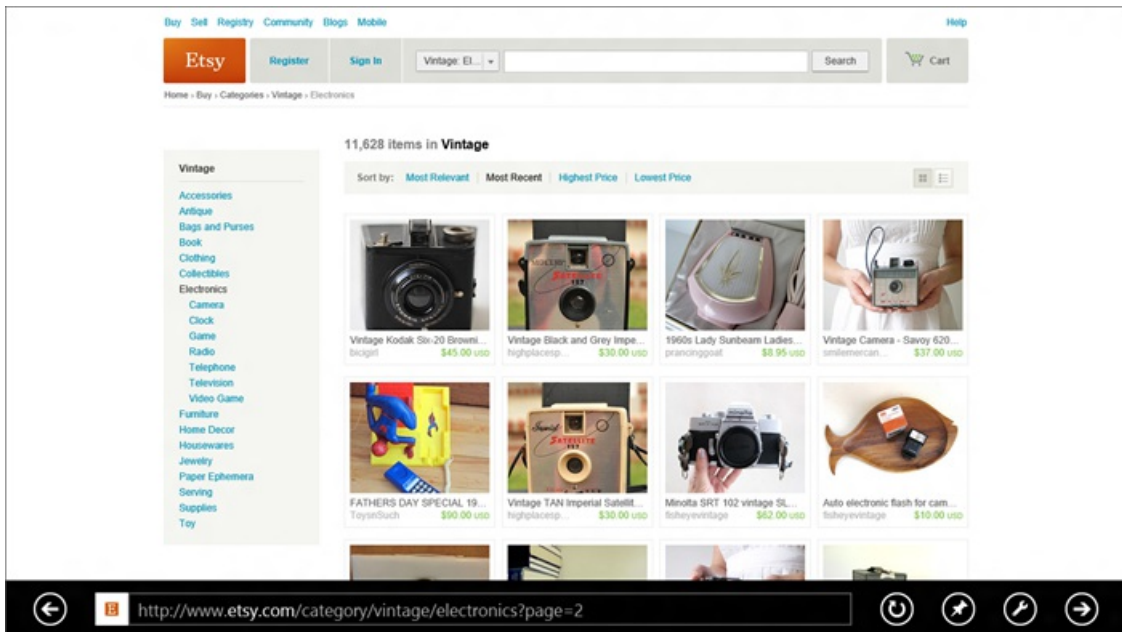
Access your favorites by typing the first few letters of the name. Favorite and pinned sites are marked with a badge for quick recognition.

The **navigation bar** in IE10 appears when you need it, again keeping the focus on websites. In the Windows Release Preview, the navigation bar consolidates easy-to-use controls (touch or keyboard/mouse) for common operations like back, forward, stop/refresh, pinning sites to the Start screen, and getting an app. The address bar shows badges and coloring for secure sites, SmartScreen, and InPrivate browsing. It also supports auto-complete as well as web search, matching the behavior of IE on the desktop. Also new in Release Preview is the "Paste and Go" command for fast navigation to copied URLs or search terms on the clipboard. The address box shows a progress indicator when a page is loading, and includes indicators for site compatibility and tracking protection. The navigation bar includes commands for Find on Page, and Open in IE on the desktop, for compatibility with sites that require legacy plug-in technologies, or for when you are using desktop tools and wish to continue using them in your existing workflows.

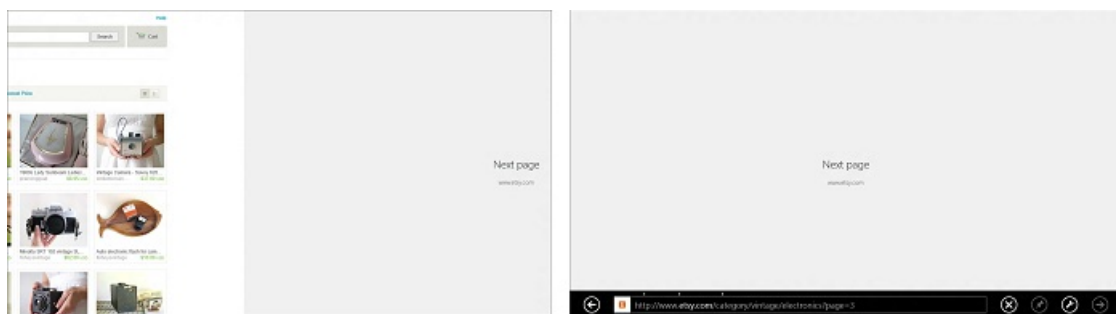


Navigation is faster with Paste and go.

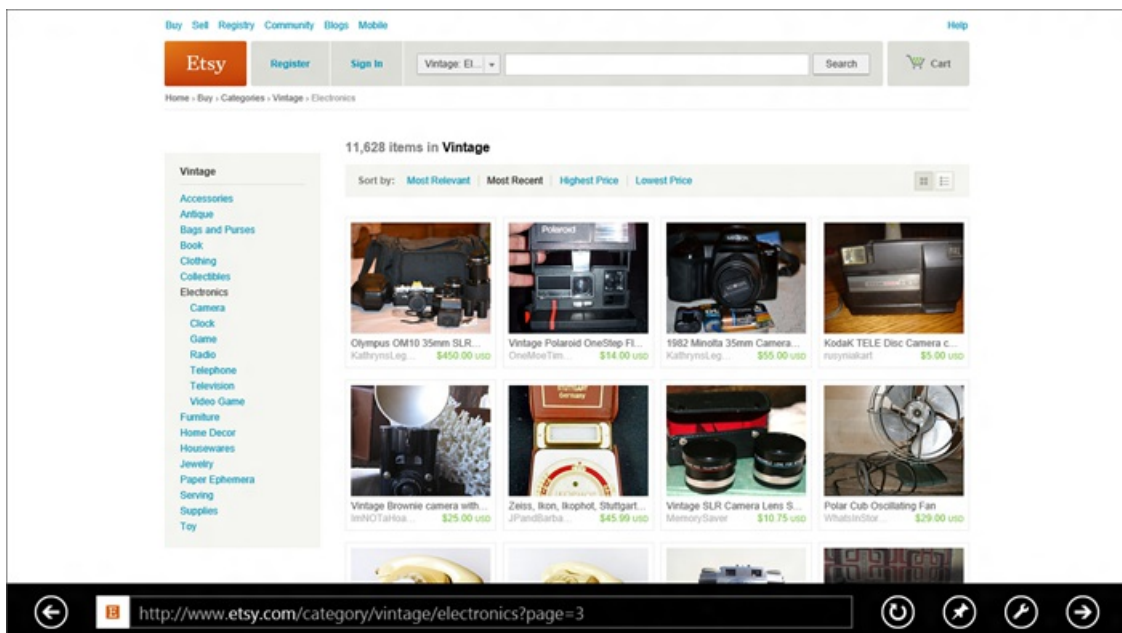
Enhanced touch browsing: In the Release Preview, IE10's Metro style experience offers a new way of browsing multi-page and sequenced content. Flip ahead enables you to navigate your favorite sites like you read a magazine by replacing the need to click on links with a more natural forward swipe gesture on touch-centric devices (and forward button with mouse). Imagine flipping through a multi-page New York Times article, through product listings on Amazon or eBay, or quickly catching up on the latest news by flipping through CNN.com, all by simply swiping forward without hunting for the "Next" link on the page.



You can swipe to flip ahead to next page listings without hunting for the "next" link.



Transitioning to next page is fast and fluid with touch.

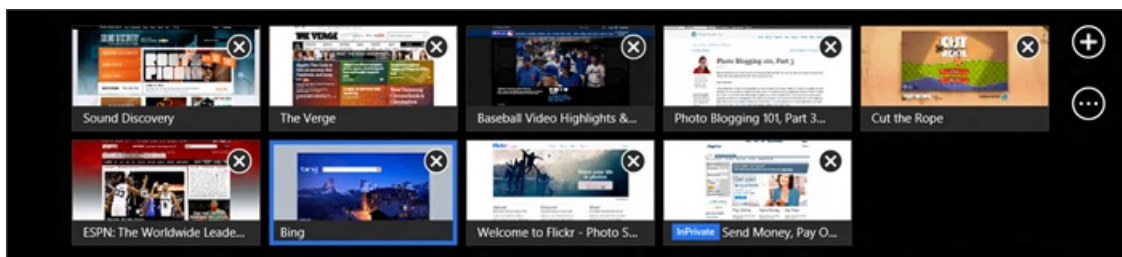


Moving through multi-page content is fast and fluid.

Once you turn on Flip Ahead, you can swipe through content spread across multiple pages to go to the next page within the same article, post or thread. When browsing sequenced content, such as blogs or news sites, and whenever you've reached the end of your multi-page content, flip ahead will suggest an appropriate next article, post or thread to continue your exploration. Using Flip Ahead requires end user opt-in, and sends your browsing history to Microsoft to improve the quality of the experience.

Other aspects of the Metro style experience have largely stayed the same:

Tabs: Browsing multiple web pages is core to any good web experience. The Metro style tab switcher appears when you swipe in from the bottom or top of the screen with touch, right-click with the mouse, or press Windows key+Z on the keyboard:



Active tabs are shown as page thumbnails with page titles in text overlays. Tabs have a touch-friendly button for closing, and button for creating a new tab, or a new InPrivate tab. IE10 shows the last 10 tabs you've used, reducing the need to actively manage your tabs. You can even clean up tabs quickly and easily with one command.

Touch keyboard: IE10 works great with physical keyboards as well as the Windows 8 touch keyboard, which it automatically adjusts to make your experience easier. For example, when you set focus in the address bar, the “?” and “.com” keys become available to quickly enter URLs:



IE automatically adjusts the touch keyboard based on where you're typing. For example, email form fields show the “@” and “.com” keys.

IE10 takes a clean, “low nag” approach to **notifications**. All alerts and user prompts come through a notification bar at the bottom of the screen. IE uses Windows 8 Metro style “fly-outs” when more interaction is needed. Notification bars automatically dismiss as appropriate. Downloads in the Metro style browser protect you from malicious software via SmartScreen’s Application Reputation, as in IE on the desktop. The Release

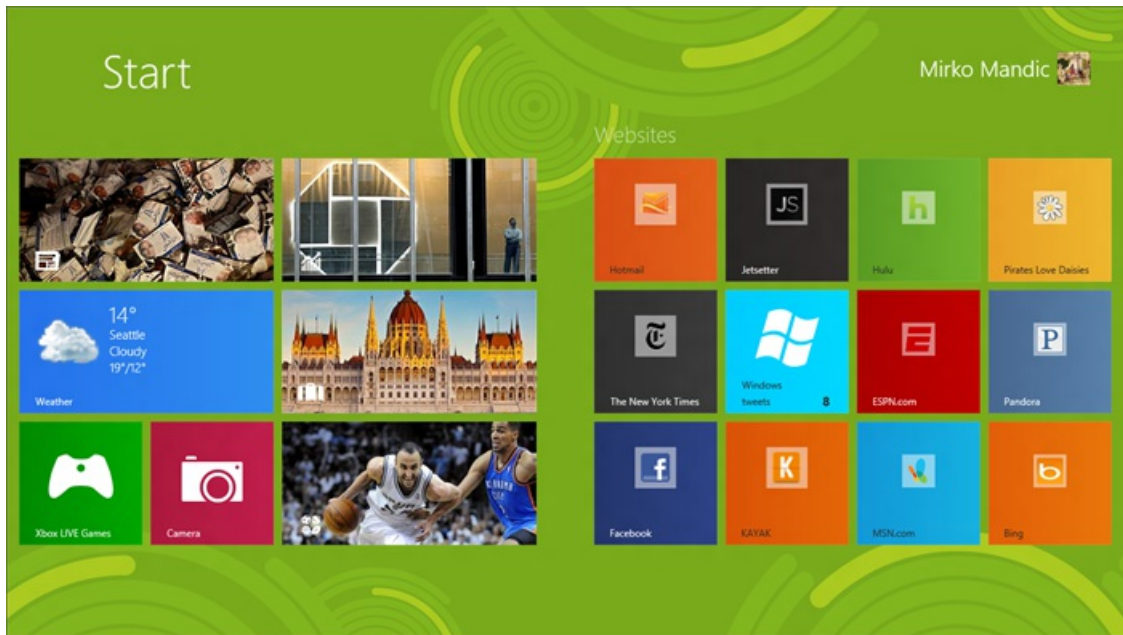
Preview adds support for “pop-up” windows as background tabs in the Metro style experience.

Connecting websites and apps in the Metro style

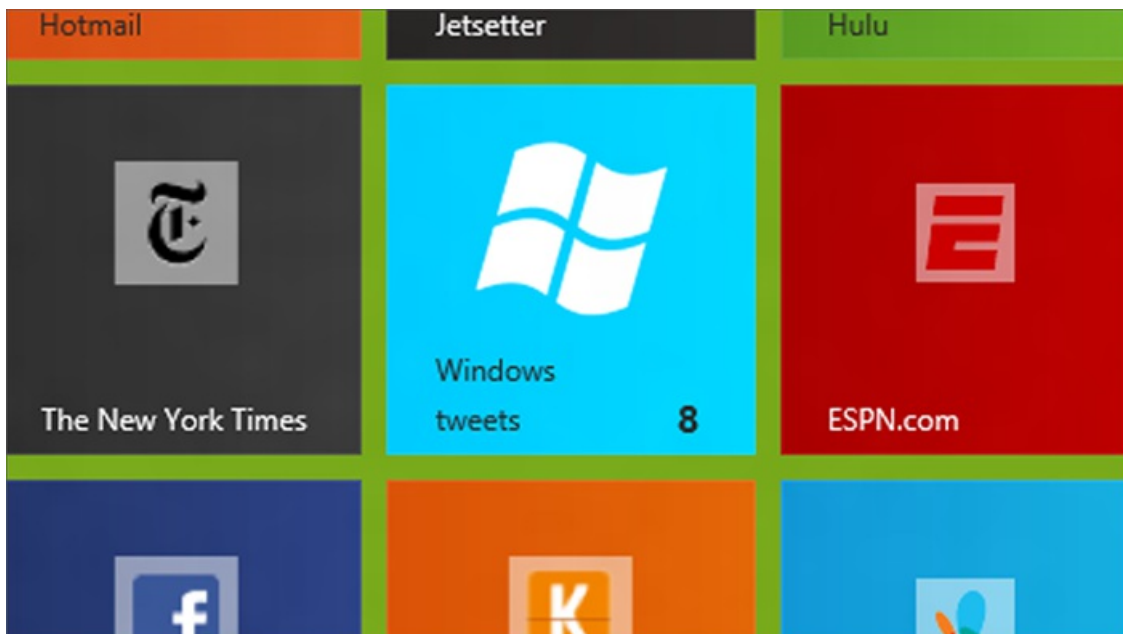
With IE10, websites are part of the Metro style experience in Windows 8. Through snap, charms, and integration with the Store and the Start screen, Metro style browsing blurs the boundaries between the web and apps.

With **site pinning**, you can personalize your Windows Start screen with the sites you use all the time. You can pin any website to the Start screen from IE10, so you have one place to access all the things you care about or need.

The tiles for pinned sites reflect the site’s color and icon. In the Release Preview developers can provide higher resolution PNG file site icon and specify the tile background color. With IE10, sites can provide background notifications for new messages and other account activity on the website. The site can also program additional commands that appear in IE’s navigation bar in a touch-friendly way, the same way that sites can program jumplists for IE on the desktop.

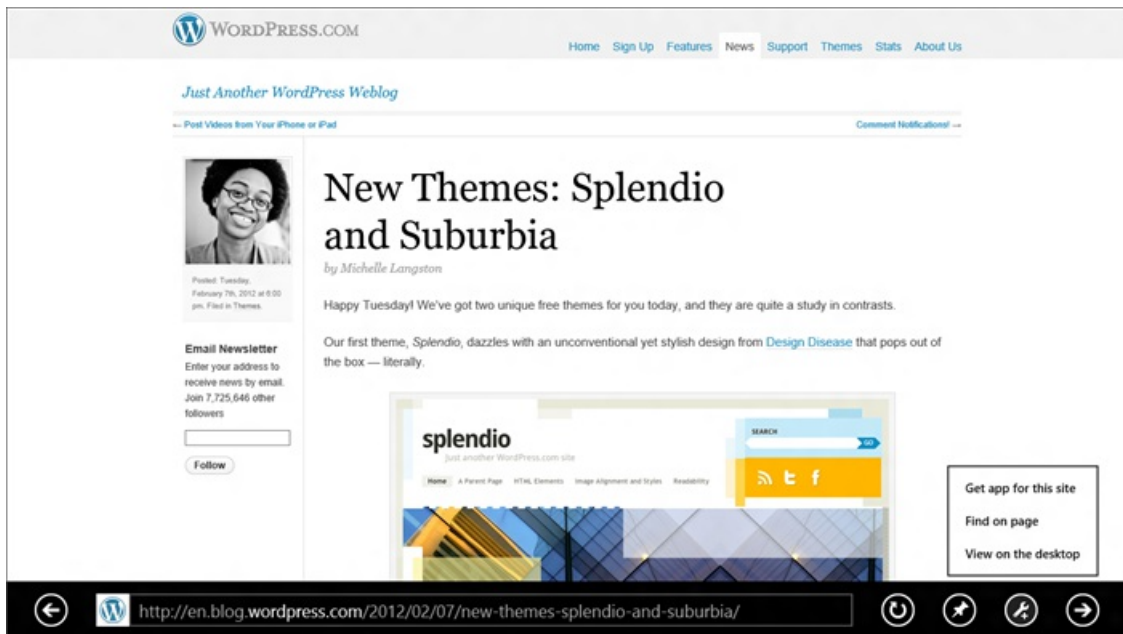


Site tiles let you go directly to your sites from the Windows 8 Start screen.



Pinned site high-resolution image, custom tile colors, and background notifications make it easier to find your favorite site and keep you up-to-date at a glance.

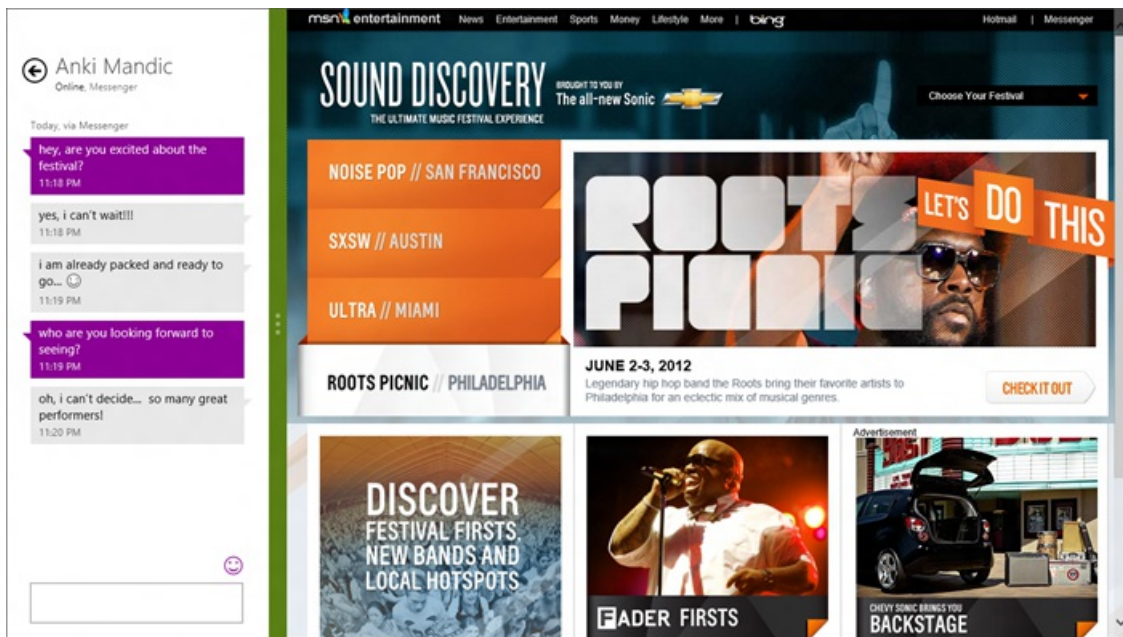
Integration with the Store makes it easy to discover and launch Metro style apps for the sites you visit in IE. In Release Preview, we have continued reducing UI concepts to make you even confident in IE. The Tools icon is updated to tell you when there is something "special" about the page. You can tap on it to go to the store to install an app. Once an app is installed, you can launch it directly from the site. For example, here's WordPress.com in IE10:



Updated tools menu lets you know when there is an app for the site.

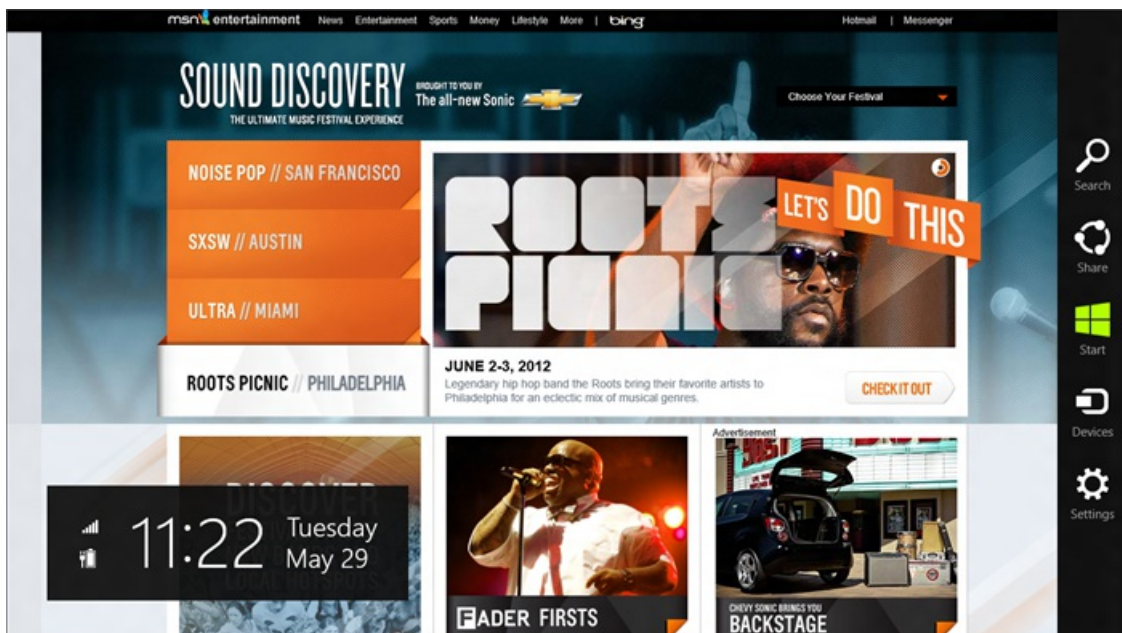
Other aspects of integrating sites in the Metro style experience include:

Snap makes it easy to use Windows 8 for more than one thing at a time. You can browse in IE10 and have side-by-side access to your mail, music, or any other application. The browser adapts to the narrow “snap” size and automatically undocks when necessary for user interaction. All of the core browsing capabilities are available when snapped – panning, pinch and double-tap zooming, and following links.



Multitasking with Windows 8 “snap” lets you put your site side-by-side with other applications like the Messaging app.

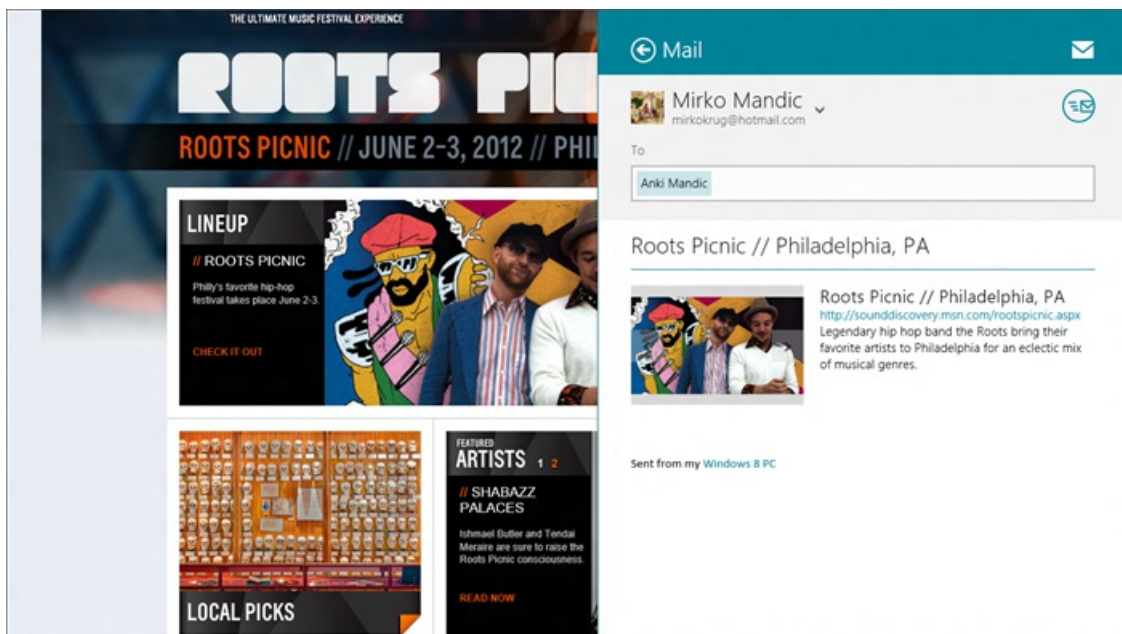
Charms provide a consistent way to perform common actions like searching and sharing in Windows 8. IE10 supports the Search, Share, Devices, and Settings charms:



The charms appear when you swipe in from the right edge, press Windows key+C, or move your mouse to the bottom or top-right corner of the screen.

For the **Search** charm, IE10 uses the default search engine, which you can set to your preference. After initiating a search in the charm fly-out, search results are shown as you type, including the same picture and instant results you see in IE on the desktop, if your search engine supports them.

With the **Share** charm, you can access any application that supports sharing (like Mail). This allows you to send a rich link preview with image, description, and hyperlink so it's easy to share more than just a link.



IE10 and Mail support sending rich link previews with image, description, and hyperlink, you can share more than just a link with very little work.

The **Devices** charm makes printing, projecting, and playing to external devices easy and consistent. For example, you can print from any webpage from IE – handy for things like airline boarding passes – by tapping or clicking the Devices charm and selecting a printer.

The **Settings** charm provides quick access to the most frequently used configuration settings for IE10. You can quickly clear browsing history, control location access, and more. Consumers get a simplified interaction with IE settings, while enthusiasts still have an easy way to access fine-grained controls through settings in IE on the desktop.

Protection from the malicious web

IE10 offers the same industry leading security, privacy, and reliability features, building on IE9's SmartScreen, XSS filtering, Application Reputation, InPrivate browsing, Tracking Protection, and hang detection and recovery. In addition, IE10 makes your security and privacy more convenient with "Enhanced Protected mode" for better isolation of website content in each tab, and new in the Release Preview is one simple

setting (on by default) for sending the Do Not Track (DNT) signal to web sites. .

Summary of other key changes from the Consumer Preview

IE10 in the Windows 8 Release Preview brings a more full-featured Metro style experience to your browsing. Again, you can read more details of changes to the underlying HTML5 engine and more on the [IE blog](#). Here are just some of the improvements to IE10 for fast and fluid browsing:

- Improved Fast & fluid touch: full independent composition for real web sites (including fixed elements, sub-scrollers, animations, and video)
- Smoother UI transitions and animations with less flicker on low-end hardware
- Support for subset of Flash in Metro style IE for top sites for media playback and gaming
- Support for full-screen HTML5 video, including double-tap zoom to full-screen
- Improved layout for site selection with “light dismiss” and notation for Favorites and Pinned site
- Improved browser command bar layout and favicon treatment, with consolidated navigation bar controls
- Adjust default web page zoom level on high res screens
- Context menu for “Save Image”
- Context menu for “Paste and Go”
- Improved touch visual feedback for following links
- Support for high-res image for pinned sites tile in start screen
- Integrated network trouble shooter in Metro style IE
- Metro style auto-complete drop down
- Flip ahead for next page navigation (user opt-in)
- Do Not Track (DNT) setting on by default

Metro style and no-compromise browsing

You used to have to make a choice between browsing the mobile web on small screens with good touch support, and browsing the full web with good mouse and keyboard support on big screens. The Metro style web experience in IE10 in the Windows 8 Release Preview means no compromises. You can browse and touch and multitask and print and share with all the power of Windows 8 and your PC. The web with IE10 is more fast and fluid, better connected to your applications, and more secure and private.

--Rob

Connecting with IPv6 in Windows 8

Steven Sinofsky | [2012-06-05T09:00:00+00:00](#)

*With [World IPv6 Launch](#) upon us, we thought it would be good to provide a look at the work in the Windows 8 Release Preview supporting IPv6. **Christopher Palmer on the core networking program management team authored this post.***
--Steven

IPv4 is the Internet Protocol that has been used for Internet connectivity for decades. However, IPv4 was never designed for such load and scale, and it is beginning to show signs of strain as the Internet grows—even though the incredible foresight of the original designers continues to power the Internet at a massive scale. Internet service providers are finding IPv4 increasingly costly to maintain; it will require an overhaul to sustain the upcoming onslaught of connected PCs and devices.

For several years, the industry, including Microsoft, has been working to roll out a completely new version of the Internet Protocol – IPv6 – across various devices, services, and network infrastructure. Windows releases since Windows XP SP3 have supported IPv6, making the IPv6 transition possible. We have engineered Windows 8 to keep you (and your apps) reliably connected as this dramatic transition takes place.

The limitations of IPv4

First, let's cover some basics. Every time you browse to a website like www.bing.com, that friendly name gets turned into an IP address, something like **23.3.105.97**. An IP address is conceptually similar to a telephone number. Just as all your contacts have telephone numbers, everything that connects to the Internet has one or more IP addresses. The “telephone directory” for the Internet is the Domain Name System (DNS). Given a name, DNS resolves the name to a set of IP addresses.

IPv4 only provided around 4 billion IP addresses. That seemed like a lot in the 1970s. But by 2015, [an estimated 15 billion devices will be connected](#) (PCs, phones, household appliances, cars, even furniture!). IPv4 simply does not have the addresses necessary to connect this many devices to the Internet.

As demand for IPv4 addresses has grown in recent years, the Internet community has found ways to “share” those vital resources. The most common way to share an IPv4 address is to use network address translation (NAT). This functionality is in most home routers, enabling computers and other devices in a household to share a single public IPv4 address.

Conventionally, ISPs provide a single IP address to each home. However, that is becoming increasingly difficult. Because of IP address depletion, unique IPv4 addresses simply aren't available for each home. Soon, whole cities or countries may be behind large-scale network address translation. Internet service providers have to develop costly and complex infrastructure to continue to support IPv4. For end users, IP address exhaustion means that location-based services, such as Bing, will not work properly, and peer-to-peer applications will face degraded performance.

IPv6 is the future

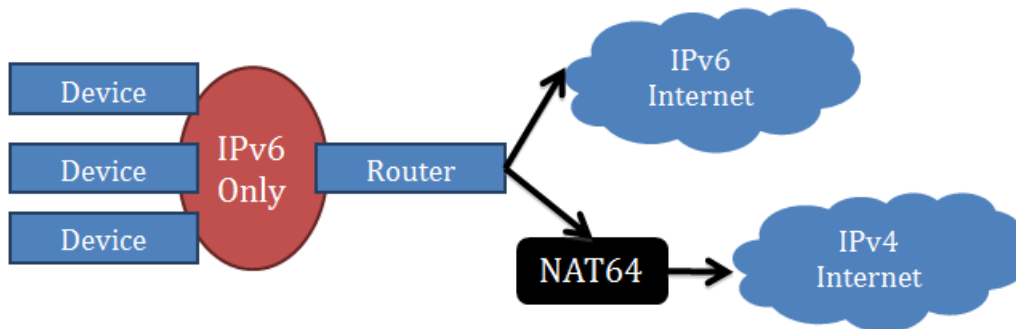
Microsoft, along with other technology companies, has been working on the deployment of IPv6 to ensure that end-users continue to have high-quality Internet access, despite the performance and connectivity limitations brought about by IPv4 address exhaustion.

The most immediate benefit of IPv6 is that it provides more than 3×10^{38} IP addresses, enough for every person to have billions of addresses all to themselves, or enough to give every star in the universe a unique address. This will allow the Internet to grow and evolve. IPv6 also provides for many security and performance improvements, like built-in support for IPsec. (What happened to IPv5, you ask? [Bing](#) can help you find out why it's being “skipped.”)

Upgrading the entire Internet to IPv6 isn't something that can be done instantly. It has taken many years to get to where we are today, and we still have many years of work to do. Currently, around 1% of devices can connect to the Internet using only IPv6.

During the transition period, most networks will fall into three categories:

- **IPv4-only networks.** This is probably what you have today, as most Internet Service Providers have only just started rolling out IPv6 support. Many devices that connect to the Internet might only support IPv4 as well.
- **IPv4 and IPv6 networks (dual-stack).** This means your Internet Service Provider is configuring your PC with both IPv4 and IPv6 addresses. This model is common in cable and dial-up networks that are transitioning.
- **IPv6-only networks.** This means your Internet Service Provider is configuring your device with **only** IPv6 addresses. Because many websites are still only on the IPv4 Internet, ISPs must use a translation device to allow access from your IPv6 network to the IPv4 Internet. This device is called a NAT64. This mode is becoming popular in the mobile environment, because having only one kind of Internet Protocol between the mobile device and the operator's infrastructure is simpler to deploy and cheaper than a dual-stack configuration. Also, mobile operators are feeling the IPv4 address exhaustion pinch most severely. Here is a basic diagram of this configuration:



You might be wondering what kind of connection you have right now. We have a widget at the bottom of this post that can show you.

Windows 8 is designed to ensure connectivity across all types of network configurations. In Windows 8, you can launch DNS look-ups using the *Resolve-DnsName* cmdlets in Windows PowerShell. Open up PowerShell and run the below command, and you will see both IPv6 and IPv4 records returned. Only websites that support IPv6 will have IPv6 records.

```
PS C:\> Resolve-DnsName www.xbox.com
```

Name	Type	TTL	Section	NameHost
www.xbox.com	CNAME	25	Answer	www.gtm.xbox.com
www.gtm.xbox.com	CNAME	25	Answer	msxbwsd.vo.11nwd.net


```

Name       : msxbwsd.vo.11nwd.net
QueryType  : AAAA
TTL        : 25
Section    : Answer
IP6Address : 2607:f4e8:130:202:230:48ff:fe92:f2c7
  
```

IPv6 Records

```

Name       : msxbwsd.vo.11nwd.net
QueryType  : AAAA
TTL        : 25
Section    : Answer
IP6Address : 2607:f4e8:130:202:230:48ff:fe92:fa11
  
```



```

Name       : msxbwsd.vo.11nwd.net
QueryType  : A
TTL        : 25
Section    : Answer
IP4Address : 68.142.93.196
  
```

IPv4 Records

```

Name       : msxbwsd.vo.11nwd.net
QueryType  : A
TTL        : 25
Section    : Answer
IP4Address : 68.142.93.195
  
```

Windows 8 on IPv4-only networks

On an IPv4-only network, devices are configured with IPv4 addresses only. This model continues to work in Windows 8 as it has in the past. In addition, Windows hosts also provide IPv6 connectivity by tunneling that traffic inside various transition technologies – an example of which is [Teredo](#), where IPv6 packets are encapsulated in IPv4 UDP packets. Now that we are starting to see the emergence of IPv6-only servers and services, Windows 8 automatically attempts IPv6 connectivity when the server does not offer an IPv4 address. Note that Teredo is enabled by default only on non-domain networks, and Teredo may not be available if your network blocks UDP.

Windows 8 on dual-stack networks

During the transition period, dual-stack networks will be the common deployment model. On a dual-stack network, devices will be configured with both IPv4 and IPv6 addresses.

Our primary focus during this transition has always been to minimize the impact of the transition for everyday users. It shouldn't matter whether your connection is over IPv4 or IPv6. You should have an Internet experience that is fast and reliable, with little evidence of the IPv6 transition, so you can just enjoy the content.

At the same time, it's also a priority for us to help the IPv6 transition move ahead. To this end, Windows prefers native IPv6 connectivity over IPv4 connectivity, if both connection modes are available.

In summary we have the dual goals of ensuring a reliable user experience, and enabling the IPv6 transition. As you might imagine, this can sometimes involve subtle tradeoffs, which have been the subject of much debate in the Internet community.

In an effort to sort out those sometimes competing goals, major websites around the world--including Bing.com, Microsoft.com, and Xbox.com--organized an event called [World IPv6 Day](#) last year. During this one-day test of the IPv6 Internet, participating websites turned on IPv6 in addition to IPv4.

The good news is that most things worked. All that goes into the Internet's correct functioning—servers, end-user devices, and content delivery networks—were able to work at scale without issue.

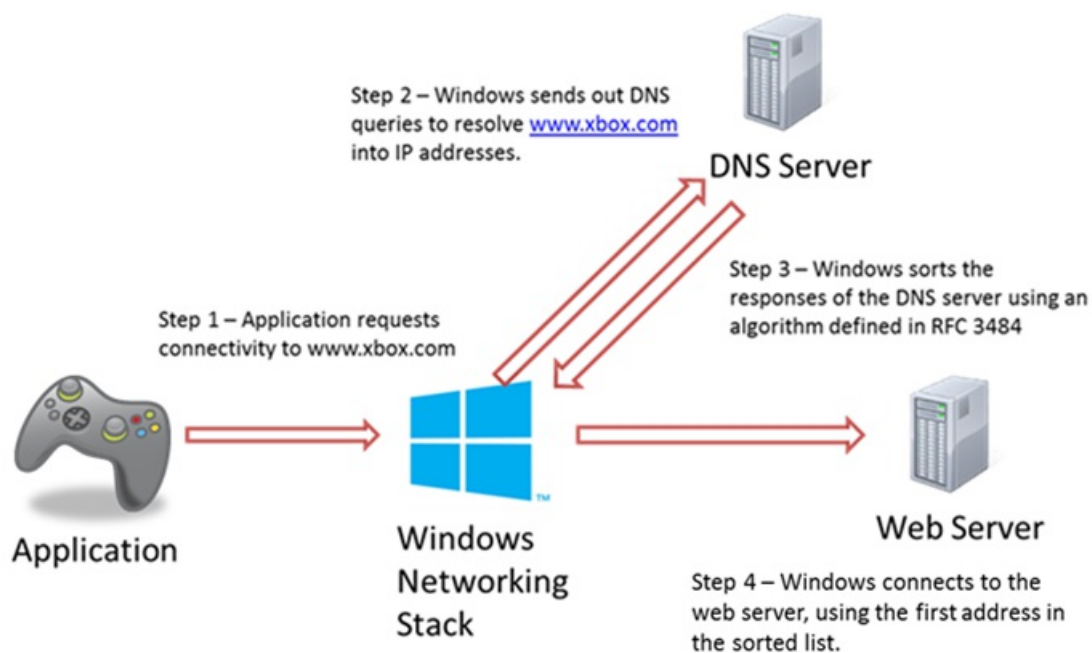
However, we also observed that a small subset of the population (0.01% of the world) was misconfigured with IPv6, seemingly because of a router or ISP issue. That's not too surprising, as IPv6 is a fairly new technology, and mistakes happen. But for those unlucky users, it could cause a significant impact on everyday experiences with the Internet.

Engineering resiliency into our connectivity algorithms for dual-stack networks

In order for a device to truly support dual-stack networks, apps must not only be able to send traffic with IPv4 and IPv6, but the OS must be smart enough to know which protocol is appropriate for the task at hand. Even more specifically, because your device might have multiple IPv4 and IPv6 addresses, and because the destination you're trying to reach might also have multiple IPv4 and IPv6 addresses, the stack must be smart enough to know which specific source and destination addresses should be used for connectivity. This functionality is called *address sorting*, and is an area that we have enhanced in Windows 8. The idea behind address sorting is to determine which address pair is likely to produce the best connection, so the application does not need to wait.

When Windows tries to connect to a dual-stack website, Windows sorts through its own and the website's IP addresses to decide which pair it should use to make the connection. (For standards buffs, address sorting is standardized in RFC 3484.)

Below is a diagram showing how Windows uses address sorting.



Traditionally, address sorting relies on Windows being correctly configured by your router. Windows analyzes the routing information provided by the router and uses that information in conjunction with address sorting to ensure fast connectivity to named resources. The RFC 3484 standard specifies that IPv6 should be preferred if IPv6 is configured by your router.

World IPv6 Day showed that some clients were configured with IPv6 routing information, but they did not actually have IPv6 connectivity to the Internet. This appears to be the result of a misconfiguration by some Internet Service Providers or buggy home routers. Windows attempts to connect to websites using IPv6, expecting it to work, but it won't! Eventually, Windows detects that the connection attempt failed and falls back to IPv4 connectivity. However, for users, connectivity to dual-stack websites can be delayed by 10-15 seconds. This obviously causes a problem for web browsers, but any network-connected app faces this issue.

As we looked into engineering a solution to this problem, we had to consider a couple of important issues. First, many enterprises deploy complex routing topologies. We had to make sure that our change did not break connectivity in these environments. Second, we needed a solution that worked not only for Internet Explorer but also all the other apps that are relying on Windows to help them connect to network resources. Those apps rely on us to remain intelligently connected throughout the IPv6 transition. Our solution needed to address the needs of existing desktop apps as well as new Metro-style apps.

Windows 8 tests IPv6 connectivity when you connect to a new network that advertises IPv6 routability, and it will only use IPv6 if IPv6 connectivity is actually functioning. This approach is a modification of our implementation of RFC 3484. Instead of sorting addresses as a result of policy, we use the actual state of the network as input to our algorithm. On a misconfigured network, this approach improves the experience not only for browsers but also for apps that connect to dual-stack destinations using standard Windows APIs.

Windows 8 performs the network connectivity test when you first connect to a new network; it caches this information and repeats the test every

30 days. The actual test for connectivity is a simple HTTP GET to an IPv6-only server that is hosted by Microsoft. (For standards buffs, this is implemented between rules 5 and 6 of destination address sorting in our implementation of RFC 3484.) Windows performs a similar network connectivity test for IPv4 connectivity. If both IPv4 and IPv6 are functioning, IPv6 will be preferred.

To make sure that Windows 8 does not cause problems on enterprise networks, the functionality has two safeguards:

- If the enterprise has provided specific routing information to a particular destination, then Windows 8 will honor that preference, regardless of the connectivity determined by Windows. In enterprise environments, Windows assumes that network administrators who configure such routes specifically thought it was a good idea to use those routes.
- This change isn't implemented on networks with web proxies. In these networks, the proxy provides connectivity to the Internet; so end-to-end testing of IPv6 connectivity is not useful. Instead, Windows 8 simply opens connections to the proxy in the most efficient manner possible.

In this way, we've ensured that apps and experiences on Windows 8 can remain reliably and speedily connected to the Internet throughout the IPv6 transition, even if your local network is misconfigured.

Ready for the future of IPv6-only networks

On an IPv6-only network, the best way to improve a user's experience is to increase the number of services and experiences that are available over IPv6. On such a network, access to the IPv4 Internet is through a NAT64. These devices can be a fragile point of failure for connectivity, and can have severe performance limitations that lead to dropped packets. They also break IPv4 peer-to-peer connectivity, needed for some multiplayer games.

Across Microsoft, we have done a lot of work to enable the growth of IPv6 deployments, both in enterprise and Internet settings. One of our most important efforts is to ensure that our server products support IPv6. IPv6 support is part of our Common Engineering Criteria (CEC). This is part of a broad company-wide commitment to customers that our business products, such as Exchange Server and SharePoint, support IPv6 in either dual-stack or IPv6-only configurations. Most Microsoft products built since 2007 have supported IPv6, but you can find out about [IPv6 support in other Microsoft products](#) on Technet. Through this effort, developers and solution providers can support IPv6 in their own products.



Microsoft is also working on IPv6 support for our own services. Earlier this year, the Internet Society announced the [World IPv6 Launch](#), a major milestone in the process of upgrading the Internet to IPv6. In June, Bing and other websites will start serving traffic over IPv6 on a permanent basis. Hardware vendors are working on IPv6 support in home routing devices, and many ISPs will start large-scale deployments of IPv6. CDNs (content delivery networks) have also started enabling support for IPv6 within their networks.

With the release of Windows 8, some of our infrastructure services will deploy IPv6 support.

Windows Update is a critical service providing ongoing support and updates to millions of users every day. More and more PCs are going to be connected to mobile broadband networks, where IPv6-only is a popular configuration. We have to make sure that downloads are reliably available to you on those networks.

For this reason the Windows Update service now supports both IPv6 and IPv4. Windows Update utilizes CDNs for worldwide distribution of updates and we are partnering with them to enable IPv6 support. Windows 8 will use IPv6, if available, to download Windows Updates so that users always get the best possible connectivity when downloading updates.

We are working with CDNs to extend IPv6 support beyond Windows 8. Once that work is complete, even Windows 7 and Windows Vista will automatically use IPv6, where it is available, for connecting to Windows Update.

Leading the way

Windows 8 is connected and ready to use, and our support of IPv6 is a key part of ensuring that connectivity for years to come. Because IPv4 wasn't designed to handle the scale of connectivity today, the Internet is undergoing a radical change in its foundation. Every connection to every website, every multiplayer game, and every video call will gradually move to IPv6.

As part of that transition, Microsoft is leading the way by ensuring that Windows 8 provides the most resilient connectivity to the Internet while

providing IPv6-ready products and services.

- Chris

***Note:** Several sections of this blog post were missing from the original publication. The missing sections were added several hours later. Apologies for the error.*

Building a rich and extensible media platform

Steven Sinofsky | [2012-06-08T08:00:00+00:00](#)

*Windows provides a broad set of technologies for consumers to experience video and audio and for developers to tap into these technologies through rich APIs. This post goes into depth on both of these aspects of the Windows media platform, which has been substantially improved for both desktop and Metro style apps. The landscape for media playback has changed significantly since Windows 7 was released, with an increased focus on streaming, and the desire for content owners to offer playback of their content on a broader array of devices, all while significantly reducing the battery power required for playback. With these new capabilities, which are part of both Windows 8 and Windows RT of course, we worked to provide industry-leading support for consumers and developers. **This post was authored by Scott Manchester, group program manager for our Media Platform and Technologies team.** –Steven*

Engaging with rich media—whether watching a movie, video chatting, or playing music—is one of the most prevalent and enjoyable things we do on our PCs today. I'd like to talk a little bit about the work we've done in Windows 8 to make a rich variety of multimedia activities possible, and to extend those capabilities to third party developers through an extensible media platform.

We had three goals in mind when designing the Windows 8 media platform:

1. **Maximize performance.** We wanted media playback to be fast and responsive, enabling the full power of the hardware while maximizing battery life on each PC.
2. **Simplify development and extensibility.** We wanted to provide a platform that could be easily extended and tailored for a given application, setting the stage for innovative custom media apps on Windows.
3. **Enable a breadth of scenarios.** A high performance, high efficiency, extensible platform can then enable a wide range of music, video, communications, and other multimedia apps.

With these three goals in mind, we set out to reimagine the media experience on the Windows platform.

Faster, more responsive media experiences

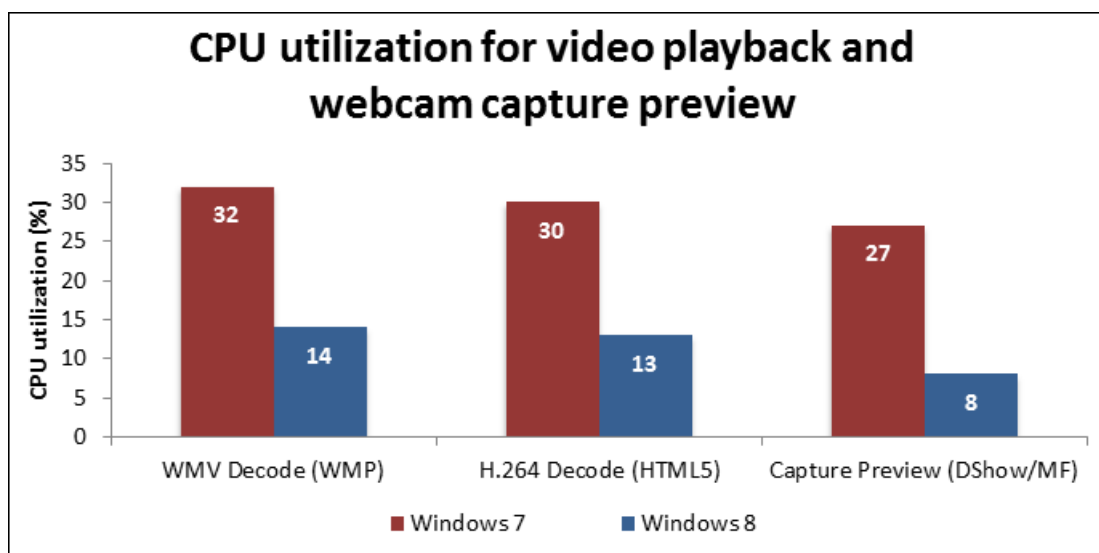
Performance is a key aspect of any user experience, but it is especially critical in multimedia scenarios. Videos need to play in real time, voice communication needs to feel instantaneous, and all of these tasks need to minimize the drain on your battery.

We measure performance by the time, computing resources, and memory that a given task takes on a system. We aimed to minimize all of those metrics. Our goals for media performance were focused on audio and video playback, transcoding, encoding, and capture.

Efficient video decoding

To get better battery life or just reduce power consumption for all media scenarios, we continue to work with partners in the silicon chip industry to enable new and faster experiences. With Windows 8 running on a Windows 8 certified PC, video decoding for common media formats will be offloaded to a dedicated hardware subsystem for media. This allows us to significantly lower CPU usage, resulting in smoother video playback and a longer battery life, as the dedicated media hardware is much more efficient than the CPU at media decoding. This improves all scenarios that require video decoding, including playback, transcoding, encoding, and capture scenarios.

The figure below shows a comparison of the average CPU utilization between Windows 7 and Windows 8 during playback of 720p VC1/H.264 video clips and webcam capture preview.



In addition to video offload, the improvements to webcam capture are made possible by the move from a DirectShow Capture API to the new, far more optimized Windows 8 Media Foundation Capture API. We've also improved software encoders for H.264 and VC-1 content so that encoding using the CPU (when it makes sense) is both fast and power-efficient.

Maximizing battery life during audio playback

Another example of the media performance improvements we've made in Windows 8 is in maximizing battery life (or just reducing power consumption) during audio playback. In addition to enabling offload of the audio pipeline (similar to the offload of video described above), we've radically improved the audio playback pipeline to be more efficient during steady-state playback. By batching up large chunks of audio data and doing all the processing for that chunk at one time, the CPU can stay asleep for over 100 times longer (over 1 second vs. 10ms), which can result in dramatically increased battery life during audio playback.

Of course, this approach isn't perfect for all scenarios since the increased buffering introduces additional delay. In the communications section below, we'll talk more about these tradeoffs and how the media stack adapts to optimize for each scenario

Audio and video offloading are just a couple of examples of the ways we've optimized the media stack in Windows 8 to provide lower CPU utilization, lower memory utilization, and better battery life for Desktop and Metro style apps.

Supporting a rich set of media scenarios

Performance is a critical aspect of the platform, but it is only as important as the features that shine because of it. In Windows 8, those features include support for modern video formats, low-latency communication streams, and a seamless connection to external media devices.

Platform tradeoffs

One of the challenges in developing a single media platform that serves different scenarios is that the platform has competing goals. For example, communication scenarios require low-latency, and audio/video encoding and playback, whose quality and performance benefit from buffering, which results in higher latency. In the next several sections, we'll touch on these challenges in the context of some of the scenarios we've worked to enable in Windows 8, including:

- Communications (e.g. Skype, Lync, etc.)
- Video playback and modern format support
- Auto-orientation of video
- Playback of premium content
- Seamless audio transitions
- Bringing the media experience to additional screens
- Emerging media capabilities

Simplifying development and extensibility

One common theme across these experiences is the extensibility that we've incorporated into the multimedia platform. Because users have a wide range of use cases, media formats, codecs, protection mechanisms, and processing, we provided our developers with the ability to customize and tailor their offerings to create great apps and websites on Windows.

As we discuss some of the media scenarios in the next several sections, we'll also cover some of the work we've done to make those scenarios extensible by developers and third-party partners. Let's dive deeper into the scenarios we've targeted for Windows 8.

Communications

Real-time communication on PCs, especially on mobile devices, has seen a huge growth over the last decade. Windows users are using services like Skype and Lync to make several billion minutes of voice and video calls per day. [TeleGeography](#) estimates that international Skype-to-Skype calls (including video calls) grew 48 percent in 2011, to 145 billion minutes. We've made a significant investment in improving the experience of video and audio calling on all Windows 8 PCs. To achieve this goal, we focused our efforts in two areas:

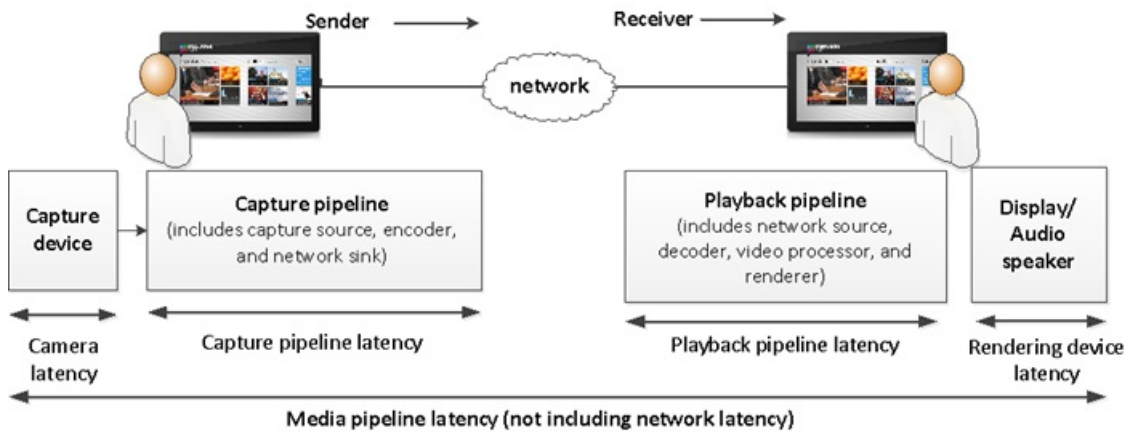
- **Enable built-in low-latency media capture and rendering.** Low latency is essential for communications apps, so Windows supports low-latency media capture and playback into the OS.
- **Support HD cameras to enhance video communication experience.** High-definition videos make your communication experience more real and enjoyable, so Windows supports HD camera devices.

Enabling low latency

When you communicate with another person, you expect near-instant responses. For this reason, communications systems generally try to minimize the end-to-end delay (also referred to as latency). In designing audio and video systems for playback, buffering is often used as both a protection against glitches caused by processing spikes or network traffic, and to reduce power consumption. However, this buffering introduces a delay into the audio and video, which is perceived as latency by the audience. In engineering Windows 8, we designed the media platform to support both playback-optimized and communication-optimized scenarios. The media infrastructure can switch between a playback mode (high

buffering, more tolerant of varying conditions) and a communications-optimized mode (low delay).

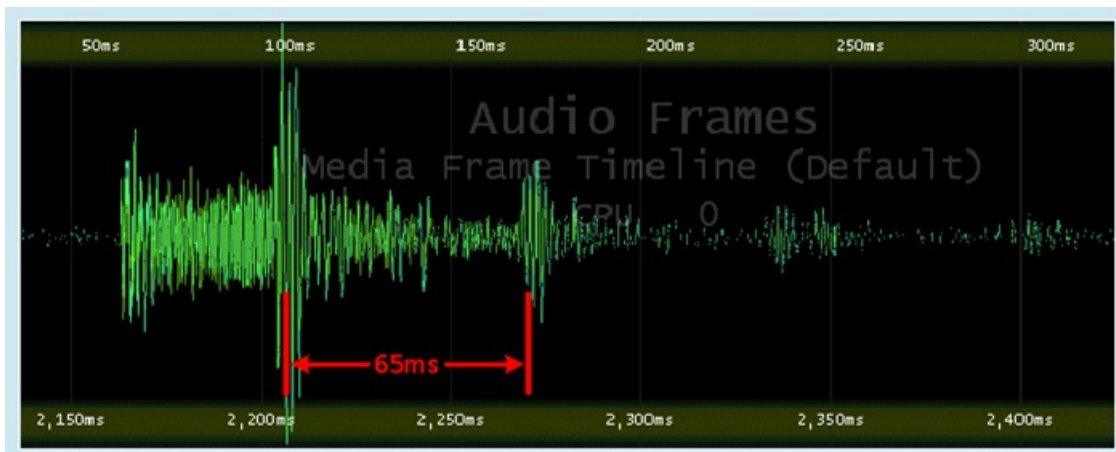
According to the [TIA/EIA 920 standard](#), the one-way audio latency that can be attributed to just the *media processing pipeline* cannot exceed 100ms in order to achieve a usable real-time communication experience. With this metric in mind, we designed a test environment to measure the end-to-end latency of the pipeline, shown in the following diagram:



There are many components to optimize to get low latency

In the case of video communication, the end-to-end or “glass-to-glass” pipeline latency is measured as the delay it takes for a video frame to be captured by the camera device and then encoded to a supported video format, streamed over the network loopback interfaces, decoded, and finally rendered by the display.

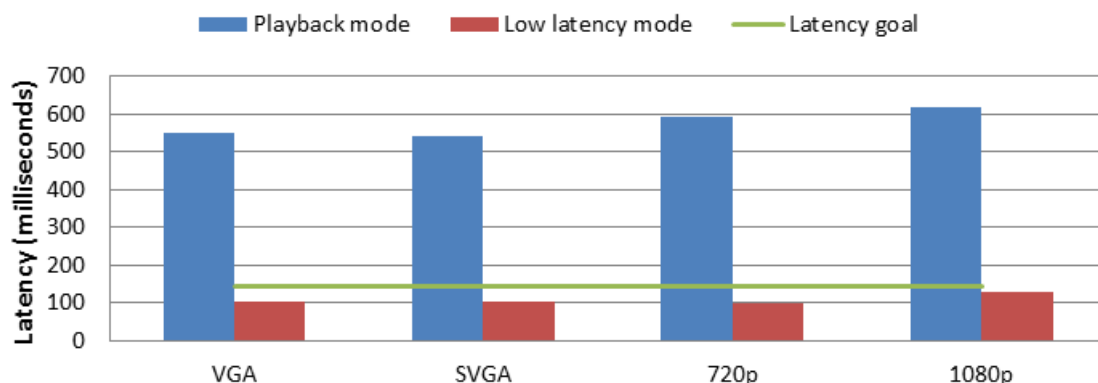
Looking at the figure below, you can see the result obtained for capturing and rendering PCM audio when the media pipeline is in low latency mode. The first set of spikes corresponds to the original spoken words at the transmitter and the second set shows those words at the receiver. The delay between the two is 65ms, well below the 100ms goal.



End-to-end pipeline latency of PCM audio: Low latency mode

The next chart shows a comparison of the pipeline latency of playback and communication-optimized mode when a video frame is captured, encoded (in H.264 format), streamed, decoded, and then displayed at various resolutions. The goal of 145ms overall latency (as deemed by TIA/EIA 920 for usable real-time video calling) is shown by the green line on the chart.

Glass-to-glass video latency



Video frames are captured at a rate of 30 frames per second and encoded into H.264

In playback mode, the average latency of the pipeline is about 575ms. This delay is necessary for a smooth playback experience when consuming video, but unacceptable for real-time video communication. In low latency mode, on the other hand, the measured latency is well under the target goal at each of the measured video resolutions.

Supporting HD video calling

Another example of the work we have done to improve communication on Windows 8 PCs is through OS support for HD cameras. New class drivers will work transparently with applications to provide support for HD video features. In addition, all of the hardware acceleration for video decoding discussed previously will be utilized for communication scenarios.

Windows 8 will offer a consistent, high-quality, hardware-accelerated, power efficient media communication experience on PCs designed for Windows 8. We have made significant investments in the media platform to improve pipeline latency, and with added support for H.264 cameras, users will be able to communicate with friends and family in high-fidelity HD video.

Video and audio support for Metro style apps

Our main goal for native media format support for Metro style apps was to ensure users and app developers could count on a consistently great playback experience across a wide variety of PC form factors, with modern formats used in mainstream scenarios such as:

- HTML5-based entertainment on the web
- Home movies captured using popular smartphones, point-and-shoot cameras, or AVC-HD cameras
- Streaming music, movies, and TV shows from popular services

The tables below show the video and audio formats that have built-in support for Metro style apps. Formats recommended for use by Metro style apps are a reflection of deep partnerships with hardware manufacturers for predictable hardware acceleration across PC form factors and predictable end-to-end scenario performance beyond playback such as capture, streaming, and transcoding.

Media type	File & stream formats	Codecs or components	Codec supported on Windows RT	Hardware-accelerated on Windows 8 certified PCs
Video	MPEG-4 **	H.264 **	Yes	Yes
	ASF **	H.263	Yes	Hardware-dependent
	MPEG-2 PS	Motion JPEG	Yes	Hardware-dependent
	MPEG-2 TS	MPEG-1 *	No	Yes
	3GPP	MPEG-2 *	No	Yes
	3GPP2	MPEG-4 (Part 2)	Yes	Hardware-dependent
	AVI***	VC-1 **	Yes	Yes
		WMV 9	Yes	Yes
		WMV 7, 8	No	No
		DV	No	No
	Raw (NV12, YUY2, RGB32)	No	N/A	
Audio	MPEG-4 **	AAC **	Yes	Hardware-dependent
	MP3 **	HE-AAC **	Yes	Hardware-dependent
	ASF **	Dolby Digital (non-disc)	Yes	Hardware-dependent
	AAC LATM	Dolby Digital Plus (non-disc)	Yes	Hardware-dependent
	AAC LOAS	MP3 **	Yes	Hardware-dependent
	ADTS	WMA ** Standard, Pro, Lossless	Yes	Hardware-dependent
	WAV	MPEG-1 Layer I, Layer II	Yes	Hardware-dependent
		MPEG-2	Yes	Hardware-dependent
		ULAW	Yes	Hardware-dependent
		PCM	Yes	Hardware-dependent
		ADPCM	Yes	Hardware-dependent
* Requires Windows 8 Media Center Pack or Windows 8 Pro Pack				
** Recommended for use by Metro style apps				
*** AVI files will play only when audio and video codecs are supported by Windows RT.				

Windows 8 has excellent support for MPEG-4, most typically comprised of H.264 video and AAC audio. Several popular codecs, including Divx and Xvid, implement the MPEG-4 Part 2 standard, so many of these files play great in Metro style apps. The same is true for modern MOV files,

which are based on the MPEG-4 Part 12 standard, such as videos captured on iOS devices. Fragmented MPEG-4 and 2K/4K resolutions are now possible. We have previously talked about MPEG-2 and DVD playback, which is available in [Windows 8 Media Center](#).

During the development of Windows 7 we talked quite a bit about CODEC support natively in Windows and the formats available through extensibility. Since then, the environment around CODECs has consistently moved towards a smaller set of well-defined and broadly-supported formats, particularly h.264 for video. Due to factors such as intellectual property and hardware support, this makes a great deal of sense. Even [browsers are making this transition](#) with HTML5. But we also recognize that some individuals have preferred formats for a variety of reasons, and we wanted to make sure Windows 8 app developers could choose to use the formats they prefer. Formats popular among the enthusiast community or with specific developers such as FLAC, MKV, and OGG, can have their own CODECs packaged as part of a Metro style app, since the Windows 8 media platform is highly extensible.

Auto-orientation of video

With the proliferation of video recording in traditional cameras, smartphones, and tablets, users can capture video while holding their device in either portrait or landscape mode – there is no “right-side-up” any longer, thanks to modern touch-based interfaces. Many of us have experienced the frustration of recording a video and realizing the camera was sideways or upside down only after viewing it on the PC. Since the video scan pattern is fixed, videos may not be oriented properly when viewed.

To overcome this problem, cameras are beginning to author orientation metadata in mainstream file formats such as MP4 and ASF when saving recorded video to storage.



To ensure a terrific viewing experience of personal videos from Windows PCs, we’ve made the following improvements to address this problem:

- Orientation metadata is now supported in MP4 and ASF (VC-1, WMV) videos.
- Videos with orientation metadata are auto-rotated during playback.
- The thumbnail for a video with orientation metadata is auto-rotated.
- Metro style apps with video capture capabilities can easily read and author orientation metadata.

Premium content

Another area where we’ve invested heavily for Windows 8 is in allowing seamless playback of premium content. Although most of the video content consumed initially on the Internet was user generated, much of the growth in the Internet video space can now be attributed to “premium content,” which includes online movie purchases through on-demand streaming video, as well as the ad-supported TV offerings. [According to IHS Screen Digest](#), 3.4 billion paid movies will be streamed online in the US in 2012—over double the number watched in 2011, and over a billion more movies than were consumed via DVD and Blu-Ray combined.

Premium video content has many of the same requirements as any other video content, but it also requires two substantial platform features in order to deliver the best experience: adaptive bitrate streaming and content protection.

Adaptive bitrate streaming

Adaptive bitrate streaming provides a smoother, more responsive video playback experience by enabling the PC to adapt to the most appropriate bitrate under varying networking and resource utilization conditions. As a result, startup and seek times can be significantly improved because the first few frames can be delivered at a lower bitrate to reduce buffering time and increase responsiveness. If network or device conditions change, the PC can negotiate a lower or higher bitrate to minimize buffering or increase video quality.

Through the extensibility of the Media Foundation Platform in Windows 8, apps can have custom media sources and adaptive bitrate media sources to support new formats. Custom media sources and streaming protocols can also take advantage of hardware offload and content protection.

The [Windows Azure Media Services](#) team is using our extensibility model to build the [Smooth Streaming Client SDK for Metro style apps](#). Smooth Streaming is Microsoft’s initiative to deliver high quality multi-bitrate content and enable Video-on-demand, Live, Linear TV, and Download-and-Play.

Content protection

Most premium Internet video content services choose to apply content protection, which is often a requirement from the content owners (e.g.

movie studios or TV networks). To enable the playback of protected content in Metro style apps, Microsoft is making available the [PlayReady Client SDK](#) for premium content services. PlayReady supports download as well as streaming, and the above-mentioned IIS Smooth Streaming Client SDK integrates seamlessly with the PlayReady Client SDK to allow services to easily build protected streaming experiences.

We recognize that there are other content protection technologies being used today in the industry. Just like with adaptive streaming, the Media Foundation extensibility model allows for third parties to integrate their custom content protection systems with built-in hardware-accelerated video decoding. If a service needs to use a custom streaming format or content protection system, it can integrate its own technology without having to compromise on decoding quality or battery runtime.

In summary, Windows 8 will enable a wider offering of premium content services for customers to choose from and enjoy on their Windows 8 devices, providing a great streaming and downloaded experience as well as great battery life when watching premium HD video content.

Seamless audio transitions

As Windows 8 enables a multitude of media scenarios, we wanted to make sure that transitioning between these scenarios was as seamless and fluid as possible. Users often run into overlapping audio-based activities – for example, while listening to a music streaming service, they attempt to watch a video clip. We wanted to provide a clean, uncluttered audio experience that would make it easier and simpler for you to listen to the content you want, when you want it.

In Windows 8, instead of mixing all audio content and sending the resulting (often incoherent) stream to the speakers, Windows can pause a stream when a second stream is played and when it makes sense to do so. In most cases, Windows prioritizes audio coming from the app that is in the foreground. When you move the app to the background, the system quiets the stream. An example is a game app where you likely don't want to listen to game audio when you've switched away from the game. However, there are cases where this is not the desired behavior – for example, if you're listening to music in the background while checking email or surfing the web. To enable these scenarios and to allow you to hear background audio when it makes sense, we've introduced stream types that reflect the type of audio being played.

Below is a list of different stream types, along with an example of the type of content expected for each stream.

Audio category	Example streams	Background capable?
Background capable media	Local and streaming audio playlists	Yes
Foreground only media	Movies, games	No
Communications	Skype, Voice-over-IP, live chatting	Yes
Alerts	Alarms, ringing notifications	No
Game media	Background music played by a game	No
Game effects	Gun shots, explosions, characters talking, all non-music sounds	No
Sound effects	Button confirmation sounds, beeps, dings	No
Other	Default audio type, and recommended for all audio media that does not need to continue playing in the background.	No

Bringing the media experience to additional screens

In Windows 7, we announced Play To, which you can use to stream media files to supported external devices from Windows Explorer and Windows Media player. In Windows 8, Play To makes it even easier and simpler to share personal media collections and HTML5 media with Play-To-enabled devices at home. Our focus for Play To was to create rich social experiences built around personal content – like sharing photos with family and friends, streaming music for a party, or watching user-generated videos from the Internet. The experience has been designed from the ground up to integrate tightly with HTML5 from existing websites and your personal media collections, whether they're stored in the local

library of a Windows PC or tablet, on another home PC or network-attached media server, or on a web server in the cloud.

Play To is now easier to discover and will deliver a consistent, high quality experience from a multitude of Metro style apps. A few of the improved user experiences include:

- **Improved setup:** On home networks (or HomeGroup) where you've allowed sharing, Play To devices are automatically discovered and installed on your PC.
- **Improved device experience:** Metro style apps work only with Windows certified Play To receivers. These devices are validated to support modern media formats, are DLNA standards-compliant, and have great performance (including the updated Xbox 360 available later this year). The desktop experience first introduced in Windows 7 has been added to the Explorer Ribbon and will continue to support all DLNA DMR devices.
- **Easier discovery:** Play To is accessible from the Devices charm, making it easy to initiate from any app that supports Play To. Just swipe in from the right edge (or point your mouse to the top-right corner), select the Devices charm, and then select the device you want to stream to.
- **Integrated into Metro style IE:** IE allows you to stream HTML5 music, video, and photos from the web to your devices.
- **Works with the new Music, Video, and Photo apps:** Apps can stream photos from a variety of sources and personal music and video collections.



Play To from the Videos app

We have also focused heavily on making it easy for developers to use Play To in their apps and websites – the functionality is available to all Metro style apps via the Play To contract. The XBox 360 will support Play To in an update later this year.

Emerging media capabilities

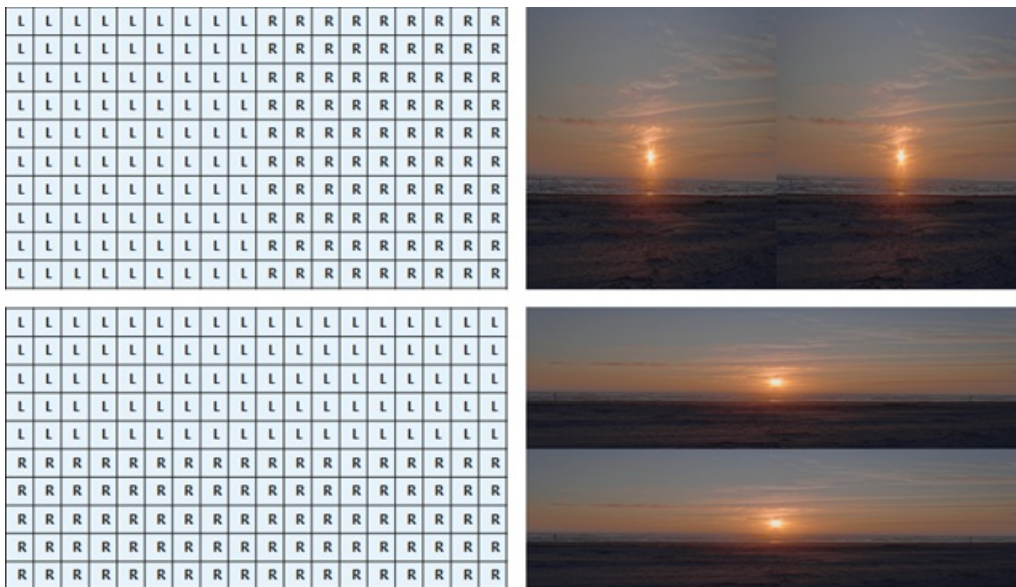
Windows is enabling support for new content types for consumption and increased flexibility for content creation and communication. Stereo 3D, accessibility, and DSP effects are three examples of how we are enabling great multimedia experiences on Windows 8

Experiencing stereo 3D video

Over the last few years, the Stereo 3D (S3D) market has evolved from hype to finished consumer products. S3D provides a 3D viewing experience by displaying two overlapping copies of a video (captured from different angles), which appear as a single 3D video when viewed with 3D glasses. Our goal is to enable a viable S3D ecosystem for Windows by enabling key gaming and video playback scenarios on a platform that abstracts away the specifics of the 3D technology from the end-user's PC.

In Windows 8, S3D support is available on DirectX 10 or higher GPUs with compatible drivers. A S3D-compatible display is needed to see S3D content. We wanted to make sure that Windows would support a wide range of display technologies with a consistent user experience, and make it easy for software and hardware to develop on our platform. As a result, specific S3D display technologies are largely made irrelevant by the graphics drivers, and a consistent set of APIs are available to apps using stereo 3D.

The Windows 8 media platform provides support for standards-compliant media formats for S3D video. H.264 video with frame-packing metadata represented as Supplemental Enhancement Information (SEI) is the typical format being adopted for online delivery, and is therefore the desirable S3D video format in Windows 8. The frame-packing formats that we support natively in the platform include both side-by-side and top-and-bottom arrangements, as in the illustration below.



Windows 8 supports a range of stereo 3D input formats, including side-by-side and top-bottom.

Delivering accessible media experiences in the web platform

Media accessibility is an important part of the Windows promise to our customers, especially for users with accessibility needs.

Subtitles provide interpretive or additional information to viewers who prefer a written transcript, those who need to see a translation in a different language, or those who need to see a transcript due to limited hearing ability.



Video playback in Windows 8 with subtitles

The web community has worked together through W3C to specify the best ways to deliver the subtitling experience through all modern web platforms. These include the following:

- The <track> element can carry subtitle and closed captions for the HTML5 video tag. This feature is now incorporated into Windows 8. Subtitle support is now available through the video tag in IE10 and in apps using HTML5.
- User controls are available on the default media controls of the video tag.
- There is native support for the WebVTT and SMPTE-TT formats that are commonly found in the web community and with partners in the TV and broadcasting industries.
- The Windows 8 media platform provides support for multiple audio tracks within a media source. Users can switch audio tracks to their preferred language, and tracks can also be used for audio descriptions for sight-impaired users. Metro style apps can now easily switch between audio tracks or even play multiple audio tracks simultaneously, for instance, a normal audio track plus an audio description.



Video playback in Windows 8 with multiple audio tracks

Adding effects to the media pipeline

The Windows 8 media platform has been designed to adapt easily. One way that we've done this is by allowing effects (often referred to as digital signal processing, or DSP) to be added to the pipeline. We've included several built-in effects, like image stabilization and horizontal flipping (which is useful for webcam preview), and we've also made it easy for applications to plug in to the Media Foundation pipeline with custom effects. In addition, we've made sure that media data can pass through the pipeline efficiently, thus minimizing the performance and power impact of adding DSPs.

Summary

The Windows 8 media platform is designed to deliver a fluid and responsive media experience with great battery life. We've engineered Windows to give you a great user experience across a broad set of scenarios, including voice communication, audio and video playback, and streaming content. As media applications continue to evolve, the media platform in Windows will enable these experiences to shine across all Windows 8 PCs.

I'll close now with a video that walks you through some of the highlights of the new media platform.

--Scott

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

Activating Windows 8 contracts in your app

Steven Sinofsky | [2012-06-11T12:00:00+00:00](#)

One topic that we've demonstrated quite a bit is how apps on a Windows 8 PC can communicate with other apps and web services. At the start of Windows 8 we chose an approach where apps can be the source or destination for data you want to share—sort of like a clipboard, but with a richer interaction model and clearer semantics. When an app implements a [contract](#), Windows 8 can provide glue between that app and any other apps on the system, and the system itself. You can see this in action when you do something simple like use the Share charm from a web page in the Metro style Internet Explorer—you can share the link via the Mail app, with someone whose contact info you've stored in the People app, and so on. You can search across apps that implement the Search contract. You can open and save files from or to any location that implements the File Open and Save Picker contracts. This innovative approach allows Windows 8 to work with any app/service pair rather than “hardcoding” a single level of support for a given app. And all of this is supported, if you choose, by your Microsoft account, which you can connect to different services, from Facebook to Twitter to LinkedIn and more. Over the course of this week, we'll do a series of posts on the new Microsoft apps, where sharing, connecting, and integration with Windows 8 are key topics. This is a repost of a developer-focused post from our [Windows 8 App Developer blog](#) and was authored by Derek Gebhard, a program manager on our User Experience team. --Steven

When you start writing Metro style apps you'll quickly come across contracts, a new and powerful concept in Windows 8. Metro style apps use contracts to declare interactions they support with other apps and with Windows. You've probably already heard about some of them: search, share, etc. Using contracts, apps become better by working with the system or with each other when users install more apps that implement contracts. In this post I'll walk you through activation, one of the main concepts to think about as you add contracts to your apps.

The Windows activation platform is used to launch Metro Style apps and to notify them of the reason why a user launched them. The reasons vary from a user starting the app using its tile on the start screen to the app being launched for a specific task such as showing a user search results for a query. Windows provides your app with the reason it was launched and if applicable any additional info needed to complete its task. Before our Windows 8 activation platform, you passed this info to apps via command-line parameters. With our new model, we also support passing live objects such as a StorageFile, ShareOperation, etc to provide the app with context. You'll see that this makes contracts all the more powerful. Let's jump into the details of what you need to know to support being launched for a contract.

Contracts: Launching Metro style apps for a purpose and with context

As you can see in the [Windows 8 Consumer Preview demo](#), Windows 8 contracts are the glue that binds your app to other Metro style apps and to the system UI. For example, the File Open Picker contract allows the user to import files from one app into another. With the Search contract, users are empowered to search an app from anywhere in the system and can quickly transfer a query between multiple apps. In all of these cases, and a lot of other contract scenarios, Windows needs to be able to launch directly to a spot in your app's UI where the user can complete a specific task quickly and efficiently. This is where our activation platform and API come into play.

Users initiate app interactions in one of two ways:

1. Through an action that requires the app's fully immersive view to be in the foreground. This is also called *main view activation*. An example is the Search contract.



Example of main view activation

2. Through an action that is hosted inline, without leaving the context of the currently running app. This is also called *hosted view activation*. Here are two examples, an app participating in the file picker and an app being used as a Share target.



Example of hosted view activation in the Picker



Example of hosted view activation for Share targets

The differences between these two are:

Main view activation

Hosted view activation

Is fully immersive and launches as the main app on screen

Renders UI within system chrome

Can be used for potentially many different tasks

Is used for a short, directed task and code is focused solely on this task

Appears in the switch list

Never shows up in the switch list

Can be closed via the close gesture

Doesn't change the view of the main window for the same app

So let's look at these activation models and apply them to a couple of common scenarios that will help you build your great Metro style apps.

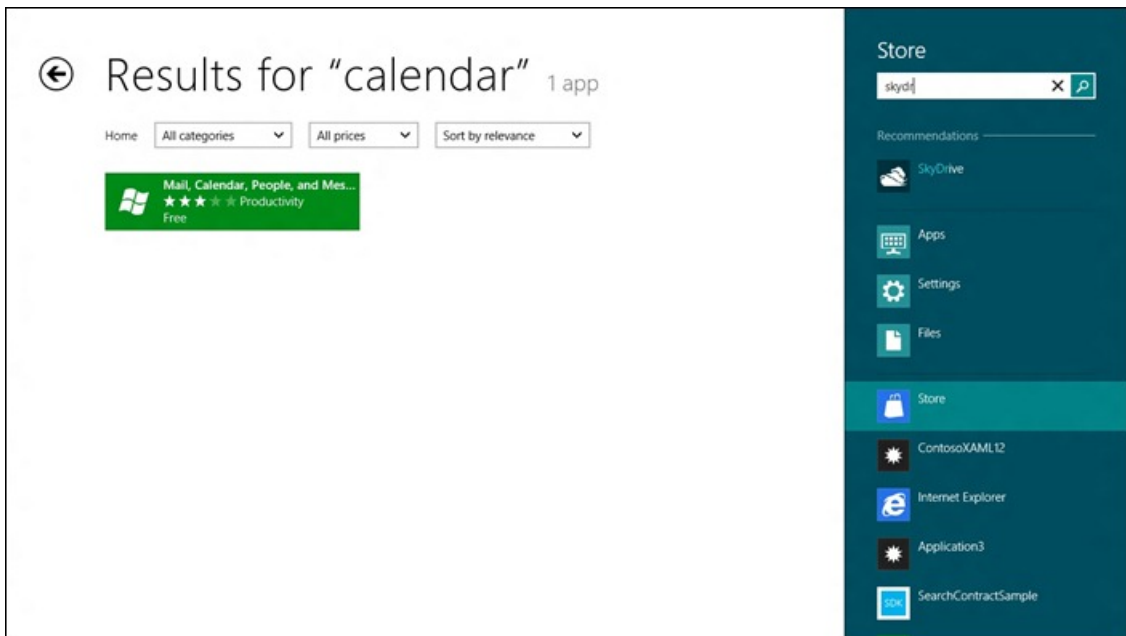
Scenario 1: Integrating Search activation in your app

In Windows 8, adding search through the Search contract lets users search your app's content from anywhere in their system at any time. If your app is the main app on screen, users can search its content immediately by using the Search charm. Otherwise, users can select the Search charm and then pick your app from the list of apps in the Search pane to search it.

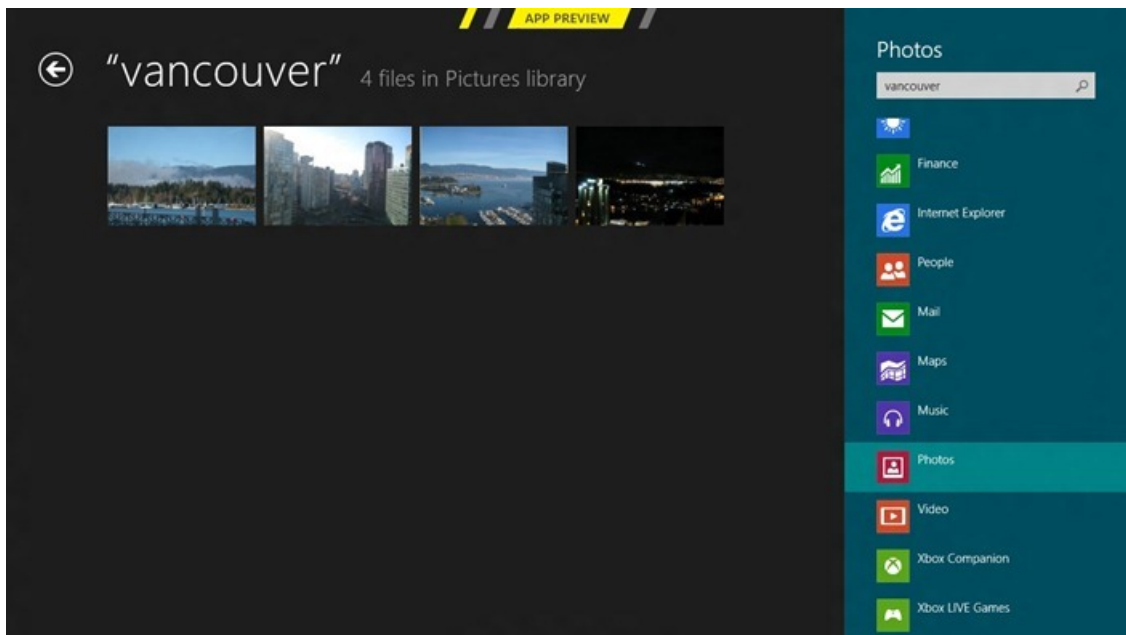
Supporting Search activation means that your app can be launched at any time to show search results for a specific query. Just like being launched from the start screen, being launched from the Search pane falls under main view activation. So, if you support multiple contracts, your app can potentially be activated for many different scenarios. In addition, your app could end up receiving this activation when it is already running, because a user may want to repurpose your main view to handle a specific scenario like showing search results. To make this work, I recommend that you:

- Delay loading of your code that isn't essential to the main view contract your app is activated to handle.
- Separate your general initialization logic that you use for all contracts from the logic that needs to be run for a specific contract.
- Ensure that any code expected to run only one time at launch isn't added into your activation handler in such a way that it can execute multiple times.
- Reload any previous state and settings when being launched from a terminated state so that your app appears to the user as always running and connected.

Check out the Store and Photos apps. They do a great job of following these recommendations when supporting Search activation.



Search in the Store app



Search in the Photos app

Let's take a look at how you can support Search activation properly in your JavaScript and XAML apps.

JavaScript apps

For JavaScript Metro style apps, activation is exposed through the [WinJS.Application.onactivated event](#). This event is fired after **DOMContentLoaded** completes if the app isn't already running or isn't suspended. Otherwise, the event is fired as soon as Windows needs to activate the app. Visual Studio tooling for JavaScript apps takes care of setting up this event registration in default.js and provides an area where you can add code that will run when a generic launch activation occurs, that is when the user launches your app from the start screen.

To extend support for Search activation in your app:

1. Add the Search declaration to your manifest using the Visual Studio Manifest Designer.
2. Place in your JavaScript's global scope any general initialization code that needs to run every time your app is started irrespective of the reason. If any of this code needs to access the DOM, add the code in a **DOMContentLoaded** event handler.
3. Register to handle being activated for Search.
4. When your app is activated for Search, navigate to your search results page and pass in the [queryText](#) you get from the activation event arguments.

If you are like me, you are probably looking for an easier way than doing this manually. Fortunately you can use Visual Studio tooling for completing most of this by right clicking your project, selecting Add > New Item, and choosing Search Contract in the dialog. Most of the code you see here, and a search UI that displays results in a way that follows our [search ux guidelines](#) is automatically created for you. But you must use the WinJS.Navigation framework with this tooling.

Here is a code snippet from my photo app's default.js file that shows support for Search activation:

```
// Register activated event handler
WinJS.Application.addEventListener("activated", function (eventObject) {
    ...
    if (eventObject.detail.kind === appModel.Activation.ActivationKind.launch) {
        ...
    } else if (eventObject.detail.kind === appModel.Activation.ActivationKind.search) {
        uri = searchPageURI;
        pageParameters = { queryText: eventObject.detail.queryText };
    }
    // Indicate to the system that the splash screen must not be torn down
    // until after processAll and navigate complete asynchronously.
    if (uri) {
        eventObject.setPromise(ui.processAll().then(function () {
            return nav.navigate(uri, pageParameters);
        }));
    }
});
```

XAML apps

For XAML Metro style apps, the [Windows.UI.Xaml.Application class](#) does a lot of the work needed for your app to support activation. This class exposes a set of strongly typed activation methods that you can override for supporting common contracts such as Search. For all contract activations that don't have a strongly typed method, you can override the [OnActivated method](#) and inspect the activation kind to determine the contract for which your app is activated.

New XAML app projects in Visual Studio come with generated code that uses the [Windows.UI.Xaml.Application class](#) to make the app capable of being activated for a generic launch. The code for handling this activation is in the class representation for your app, found in the App.xaml.cs/cpp/vb files.

To extend support for Search activation in your app:

1. Add the Search declaration to your manifest using the Visual Studio Manifest Designer.
2. Place in the App constructor of App.xaml.cs/cpp/vb any general initialization code that needs to run every time your application is started irrespective of the reason.
3. Override the strongly typed [OnSearchActivated method](#) in App.xaml.cs/cpp/vb to handle search activation.
4. Load your Search UI and show search results for the query you receive in the [SearchActivatedEventArgs](#).

Again, just like for JavaScript apps, there is an easier way than manually doing this work. You can use Visual Studio tooling for completing a lot of this work. Just right click on your project, select Add > New Item, and choose Search Contract in the dialog. Most of the code you see here, and a search UI that displays results in a way that follows our [Search UX guidelines](#) is automatically created for you.

Here are snippets of C# code from my photo app that shows support for Search activation.

We must override the [OnSearchActivated method](#) to support activation for Search:

```
protected override void OnSearchActivated(SearchActivatedEventArgs args)
{
    // Load Search UI
    PhotoApp.SearchResultsPage.Activate(args.QueryText);
}
```

The **Activate** method of the **SearchResultsPage** sets up a UI that shows search results for the user's search query:

```
// SearchResultsPage.xaml.cs code snippet
public static void Activate(String queryText)
{
    // If the window isn't already using Frame navigation, insert our own frame
    var previousContent = Window.Current.Content;
    var frame = previousContent as Frame;
    if (frame == null)
    {
        frame = new Frame();
        Window.Current.Content = frame;
    }
    // Use navigation to display the results, packing both the query text and the previous
    // Window content into a single parameter object
    frame.Navigate(typeof(SearchResultsPage1),
```

```

    new Tuple<String, UIElement>(queryText, previousContent));
    // The window must be activated in 15 seconds
    Window.Current.Activate();
}

```

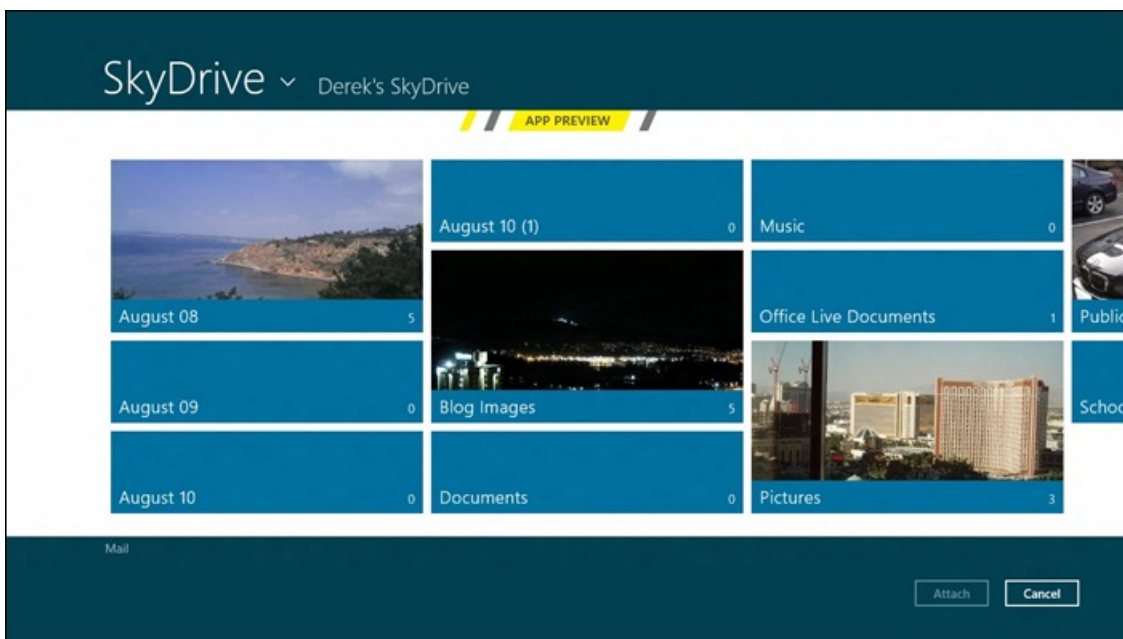
The logic and principles showcased here don't just apply to adding Search activation support. You can use the same techniques when adding support for [Protocols](#), [File Associations](#), and [Device AutoPlay](#) as these are also main view activation contracts.

Scenario 2: Integrating File Open Picker activation in your app

A Metro style app can call the file picker to let the user browse their system and pick files or folders for the app to operate on or to let the user save a file using a new name, file type, or location ("Save As"). Apps can also use the file picker as an interface to provide other apps with files, a save location, or even file updates. By incorporating the File Open Picker contract, you can help users pick files from your app directly within another app. Users gain freedom and flexibility to choose files that your app stores and presents.

Launching an app for the File Open Picker contract falls under hosted view activation. The app's UI is hosted inside of the file picker and the code that runs for this activation must be solely focused on the task of enabling users to pick their files. It is important that your app is as fast as possible here to give users a great experience. Don't load any code or libraries that are unnecessary for the specific hosted view activation task.

I recommend looking at the SkyDrive app because it is a great example of supporting File Open Picker activation and focusing solely on the task of allowing users to pick files.



File Open Picker support in the SkyDrive app

Let's take a look at how you can support File Open Picker activation properly in your JavaScript and XAML apps.

JavaScript apps

For JavaScript Metro style apps, the hosted view activation behaves the same as main view activation, except for one key difference: hosted view activation always occurs in a new window and script context. This means that your code for handling this activation can't access libraries, global variables, or the DOM of your main app.

To extending your app to support File Open Picker activation:

1. Create a new HTML page that is specifically designed to handle only the File Open Picker contract.
2. Add the File Open Picker declaration in the Visual Studio manifest designer and specify the newly created HTML page as the start page.
3. Load only JavaScript and other resources in this page that are necessary for supporting the File Open Picker contract to improve performance.
4. Structure the activation event handler to handle only activation for the File Open Picker contract. This handler is called only once during the lifetime of the file picking task.
5. Use the activation event arguments to interact with the file picker.

To save time you can use Visual Studio tooling for completing this work. Just right click on your project, select Add > New Item, and choose File Picker Contract in the dialog. Most of what you see next is automatically created for you in your project.

Here is a code snippet from my photo app's fileOpenPicker.js file for handling File Open Picker activation:

```

// Register activated event handler for handling File Open Picker activation
WinJS.Application.addEventListener("activated", function (eventObject) {
    if (eventObject.detail.kind === Windows.ApplicationModel.Activation.ActivationKind.fileOpenPicker) {
        pickerUI = eventObject.detail.fileOpenPickerUI;
        pickerUI.onfileremoved = fileRemovedFromPickerUI;
        ...
    }
});

WinJS.Application.start();

```

XAML apps

For XAML Metro style apps, you support hosted view activation in your app similarly to main view activation. The biggest difference is that now your app must create a new thread and new window to handle the activation. The Visual Studio template code handles all of the work to create the new thread and new window on your behalf for hosted view activations.

To handle File Open Picker activation a XAML app must:

1. Add the File Open Picker declaration to your manifest using the Visual Studio Manifest Designer.
2. Override the [OnFileOpenPickerActivated method](#) in App.Xaml.cs/cpp/vb and load your page that will handle this contract.
3. Pass in the [FileOpenPickerActivatedEventArgs](#) to the page handling this contract so that it can interact with the file picker.

To save time you can use Visual Studio tooling for completing this work. Just right click your project, select Add > New Item, and choose File Picker Contract in the dialog. Most of what you see next is automatically created for you in your project.

Here is a snippet of C# code from my photo app for handling File Open Picker activation:

```

// App.xaml.cs code snippet
protected override void OnFileOpenPickerActivated(FileOpenPickerActivatedEventArgs args)
{
    var fileOpenPickerPage = new PhotoApp.FileOpenPickerPage();
    fileOpenPickerPage.Activate(args);
}

// FileOpenPickerPage.xaml.cs code snippet
public void Activate(FileOpenPickerActivatedEventArgs args)
{
    this._fileOpenPickerUI = args.FileOpenPickerUI;
    this._fileOpenPickerUI.FileRemoved += FileOpenPickerUI_FileRemoved;

    // Show the user's photos in the Picker UI
    ...

    Window.Current.Content = this;
    // The window must be activated in 15 seconds
    Window.Current.Activate();
}

```

The logic and principles showcased here don't just apply to adding File Open Picker activation support. You can use the same techniques when adding support for [Share Target](#), [File Save Picker](#), [Contact Picker](#), [Camera Settings](#), and [Print Task Settings](#) as these are also hosted view activation contracts.

In closing

I showed you how Search, File Picker, and other Windows 8 contracts offer the ability to drive users to your app for completing a specific task from other parts of the system and even other apps in certain scenarios. Users will expect these experiences in your app are fast and fluid because Windows and your app are both aware of their intent and the task they are trying to complete. Implementing your app activation correctly is core to creating a great experience for these contracts. Even if you are just working on the core of an app and are not using any contracts, it is good to keep these tips in mind as you set up your generic launch activation. This way you can easily extend your app in the future to support contracts without refactoring your code.

Things to remember:

1. Place any general app initialization logic in a location where it will be executed independent of how your app is activated.
2. Your activation handlers can be executed even when your app is already running or is suspended. Make sure this can't cause any unintended consequences for your app.
3. Visual Studio tooling can do a lot of the work for you to support the Search, Share Target, and File Open Picker contracts. All you need to do is right click your project and select Add > New Item.
4. When receiving a hosted view activation, load only the code necessary for the task associated with the activation.

To learn more about activation and contracts in Windows 8, you can follow these links or ask questions in our [forums](#):

Documentation

- [JavaScript activation](#)
- [XAML activation](#)
- [Search contract](#)
- [File Open Picker contract](#)
- [Share Target contract](#)
- [File Save Picker contract](#)
- [Contact Picker contract](#)
- [Camera Settings contract](#)
- [Print Task Settings contract](#)
- [Cached File Updater contract](#)
- [Protocol contract](#)
- [File Association contract](#)
- [Device AutoPlay contract](#)

Samples

- [Activation](#)
- [Search contract](#)
- [File Open Picker contract](#)
- [Share Target contract](#)
- [File Save Picker contract](#)
- [Contact Picker contract](#)
- [Print Task Settings contract](#)
- [Protocol contract](#)
- [File Association contract](#)

Thanks,

Derek Gebhard
Program Manager, Windows User Experience

Contributions by: Jake Sabulsky, Marco Matos, Daniel Oliver

The People app: the complete, cloud-powered address book for Windows 8

Steven Sinofsky | [2012-06-13T10:15:00+00:00](#)

Managing "contacts" has been a bit of a challenge for many, especially as the number of places that contacts can be stored and the number of PCs and devices we use to access those contacts has increased. Storing contacts in the cloud for easy roaming and connectivity is a part of the solution. With Windows 8 and the new People app, we are taking cloud storage a step further by optionally connecting it to other services you already use. This brings together email contacts and contacts from your service / social accounts in one easy to access and use place that roams across your Windows 8 PCs and phone.

In this post, Jeff Kunins, a group program manager on the Windows Live team, details the People app. This is the first of a series of posts on the new service-connected apps that are currently in App Preview. --Steven

Modern devices come with an address book or contact list because the people we communicate and share with are so important to how we use those devices. Email, texting, phone and video calls, social updates and comments – these are but a few of the people-based activities we do with the phones, PCs, and tablets we use every day. With Windows 8 we set out to meet this fundamental need with a new kind of contact experience: the People app.

The People app in Windows 8 is a modern take on the flat contact lists of the past—it's built for the way you communicate today, and it's connected to the cloud services you already use. The People app connects to your email and social accounts, bringing together all your contacts (and what they're up to) in one convenient place. Windows 8 Consumer Preview users have already used the People app millions of times and received millions of social notifications on its live tiles. We are proud of the [early enthusiasm](#) for our approach, and thankful for everyone's helpful feedback on how we can improve this early preview version. We thought we would take some time to share more of our perspective on the modern social address book, and how our point of view is driving the evolution of the People app in Windows 8.

Modern devices like Windows 8 and [Windows Phone](#) require an address book that's crafted around four simple principles:

1. **Complete & Connected** – All your personal and work contacts are there, alive with their social activities and photos, letting you instantly engage and react to them. Data syncs from your email and social accounts rather than getting this info from a one-time import, and you get a simple unified contact card for each person, regardless of how many versions of their contact info you have from different accounts.
2. **Designed for Windows 8** – On modern devices the address book is a core part of the overall experience, therefore it is important to design it with the whole system in mind. The People app follows Metro style design principles so it is fast and fluid, and it works together with all your other apps through the Share and Picker contracts.
3. **Cloud-powered** – your contacts and settings are effortlessly backed up, so “they just work” when you sign in from a new device, or even from the web. And when you pin a contact to your Start screen, the live tile lights up with real-time notifications about new photos, comments, and tweets.
4. **In control** – you decide what you share with whom across your home, work, and social networks. And of course, those networks decide what information is shared and connected, respecting their policies and customer privacy.

Here's a short video illustrating these principles in the People app:

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

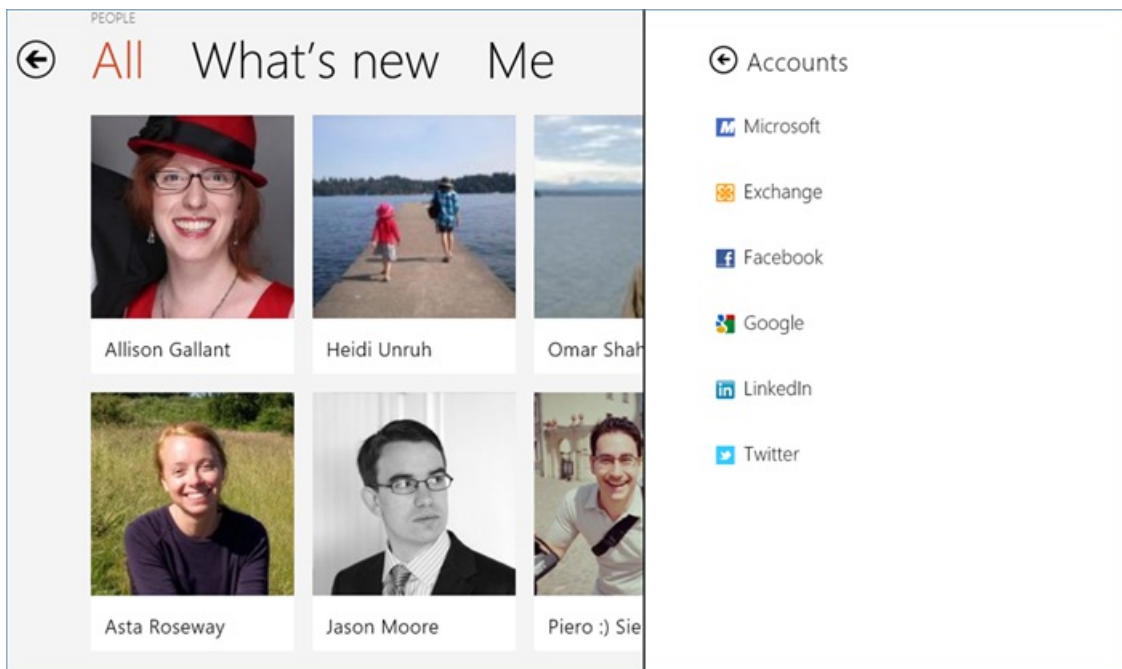
[High quality MP4](#) | [Lower quality MP4](#)

Complete & Connected

[We've talked before](#) about how people should be able to connect the services they already use, without needing to re-spam their friends with invitations. For many years, one of the primary ways that you got contact data into an email account or a social network was to import your contacts (and in some cases, import, and then re-invite them) from another account. This even holds true for mobile phones—how many people do you know who have delayed buying a new phone just to avoid the crazy hassle of “transferring” the contacts from the old phone to the new one?

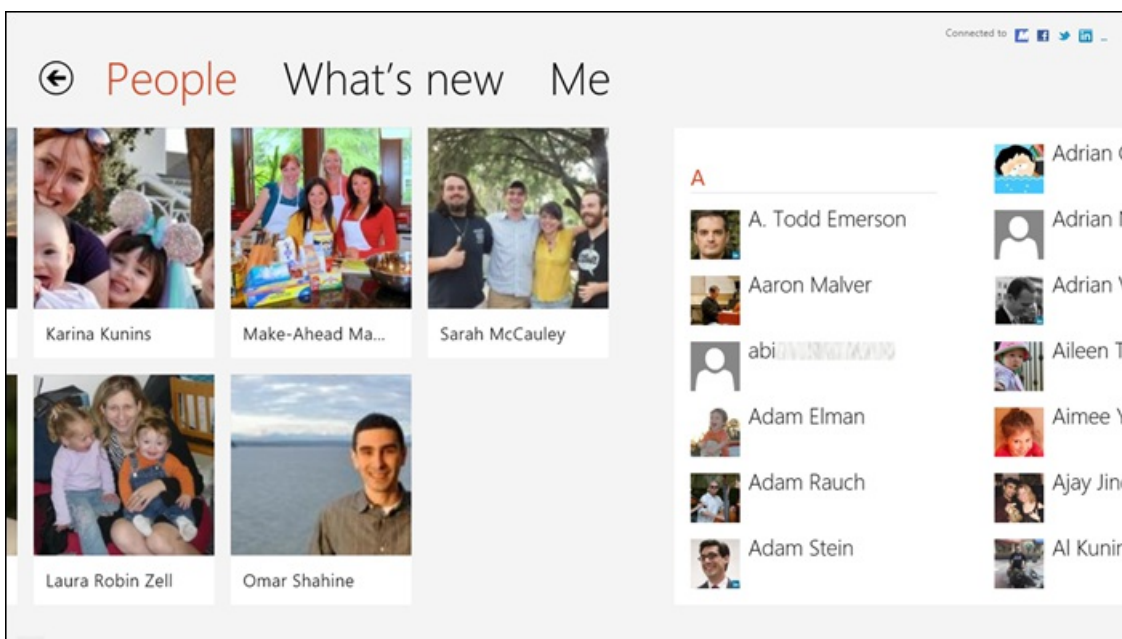
Many of us have had this problem at least partially solved with smart phones that sync our email accounts—but it should just work for everyone, with one easy place for all our contacts and all our accounts.

So, the People app does this. It uses Exchange ActiveSync, as well as the secure, standards-based APIs (OAuth, REST, etc.) exposed by our partners like Facebook, Twitter, and LinkedIn to sync a copy of your contact list from the cloud. It's always up to date with new friends you add (and respects deletions if you un-friend them :-)), so you don't have the problems of a brittle one-time import.

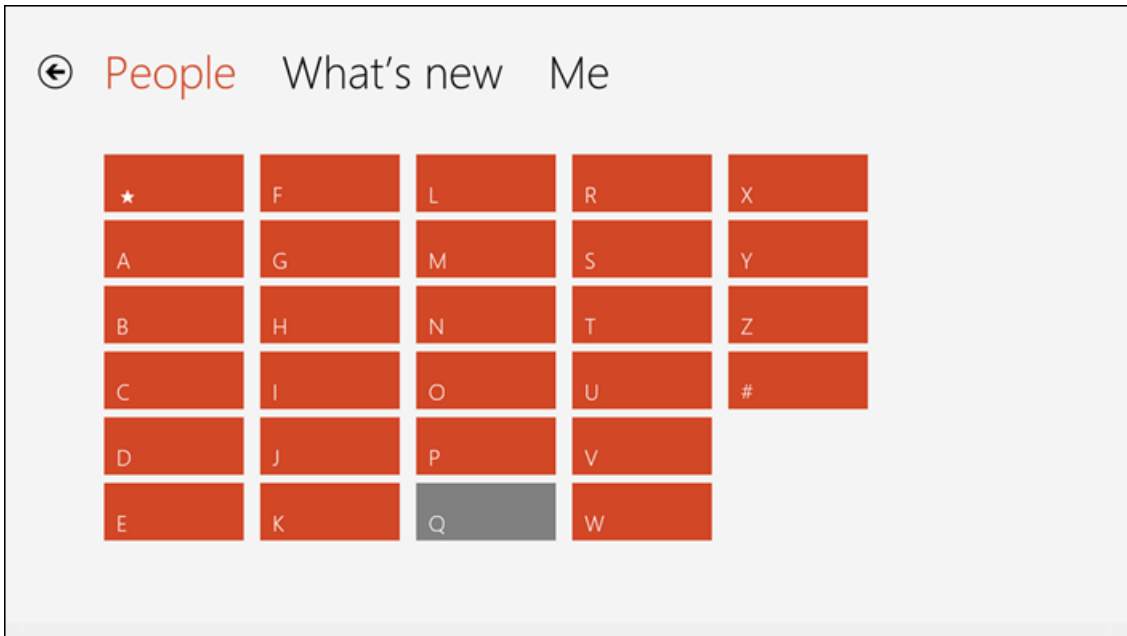


Connecting accounts like Facebook, Twitter, and Exchange

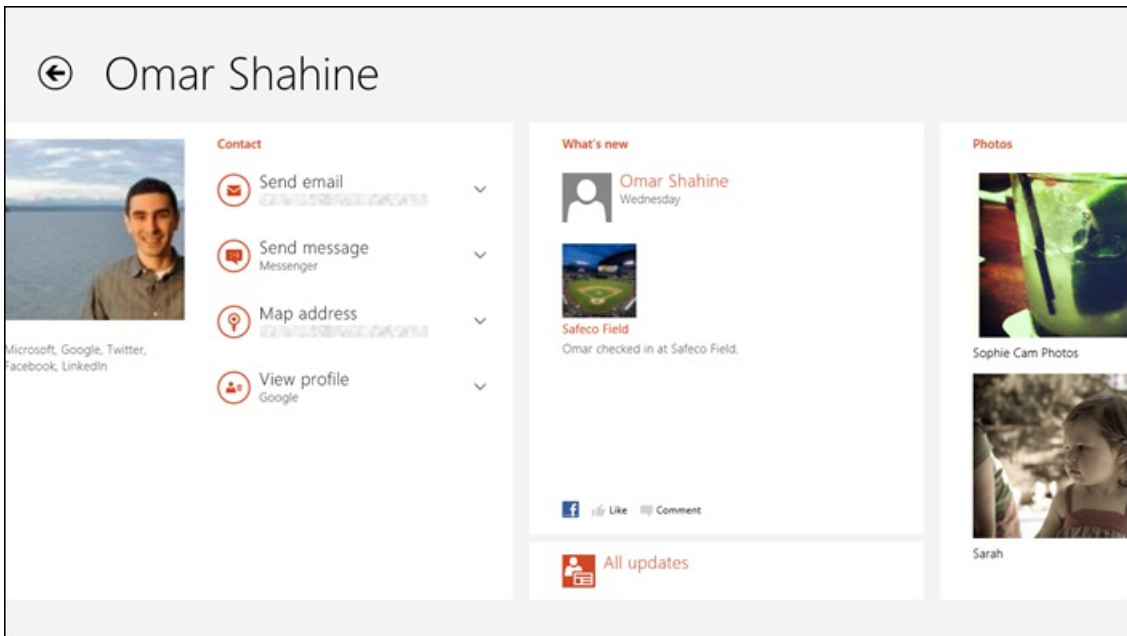
Once your contacts are in the People app, we give you a beautifully tailored experience where you can see and comment on their social activities and photos, view their contact details, or send them a message via whatever service you and that contact have in common. Whether you're browsing the summary of "What's new" across your contacts and networks, or just looking at a specific contact, it's easy to catch up and stay connected.



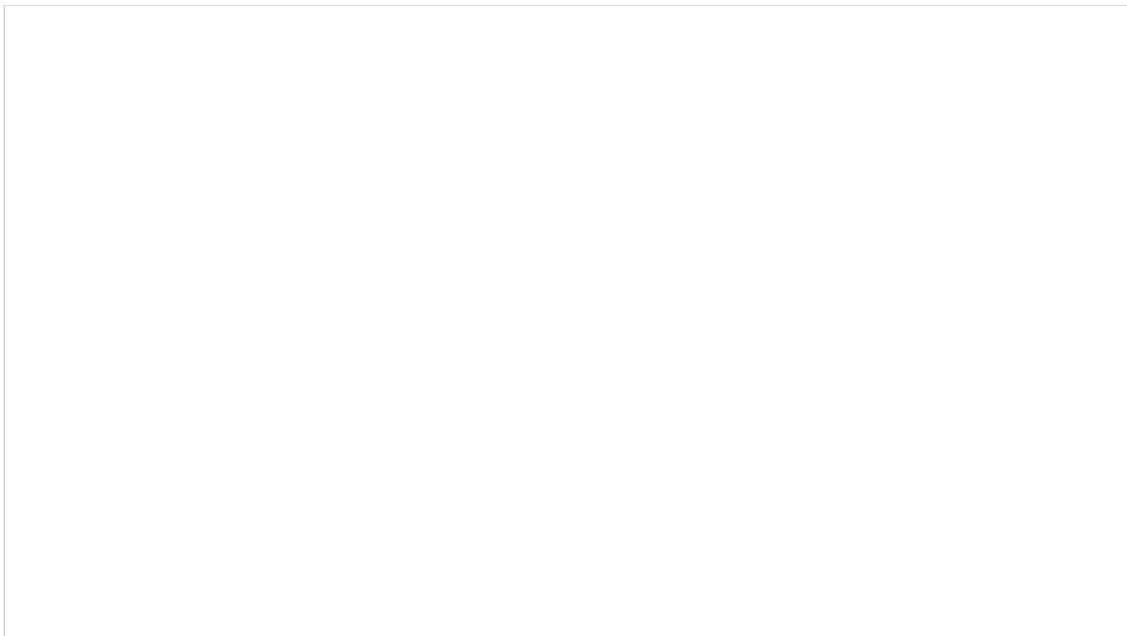
Main contact list view features large tiles for your favorite contacts



Windows 8 "semantic zoom" lets you quickly pinch to navigate your contact list



Looking at a single contact gets you their contact info and what they're up to



Tap “What’s new” for an at-a-glance summary from your social networks



Looking at a single social update lets you see all the comments and add to them

One challenge that any modern address book like the People app needs to handle is duplicates—we all tend to be friends with and have info on a given person from many different accounts. For example, I have contacts for my friend Omar Shahine in Exchange and Hotmail, we’re friends on Facebook and colleagues on [LinkedIn](#), and of course I follow him on [Twitter](#).

The People app – just like Windows Phone and Hotmail – automatically detects that all of these contacts are the same human being (my friend Omar), and presents them to me as a single “linked” contact with all the data together in one place, on one tile, etc. And very importantly, it does this without messing with any of the underlying source data.

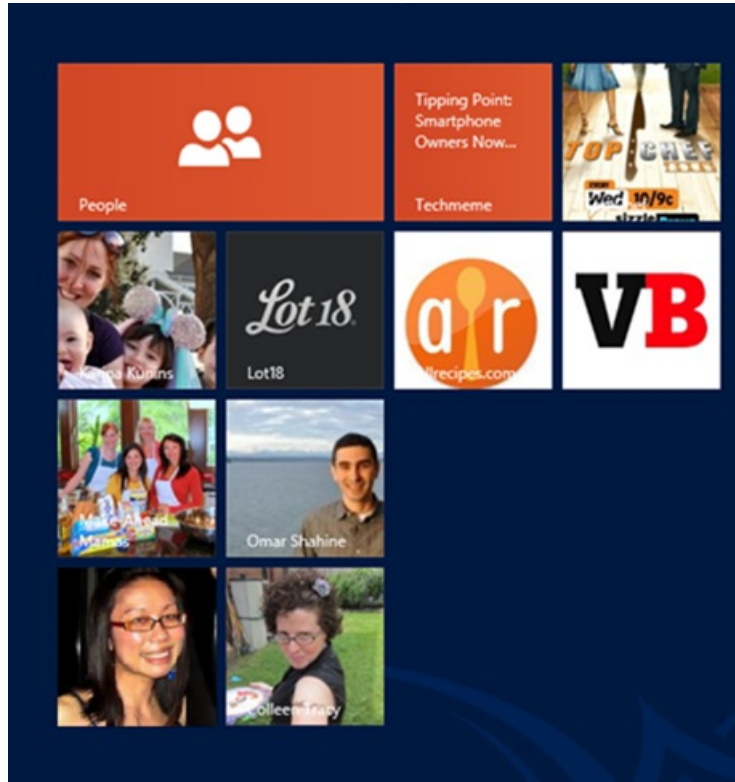
We do our best to get this “right” automatically, but recognizing that we can’t be perfect, we want users to be able to edit and add/remove their own links. We already provide this on Hotmail and Windows Phone, and we’ll be adding that to the Windows 8 version over time.

By connecting multiple accounts and linking your duplicate contacts, we’re able to create a contact card for everyone in your address book, regardless of how you’re connected to them. So, at a glance I can see a quick summary, and then I’m one click away from common tasks – sending an email, starting a chat, getting map information, finding a phone number or address, or just browsing through their recent activity.

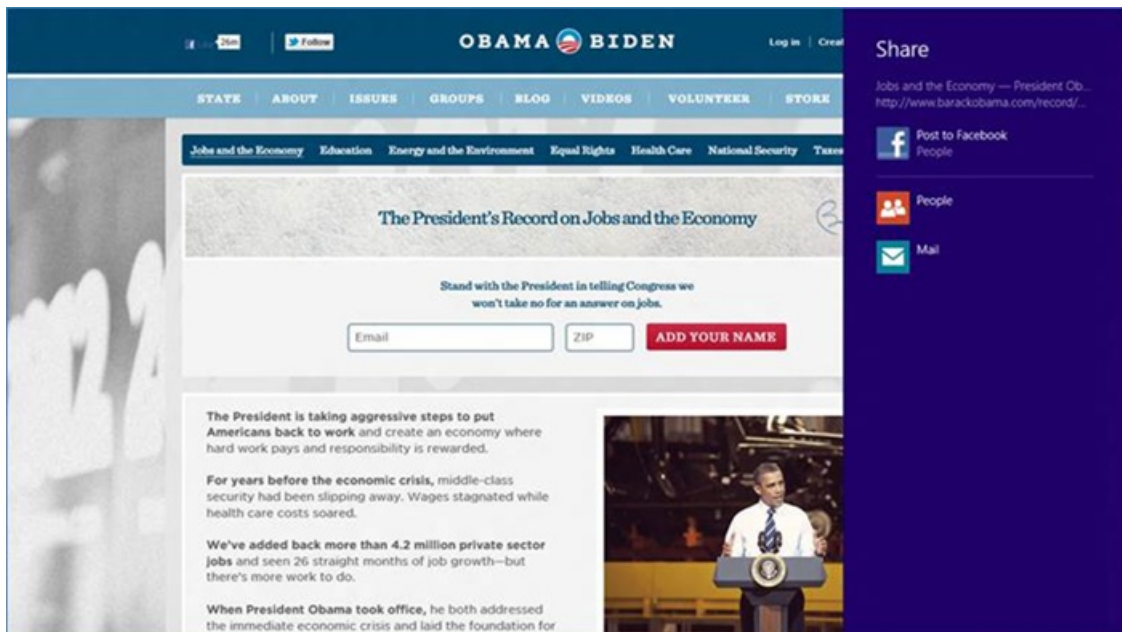
Designed for Windows 8

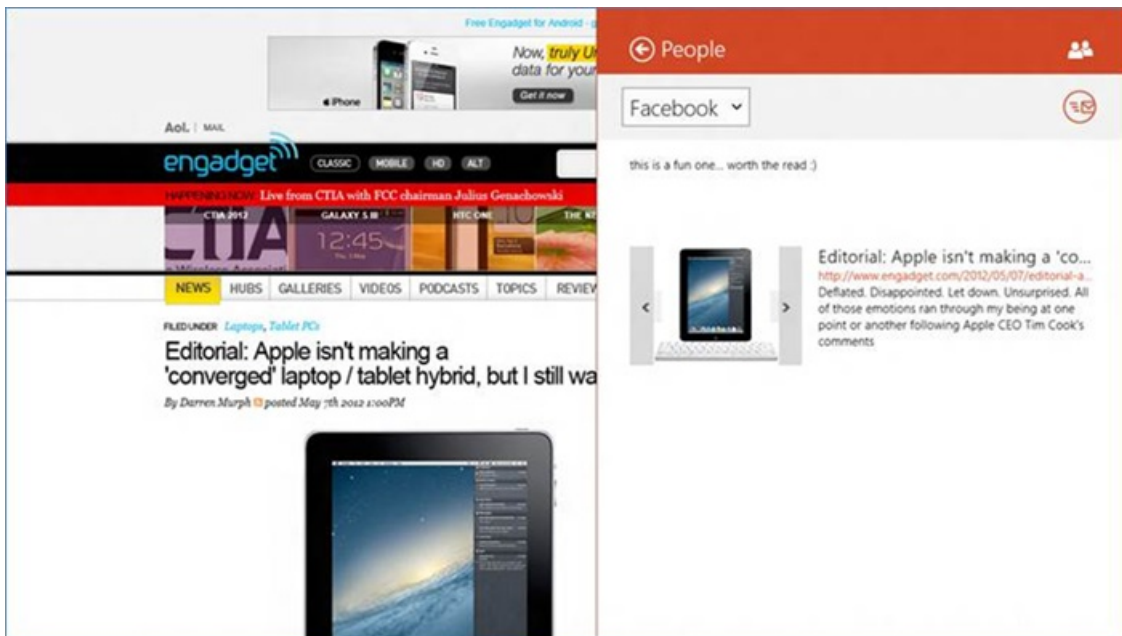
A second principle we followed was to design the app to take advantage of the power of Windows 8. Unlike other systems, in Windows 8, apps can connect to other apps and to the OS itself through APIs that we call [contracts](#). This means not only are the built-in applications like Mail and Messaging powered by People’s contact list – so are the other apps you install on your Windows 8 device.

One of the highlights of the Metro style Start screen are all the tiles that are alive with activity and provide one-touch access to the apps and content you're interested in. The People app takes advantage of the [secondary tiles](#) feature, which lets you have additional tiles that immediately link to that part of the underlying app. So, when I pin contacts like TechMeme, Top Chef, my wife, and my friend Omar to my Start screen, it's just one tap to get right to their contact info and activity, and one more touch to send them mail or look at their latest photos.



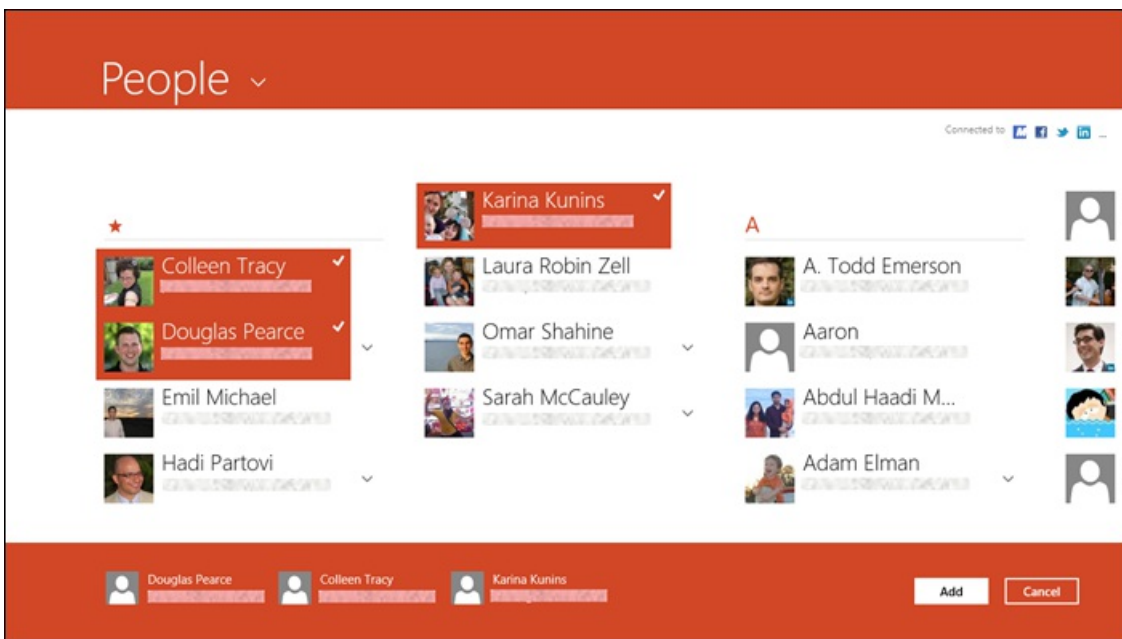
Also new since the Consumer Preview is that the People app now supports the [Share contract](#), allowing you to post to Facebook or Twitter from any Windows 8 app, including Internet Explorer. So, just by connecting your accounts to the People app, with a few quick touches you can share your latest favorite article with your friends and followers.

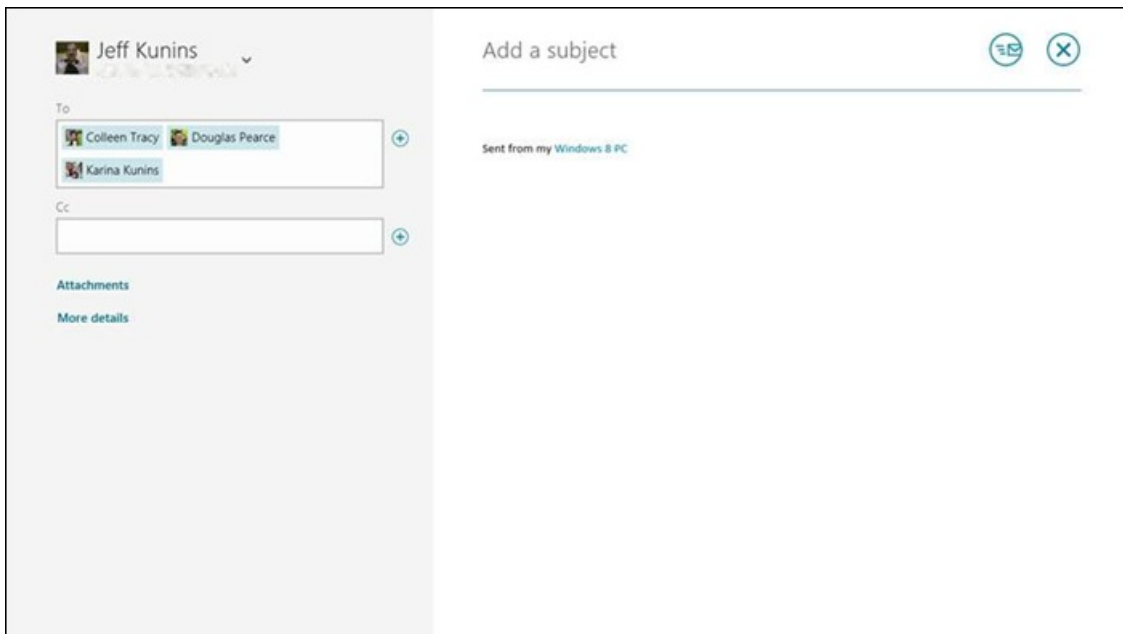




Any Windows 8 app (like IE) can use the Share charm to let you post to Facebook and Twitter

Lastly, another great feature is the [People picker contract](#)—with this, any Windows 8 app can speed up simple tasks like sending a package from a website or emailing a list of friends by letting you quickly select contacts from the People app. And unlike a silent, full-access API, this never happens without bringing up the system-brokered user experience that you're in control of.





Any app can invoke the People Picker, letting you choose contacts to use

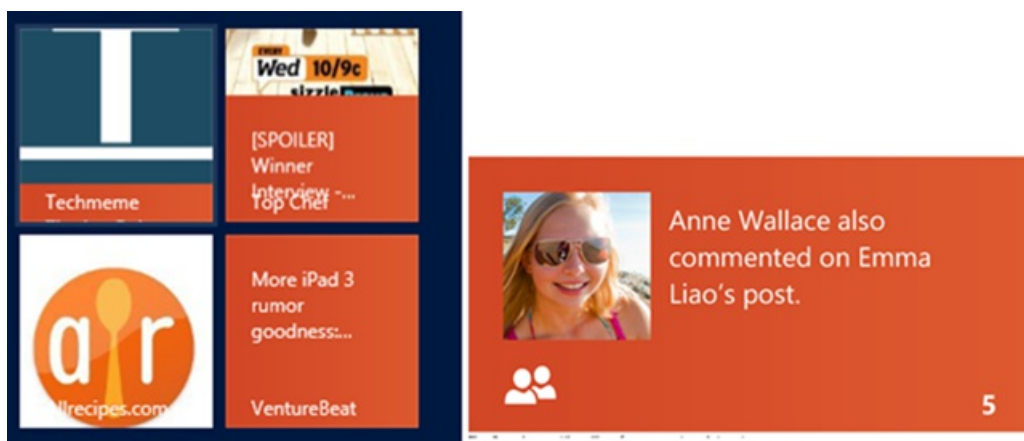
Cloud-powered

None of the features we've talked about so far would be possible if the app and your device weren't cloud-powered, so we can sync data from your various accounts and display it within the app. But if all of your data and settings were solely managed by the "client" app itself (like a traditional mobile phone or email app), then you'd have a few problems: (1) once you set this up on one device like your phone, you'd have to do it all over again when you got a new PC at home or at work, (2) you wouldn't get any of the benefits of this unified experience when you were at someone else's machine just using a web browser, and (3) when services like Facebook or LinkedIn evolve their APIs you'd have to upgrade to a new version of the app before things would work right again.

Our approach is to use your [Microsoft account](#) and the cloud to safely cache your settings, so that when you go to a new device or even access your contact list from the web, things will still "just work" as you'd expect. Additionally, we make many of the API calls to networks like Facebook and Twitter from the cloud, so that we can often adjust to how those APIs evolve without making everyone update to a new version of the app.

A great example of how you benefit from the cloud is that the People app remembers your connection to Facebook, Twitter, and LinkedIn, no matter what device you sign in from. Over **50 million people** have already connected social networks to their Microsoft account through their use of social features in Windows 8, Windows Phone, Hotmail, Messenger, and SkyDrive. Every one of those people, when they sign in to a new Windows 8 PC for the very first time, will automatically have their People app populated with their complete, connected, cloud-powered address book.

Being cloud-powered means that not only do you have an always up-to-date copy of your contact list on your device, but that your People app tile and tiles for your pinned contacts automatically light up with the latest notifications from your social networks. For example, when someone comments on the photos you just posted, the People app tile (and the "Me" Notifications tab in the app itself) will animate with that update and encourage you to take a look. Similarly, each time one of your pinned contacts does something new, their tile will show that activity.



In control

Of course, we respect the policies of each data source we connect to – for example, the People app doesn't currently sync any Exchange data to the cloud. This ensures that data that is governed by your employer's policies aren't even temporarily cached in a third-party data center, even

though it also means you have to set up your Exchange accounts separately on each device and you can't get to them from contacts.live.com. Similarly, Facebook has different policies for syncing contacts' email addresses and phone numbers on specific mobile apps vs. other devices and the web. So, to get Facebook contacts on your Windows Phone you connect it separately in order to have that additional data available to you in the People app on Windows Phone. Twitter also has smart, specific policies regarding how tweets are displayed, which are important to get right.

We also recognize that each person uses their address book and networks in slightly different ways. So we put you in control of what you share with which network and individuals on the network. For example, when you connect Facebook you can decide to connect just the address book, or additional features like instant messaging.

Moving forward

We're excited to see the initial response to our point of view that modern devices should come with a complete, connected, and cloud-powered address book that you're in control of. We hope you enjoy the key additions we've been able to add so far since the Consumer Preview, such as Semantic Zoom and letting you share to people you know on Facebook and Twitter from any Windows 8 app that uses the Share contract.

--Jeff Kunins

Building the Mail app

Steven Sinofsky | [2012-06-14T11:00:00+00:00](#)

*Hundreds of thousands of folks have been using the “App Preview” of Mail on a daily basis since the Windows 8 Release Preview. We’ve also been updating it along the way through the new Windows Store with more updates planned. In this post we go into the background of the Mail app and talk about the design and features, especially relative to Metro style design principles. This isn’t an exhaustive list of Mail features or features yet to be added and primarily focuses on the design and integration with Windows 8. **This post was written by Jeremy Epling, a lead program manager on the Windows Live team.** This is the second in a series of posts on the new apps. --Steven*

When we started planning the email experience for Windows 8, our goal was to create an app that embodied the Metro style design principles. It needed to be fast and fluid, be great with touch and a keyboard and mouse, focus on your content, provide the right features at the right time, and fulfill our expectations of email on modern devices. Starting from scratch gave us an opportunity to carry forward the essential functions of an email app, while also designing features with a fresh eye and taking advantage of what Windows 8 offers uniquely.

How people use email today

At the start of our design process, we conducted research into how people use email today. Email has been around for decades. It’s changed a lot and so have our expectations.

Multiple email accounts are common. The average user has 2 to 3 email accounts. One is for work, one is personal, and another account might be used primarily for mailing lists and coupons, or isn’t used frequently, like an account from a school that you no longer attend.

We receive a lot of email. Our data shows that those who we would consider light email users receive more than about 180 messages a week, while heavy email users receive more than 2100 messages a week. These numbers are growing as more services come online and support newsletters, coupons, receipts, and other types of messages via email. We need to make it easy to quickly get through all your email.

Folders aren’t used that often. This is probably a surprise to many people who rely heavily on folders, which is a very common practice in many enterprise environments, and for enthusiasts. At some enterprises, users have up to 50 folders, while the majority of people using Exchange and Hotmail have far fewer folders. The right balance for Mail was to make folders easy to use, but not to optimize for 50+ folders and deeply-nested hierarchies.

Email is real-time. While email is often used for asynchronous communication, where you don’t expect an immediate response, more and more, the expectation is for an immediate, real-time response. After you sign up for a new service, you’re often told to expect an email immediately. We expect to be notified the instant a new email comes in and most people check their email frequently throughout the day or leave it running all day long so they can see every message as it comes in.

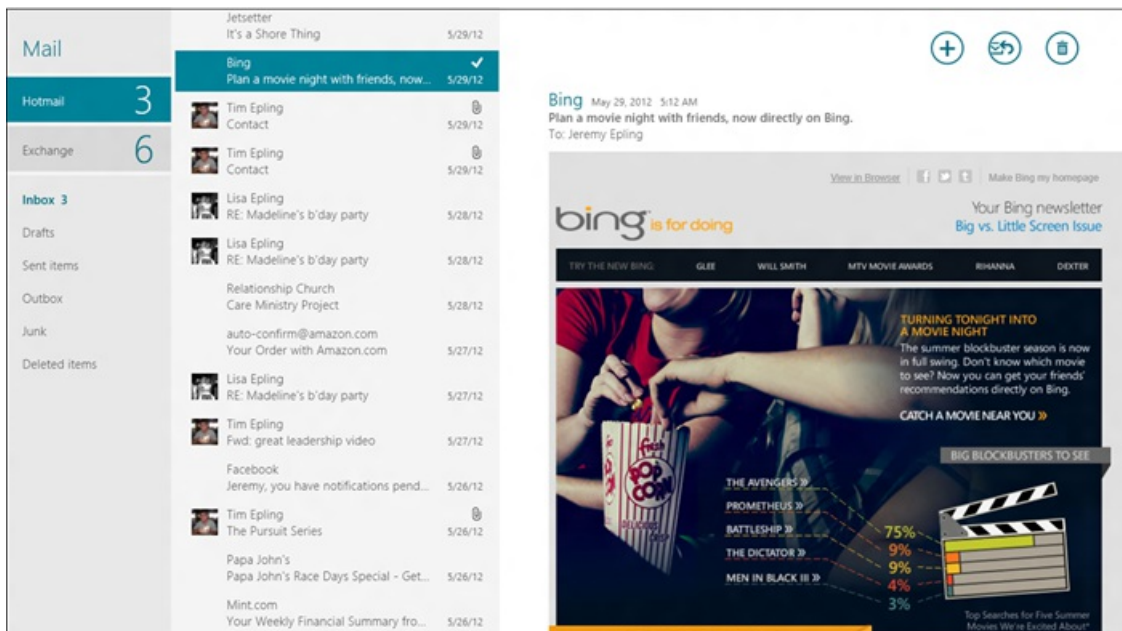
People expect consistency with mobile phone experiences. Many people are using their phones in conjunction with their PCs. In fact, they’re using their phones for triage, reading, and filing away of mail (among other things). The importance of consistency between your phone view and PC view of email are more important than ever. The use of standard protocols such as [Exchange Active Sync](#) as implemented in the Mail app are increasingly important, especially because this protocol allows for syncing of contacts and calendar, in addition to mail. (Don’t worry, support for other protocols, such as IMAP, are on the way.)

We took these trends into account as guiding principles as we began to consider how people would use the Mail app to manage their email, write messages, and stay up-to-date.

Managing email

Accounts and folders

One of the goals of Metro style design is to emphasize the content of the app, and de-emphasize UI commands or navigation that you use rarely. We wanted the Mail app to allow you to focus on the most important aspects of doing email. The 16:9 aspect ratio of Windows 8 made it possible for us to comfortably fit all the essential pieces of content that we use every day: accounts, folders, messages, and a reading pane. It provides an easy way for you to quickly take in all of your email without switching views, and while still feeling open. This is a change from the way it appeared in Consumer Preview, where we only showed messages and a reading pane. We realized that switching accounts and folders wasn’t fast and fluid enough (and customer feedback supported this conclusion). Because folders and accounts are key pieces of content that you need to see, we updated the design to show them in Release Preview.



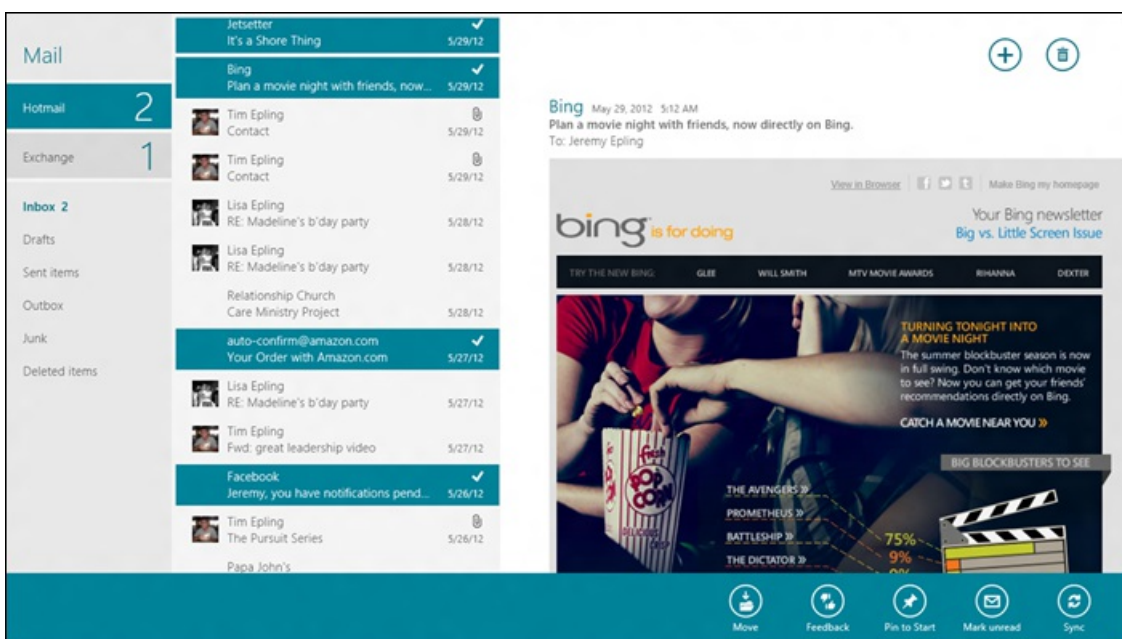
3-pane design of the Mail app

Both of these realizations led to the 3-pane design that you see in the Release Preview today. This design shows your different accounts so it is just one tap to switch between them. The unread count on each account makes it easy to see if you have new mail to look at in that account. The same is true for folders. Even though most users don't have many folders, they are a core way that many people use email, so we made it easy to quickly switch between them. An always-present folder list is especially valuable for people who use server rules to automatically filter their email into specific folders.

Commands

We spent a lot of time deciding which functionality, or commands, would be always visible in the app so that most people wouldn't be distracted by commands they never used. We decided to include commands for the tasks that every person uses almost every time they launch Mail: creating, responding to, and deleting messages. All the respond commands are grouped into a single top-level command since they perform a similar function. Delete is in the corner, which also aligns with the Cancel command when writing an email.

Some people change the read/unread state of emails or move emails frequently, but for the majority of users, this is actually a pretty rare task. Deleting, starting a new mail, and responding to mail dramatically overshadow these as common tasks, so we made sure these commands would always be visible. The other commands are quickly accessible via the app bar at the bottom of the screen (Windows key + Z, or swipe up from the bottom of the screen, or right-click to invoke the app bar). If you select multiple messages, we anticipate that you're likely to use "Mark as read" or "Move," so we automatically bring up the app bar for you.



The app bar automatically appears when you select multiple messages

Message list

In Release Preview, we also updated the message list to show as many messages as possible, to help you get through your email more quickly. The message list spans from the top to the bottom of the app, doesn't show a message preview, and uses a smaller font than before. At a resolution of 1366x768, this allows you to see 14 messages instead of the 8.5 messages you could see in Consumer Preview. It's a delicate balance to create a message list that has large enough items to work great for touch, but still provides the density that many enthusiasts expect. We also added profile pictures to the message list, so you can quickly spot messages from the people you care about most, like friends and family. This helps the message list come alive and provide a more personal experience. The profile picture comes from your friends on different social networks that you've connected to your Microsoft account.

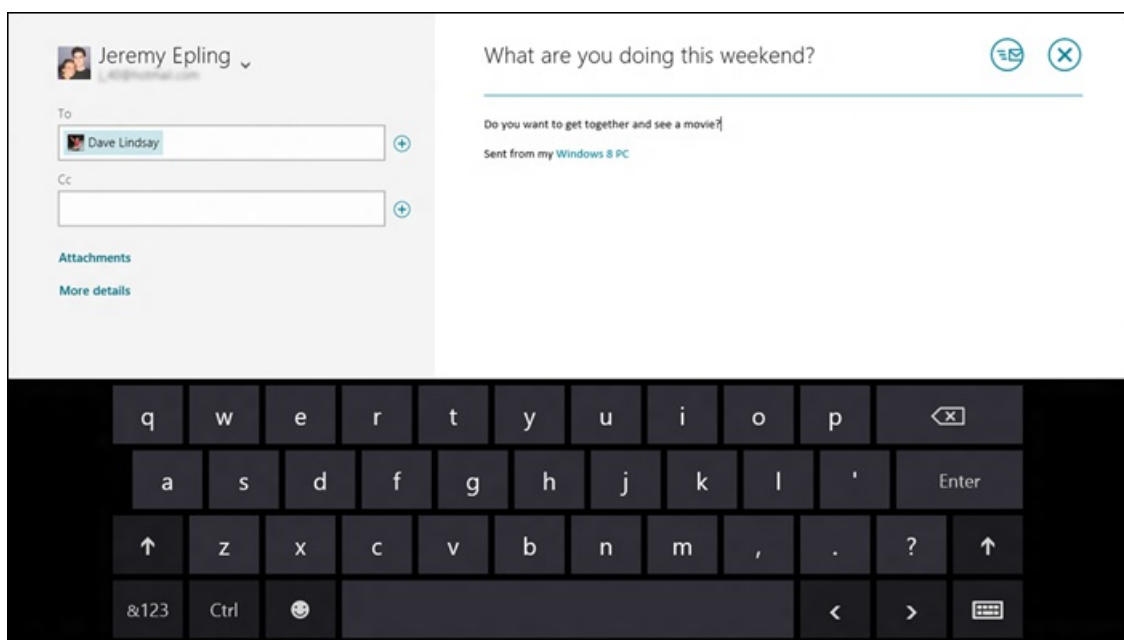
Reading pane

The reading pane makes it fast and fluid to switch between messages, so you don't need to go through a full page transition. The reading pane is optimized to be 640px wide so it can fit newsletters, receipts, and other commercial mail without showing a horizontal scrollbar. Also, we've found that when using our default reading font, 640px is the optimal width for reading a line of text so you don't get eye fatigue or lose your place. When you receive an email we restrict the text to conform to this optimal line length, whenever possible.

There is a large profile picture so it's easy to see who sent the message. We made the subject line bold to make it stand out more in the reading pane, since it sets the context of the message. If you know on the sender or another recipient of the message via a social network that's connected to your Microsoft account, you can tap on her name to view her profile page in the People app. From there, you can see her latest status, send an IM, etc.

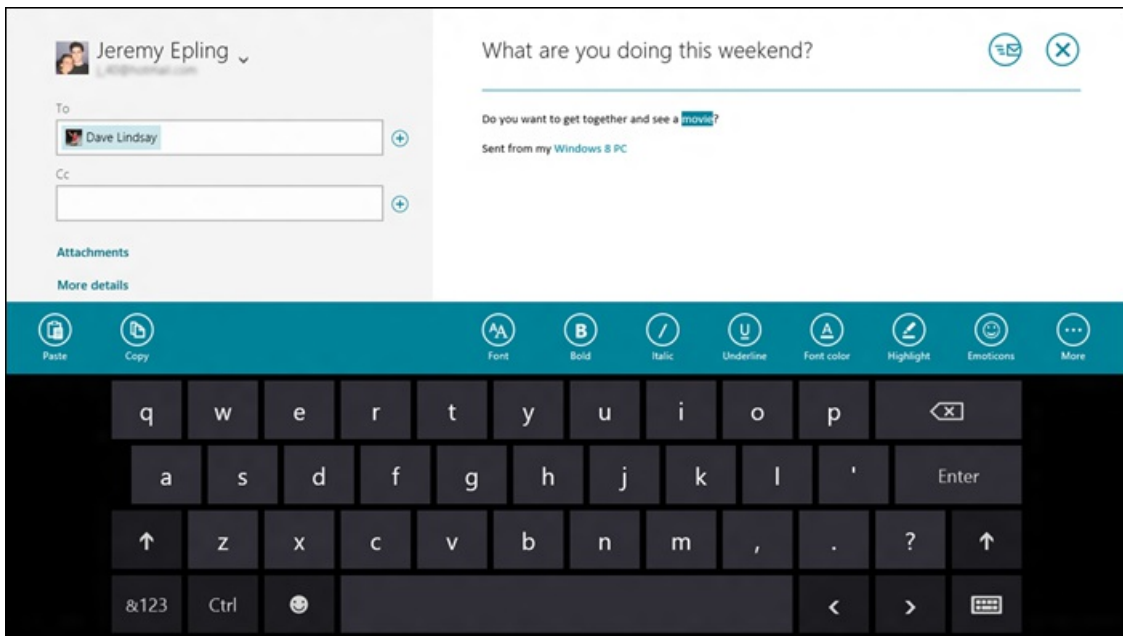
Writing email

The screen you see when writing an email is composed of two panes, side-by-side, so that you have more room to write your message. The touch keyboard limits the amount of vertical space available, so it didn't make sense to put the To, Cc, and other information above the body of the email. To create more space for your content, we put the To and Cc lines in one pane, and the subject and body into another. This also groups the info logically: all the information related to addressing the message is in one group, and your content is in another. The formatting commands are hidden by default to give you more space to write your email.



Writing an email

Even though formatting is not used frequently in email, it is critical when you need it. To make it easier to format messages, Mail automatically shows the formatting commands when you select text in the message pane. After you apply formatting, the commands go away so you have more room to focus on what you're writing. Our goal is to provide the right commands at the right time. When you've selected text, it's most likely because you want to copy or format that text, so those options automatically come up. Many of the formatting keyboard shortcuts that you're used to from Outlook work as well.



When writing an email, select some text and the formatting commands automatically appear above the keyboard.

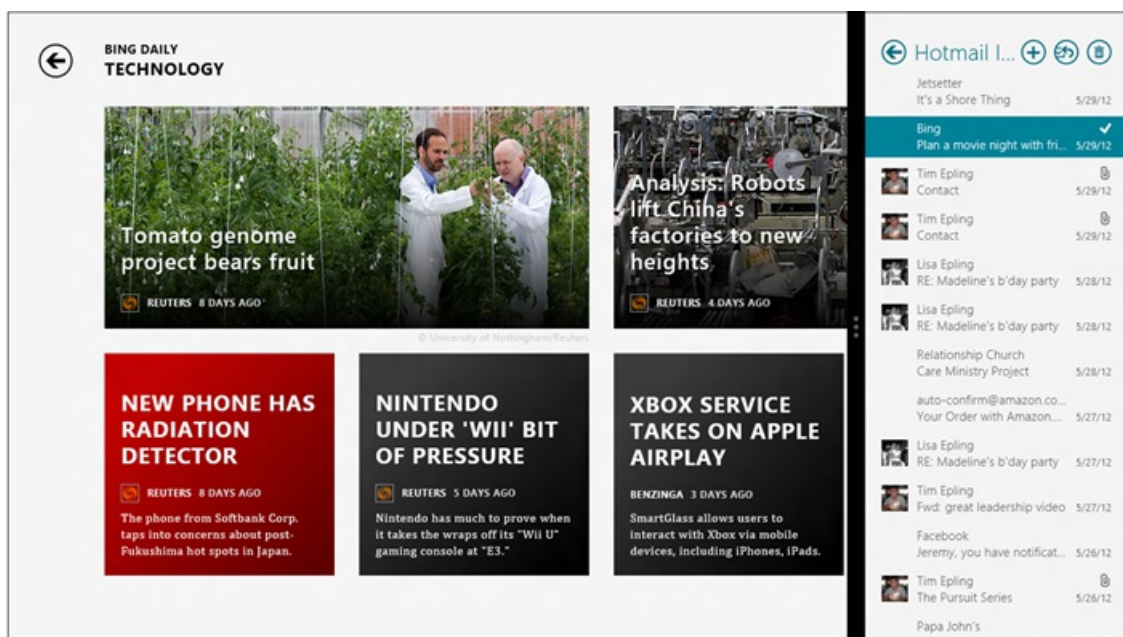
Designed for Windows 8

Another part of designing a great Metro style app is to make sure it takes advantage of the unique aspects of Windows 8. Mail does this by deeply integrating into the operating system to make it easier to share, print, and stay up-to-date on your email.

Snap

I frequently snap Mail to the side of another app (or the desktop) so that I can easily stay on top of it while I'm doing something else. It allows me to instantly see when I have new email and act on it. I can delete, move, or respond directly from the snapped Mail pane, so I can quickly get back to what I was doing. In Release Preview, you can also switch accounts and folders in the snapped state, so you can stay on top of any folder or account while you're using another app. With these updates, it's easy to keep the snapped view of Mail up all day long.

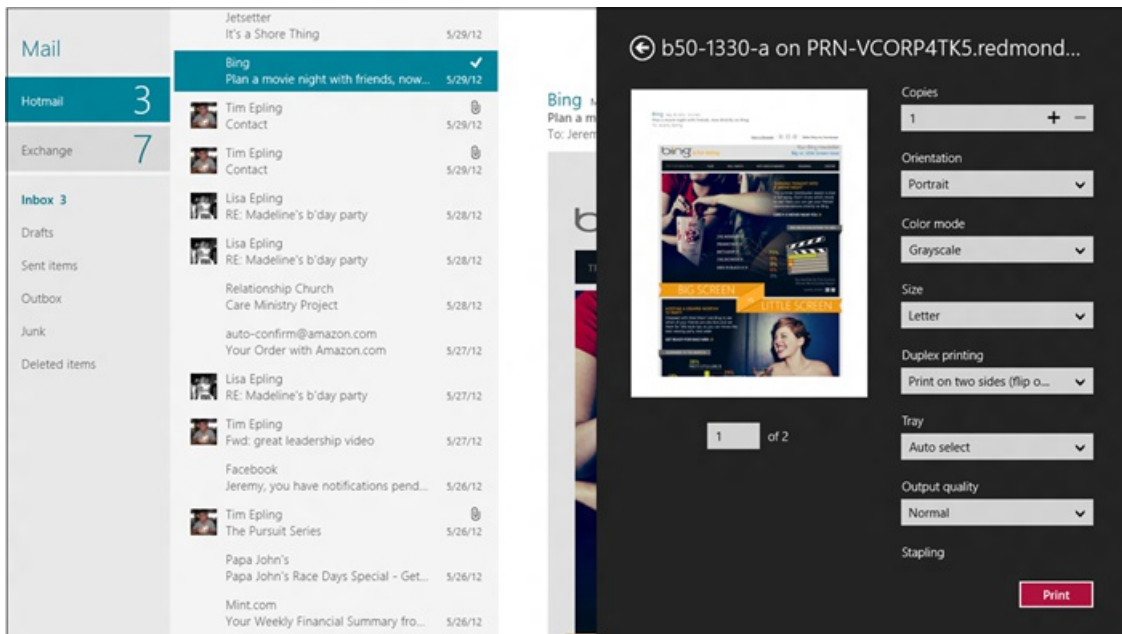
This is really useful if you are composing a long email and need to copy and paste things into it from multiple apps. You can start a new message, snap Mail to the side, and then on the main part of the screen, switch between other apps to get everything you need and directly paste it into the message.



Mail snapped to the side of IE

Print

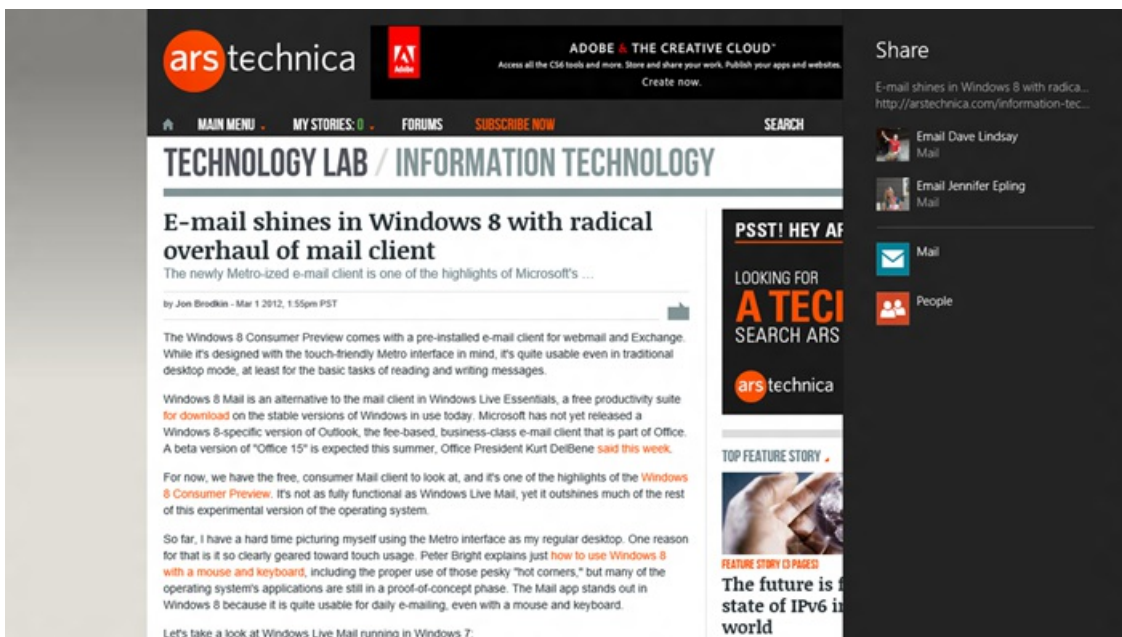
Printing is still a common need—to print a ticket, receipt, or coupon you received in email—and it's something every Windows user expects to just work. In Release Preview, you just need to select the email you want to print, then open the Devices charm and select the printer you want to print it with.



Printing in Mail

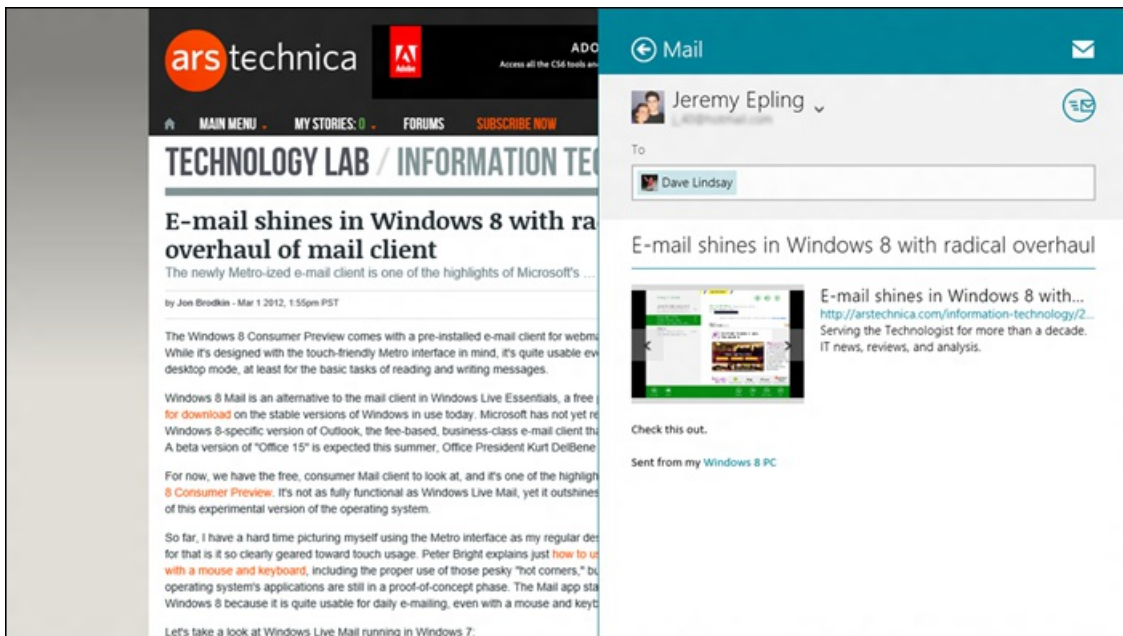
Share

Mail integrates with the [Share contract](#) so that you can easily share to Mail from any app. Many times you don't want to send something to your entire social network. Instead, you want to send a link, some photos, or a game score to just a few of your friends. Mail provides a great way to accomplish targeted, private sharing from other apps via the Share charm. If you're sharing with the same group of people again and again, Windows remembers that group so it's easier to share with them the next time.



Open the Share charm from IE and you'll see a list of the people you commonly share with using Mail

Mail supports sharing text, links, and pictures. If the app provides a public URL, Mail automatically grabs a picture, title, and description from the webpage. Then, you can add your message and send it to your friends. Using Mail from the Share charm looks and behaves the same as the when you compose a new message in the Mail app, so all your formatting keyboard shortcuts still work, like CTRL+B for bold.



When you share from a webpage, you can send a preview in Mail

Live tiles

We expect modern devices to always be up-to-date with the latest info. The Mail tile does this by rotating through the last 5 unread and unseen messages. This lets you know if there is something new since the last time you checked your email.

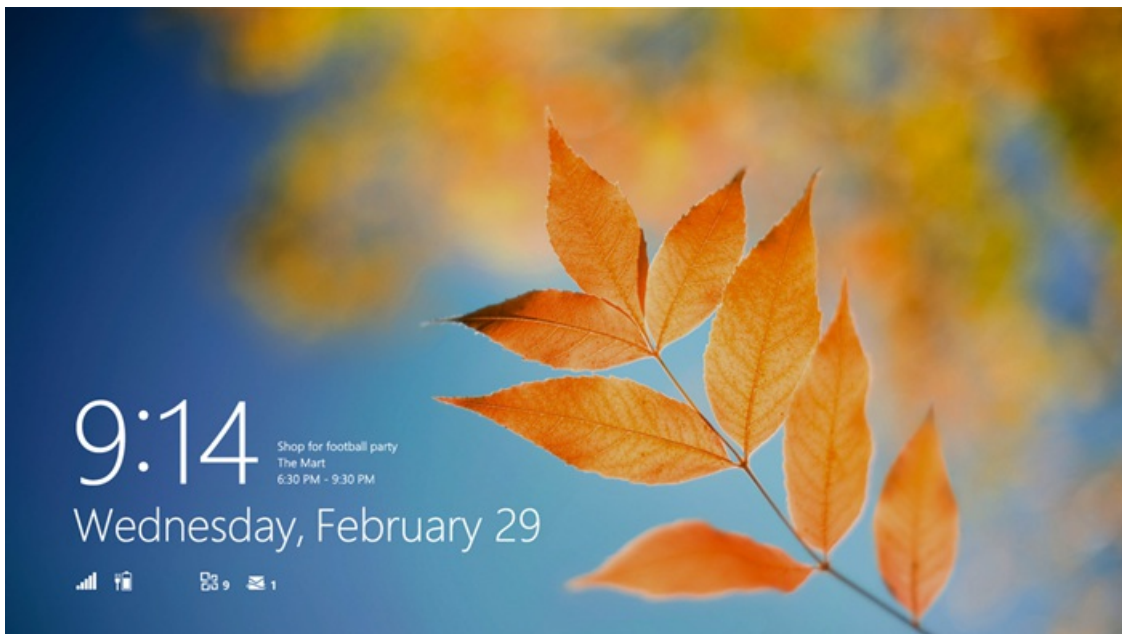
You can also create a [secondary tile](#) for any email folder or account, and pin that to the Start screen to see live updates of new mail in just that folder or account. This is very convenient if you use server rules to automatically move email to another folder.

For example, I've arranged Start to have separate tiles for my corporate Exchange account and my Hotmail account, so I can easily tell if I have new email in either account.



You can pin tiles to the Start screen for any email account or folder

You can also put these secondary tiles on the lock screen so it's easy to see if you have new email and what folder it's in, without signing in to your device.



Lock screen with new mail counts for both Exchange and Hotmail

If you want to know immediately when new email arrives, you can turn on notifications for each account by going to Settings, and then Accounts.



New email notification

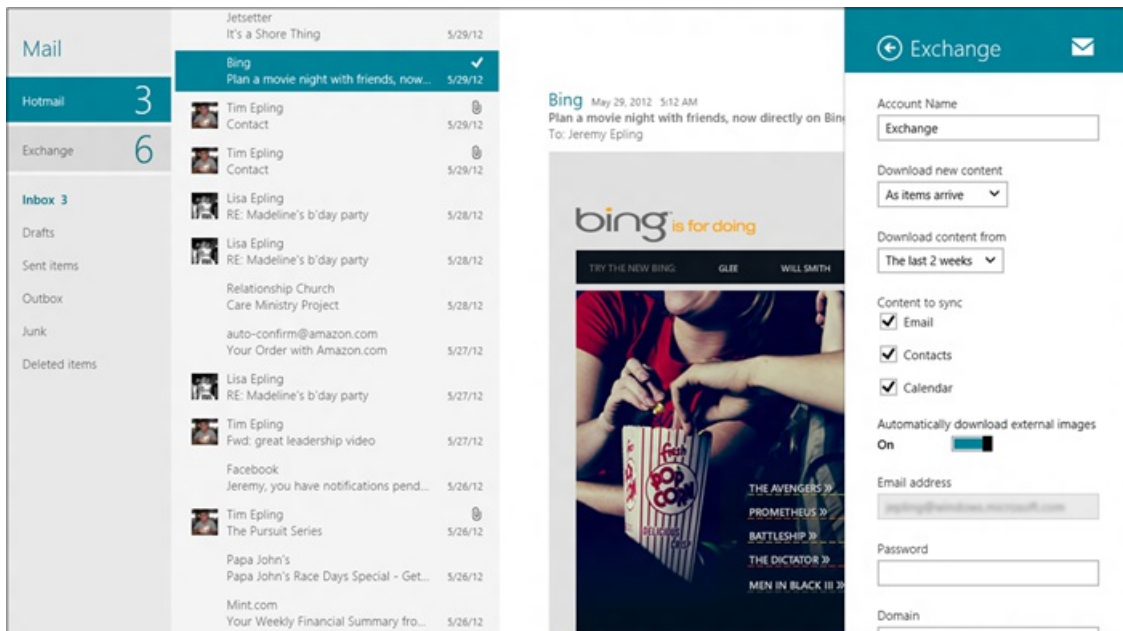
Of course, all of these are customizable, and if you prefer, you can reduce the amount of information that appears on the tiles or the lock screen. You can turn off Live tiles individually for each of your pinned folders or accounts. Notifications can be controlled for each account from the Settings charm, and include a sleep mode with several levels of silencing.

Always up to date

On a modern device, we just expect to have our email (and other information) always up to date. We also expect our apps to be respectful of CPU usage and battery life. Mail strikes this balance by using the [background processing APIs](#) built into Windows 8. These allow the Mail app to be suspended, but still wake up the email sync engine when new email arrives (push) or when a timer fires (polling). We've factored the email sync engine into a separate process so that only the minimal amount of functionality is run to keep you up to date and preserve the battery. This is what drives the Mail app's tile and notifications.

Mail has account-specific settings, so that you can choose the configuration that works best for each account on your device. By default, all accounts will download new email "as items arrive" (push), but you can configure that to happen every 15 minutes, 30 minutes, 1 hour, or manually.

We want to prevent [bill shock](#), so we try to use the minimal amount of data necessary while still delivering a great experience. One way we prevent out-of-control data usage is to only download the last 2 weeks of email by default. We've found that most people only actively engage with the last 2 weeks of email, so we don't download all of the messages in your multi-gigabyte email account unless you specifically configure it that way for any given account. Also, this provides a much faster first download, so you can start acting on your messages more quickly.



Mail default account settings for Exchange

If you're on a metered network where you pay per MB of data you use, Mail uses the new [networking APIs in Windows 8](#) to detect this, and only downloads the first 20KB of each message body and no attachments. For the majority of messages, this is the entire thing. If it's a larger email, it's just one tap to download the rest of the message. If you're on Wi-Fi, the entire message is downloaded by default.

What's next

Email is an important part of our daily lives. We're still continually working to improve Mail and have many more features on the way. Today we're excited about the response to the app and the first preview release. We believe that people want a great email app that meets their modern expectations and, based on usage, we are seeing this in the Windows 8 Mail app.

Thanks for all the great feedback, and keep it coming.

Jeremy Epling
Lead Program Manager
Windows Mail

Designing the Windows 8 Calendar app

Steven Sinofsky | [2012-06-15T15:00:00+00:00](#)

*This post builds on the [Mail app](#) and [People app posts](#), and details the Calendar app. We've worked hard to integrate these apps together into a seamless communication suite that connects to the cloud services most important to you. This post details the integration with Windows 8, some of the features in the current preview, and features on the way. We also look at a little bit of the design history and iteration as some background. **Colin Anthony, a lead program manager on the Windows Live team, authored this post.** --Steven*

When we set out to design the Calendar app for Windows 8, there was no shortage of directional possibilities. Given the long history of calendars in society, and the diversity of Windows customers, we asked ourselves: What are the essential attributes of a great calendaring experience and how can we bring them to life by using the uniquely rich capabilities of Windows 8?

At its heart, a great calendar should do the following:

- **Show your life clearly.** You should have crystal clear visibility into what's happening in your life – at home, at work, and at school.
- **Make it easy to get around.** Moving back and forth in time should be quick and efficient. Opening events and appointments should feel natural.
- **Make it easy to add new items.** New things are always coming up in your life. A great calendar makes it easy to make new plans.
- **Keep you on time.** Well laid plans aren't very useful if you show up late! 😊
- **Be ready to do more.** As you get busier, scheduling gets more complicated. Calendar should gracefully handle your needs as they change.

Showing your life clearly

One of the most important functions of a calendar is its ability to answer the questions “What’s going on for me today?” and “What’s coming up next?” As we designed the Calendar app, we focused on providing that clarity and eliminating distractions. Given all the potential capabilities of a digital calendar, keeping this focus can actually be quite difficult. The temptation to add extra bells and whistles can be very real. At the same time, we realized that focus is one thing paper calendars have always been quite good at --- they simply present you with the calendar grid and the information you've written on it.

With that insight and the clear principles of what it means to be [a great Metro style app](#), we committed to clarity of presentation. The app focuses on the calendar and your content, above all else.

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	
	Memorial Day		1:30 Design meeting			1:30 Book club	
27		28	29	30	31	1	2
	1:30 Team meeting			1:30 Client meeting		1:30 Practice	
3		4	5	6	7	8	9
				Flag Day	1:30 Team meeting		
10		11	12	13	14	15	16
Father's Day		1:30 Project due 1:30 Lunch with Jim			1:30 Conference call	1:30 Softball game	
17		18	19	20	21	22	23
	Design conference						
24		25	26	27	28	29	30

A calendar with no distractions - your schedule is the focus.

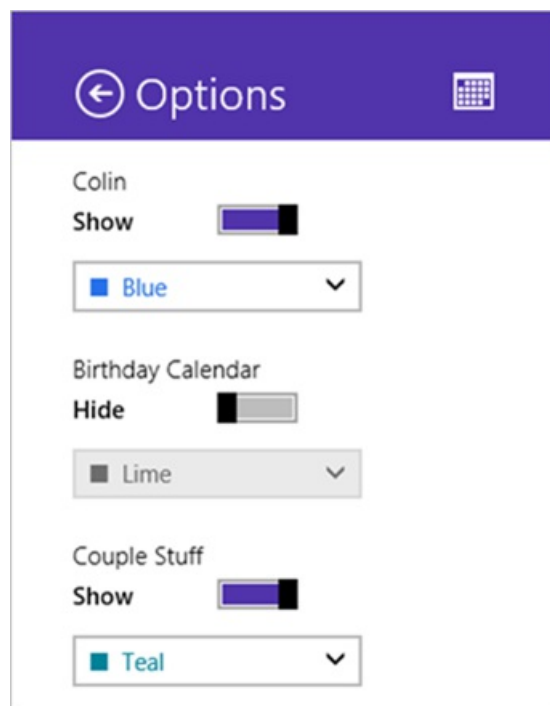
Of course, having an easy-to-read calendar is only useful if all your information is available. Whether you're keeping track of appointments at work, managing family activities, or mapping out your class schedule at school, Calendar brings everything together to provide a more complete picture of your life.

Friday	Saturday
8a Client meeting	9a Volunteering
7p Sarah's graduation party	7p Family movie night
6	7

Work and personal appointments are available in one view

You're also in complete control of how these calendars appear in the app. For example, if a certain calendar has special meaning for you, you can make it stand out by changing its color. If you're overwhelmed by the number of birthdays showing up from your favorite social network, you can hide those items in order to see everything else more clearly.

All of these personalization controls are tucked away neatly in Settings to avoid distracting you when you don't need them.



Having less frequently used Calendar options tucked away in Settings provides control without distractions.

Making it easy to get around

Another important function of the calendar is the ability to move around in time. Our most important goals were:

- Making it simple to move forward and backward in time
- Making view switching predictable
- Making it easy to open and view existing events

Moving forward and backward in time

One thing was apparent to us early on: We wanted a simple model where a gesture forward would take you into the future, and a gesture backwards would take you into the past—a direct way of moving around in time. No buttons to click or extra controls to manipulate.

But even in this simple design there's a question of how far forward and how far back each movement should take you. Not only were there user experience tradeoffs to be considered, but technical ones as well.

If we decided one swipe could move you forward multiple months (depending on the speed of the gesture) that would give you a natural and powerful way to make huge leaps forward in time. But there's a challenge in making the landing place predictable. That is, if you're starting in June, and want to move to July, you might end up in August if you swipe too quickly. If this happens frequently enough, it can feel as if you're not in control.

Additionally, since the boundaries between months are well defined—landing halfway between July and August doesn't make sense—accuracy is important. Contrast this to panning a two-dimensional grid of photos, where “getting close” is ok as long as you arrive in the general region of the photos you're looking for.

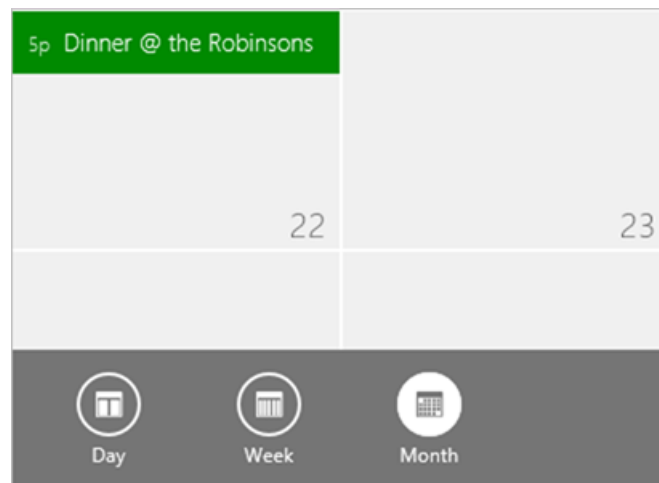
To avoid these pitfalls, we went with an approach that enables you to consistently move forward one month at a time—by swiping once, hitting page down on the keyboard, or clicking the Forward button. This makes your experience much more predictable, putting you in control. If you're in June and you want to arrive at August, you simply click or swipe twice. That's it. You're guaranteed never to overshoot the intended target.



Simple movements forward are quick and predictable

Switching views

When using the app, most people want to stay in the same view the majority of the time, depending on how busy their schedules tend to be. People with fewer appointments tend to prefer Month view. Those with very busy schedules tend to prefer Day view. So, rather than making view switching a top level command that would always be visible, we opted to place it on the app bar instead. You can easily pull up the app bar in Windows 8 with a right-click or swipe up from the bottom edge, so you have access to the view switcher when you need it, without being distracted by it the rest of the time.

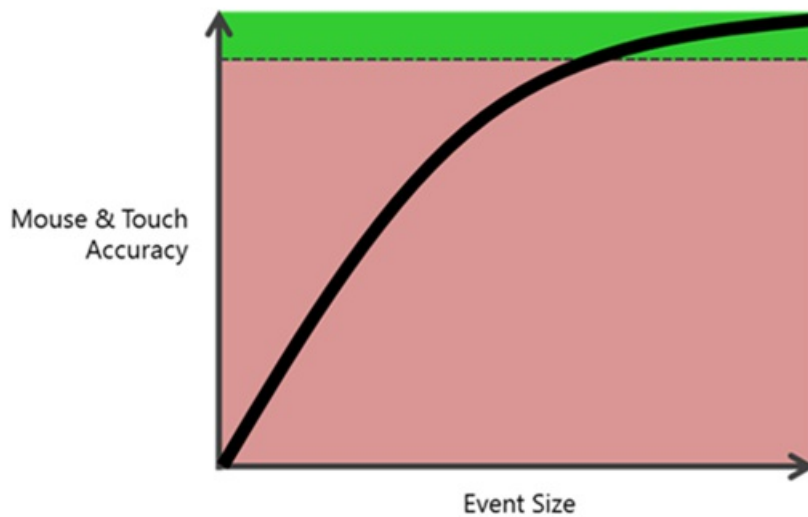


Opening the app bar gives you simple switching between views when needed

Opening individual events

Making it easy to open events is an interesting problem because of the tension between two variables:

- Making each event larger (in height and width) makes them easier to target with the mouse or your finger.
- Making events smaller means you can see more content at one time.



We needed to find a solution that would give you easy targeting while also showing enough content in each view. This was especially important in Month view, which has the tightest space constraints for each particular day. Keeping the key variables in mind, we designed the event sizes within a sweet spot, one in which Month view typically shows 2 events per day (the average for the majority customers) while maintaining nearly 100% accuracy for mouse and touch targeting. And, of course, when you have busier days at home, work, and school, switching to Week or Day views shows you all the events you need while providing the same high accuracy.

Making it easy to add new plans

Because new events can come up in life so quickly, we wanted event creation to be direct and instantly available at all times. When you think about paper calendars, they're very direct—you identify a date, move your hand towards it and start writing. It all happens quite naturally, and we do it without much thought because of the paper calendar's simplicity and purity of design. In a similar way, you can add new events to the Calendar in Windows 8 by simply clicking or tapping on the day or time you want.

	11	12	13
	18	19	20
	25	26	27

Click-to-add is like putting pen to paper

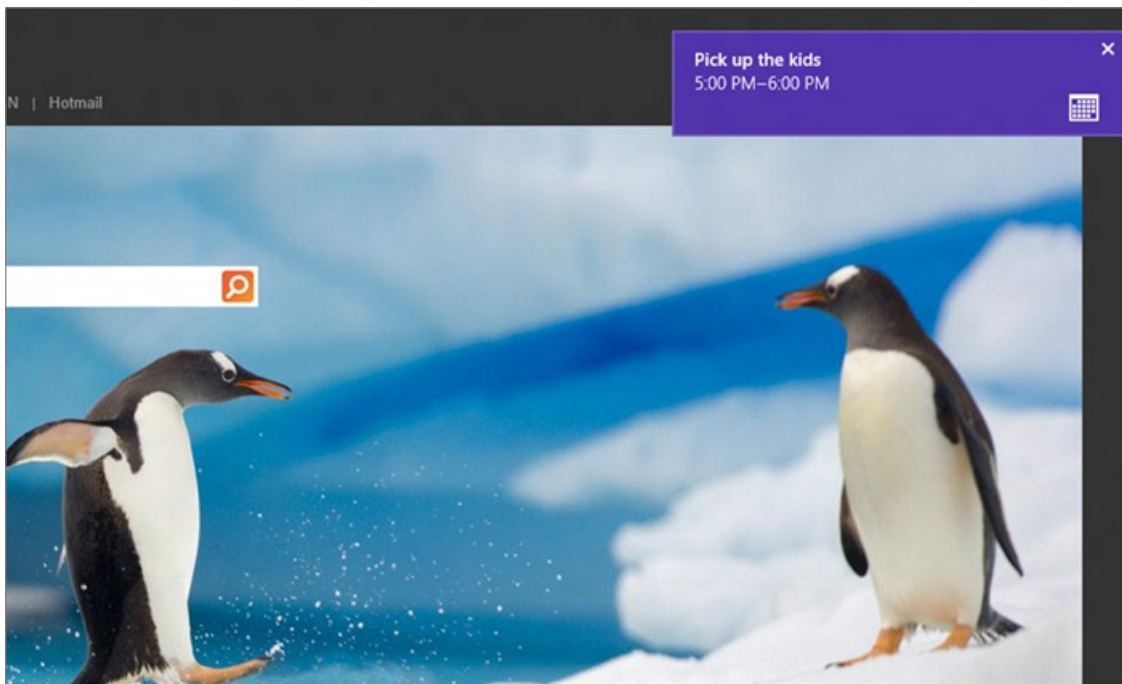
This design—using the grid as the primary command surface—means no other buttons are necessary to complete this frequent task. Everything happens in the calendar grid. This reinforces our Metro design principle of content over chrome. And it allows for a pure experience that isn't just about great consumption, but also about simple and direct creation.

Keeping you on time

Of course, even given all of the above, a calendar is only valuable if it helps you stay on time. Towards that goal, we designed Calendar with a range of smart capabilities to ensure you don't miss any important events.

Notifications and reminders

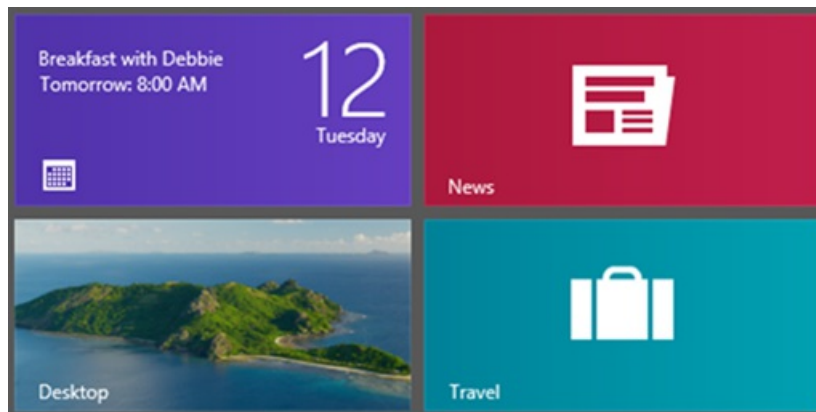
In the full page for an event, you can set any reminder time you want. At the desired time, you'll get a lightweight popup to remind you.



Notifications tell you about upcoming events

Notifications don't block your current task. They don't force you to interact with them. Using the built-in notifications system in Windows 8, a Calendar notification simply nudges you about what's coming up and gets out of your way.

In addition, similar to Windows Phone, you can see information about upcoming events on the lock screen and on the Start tile:



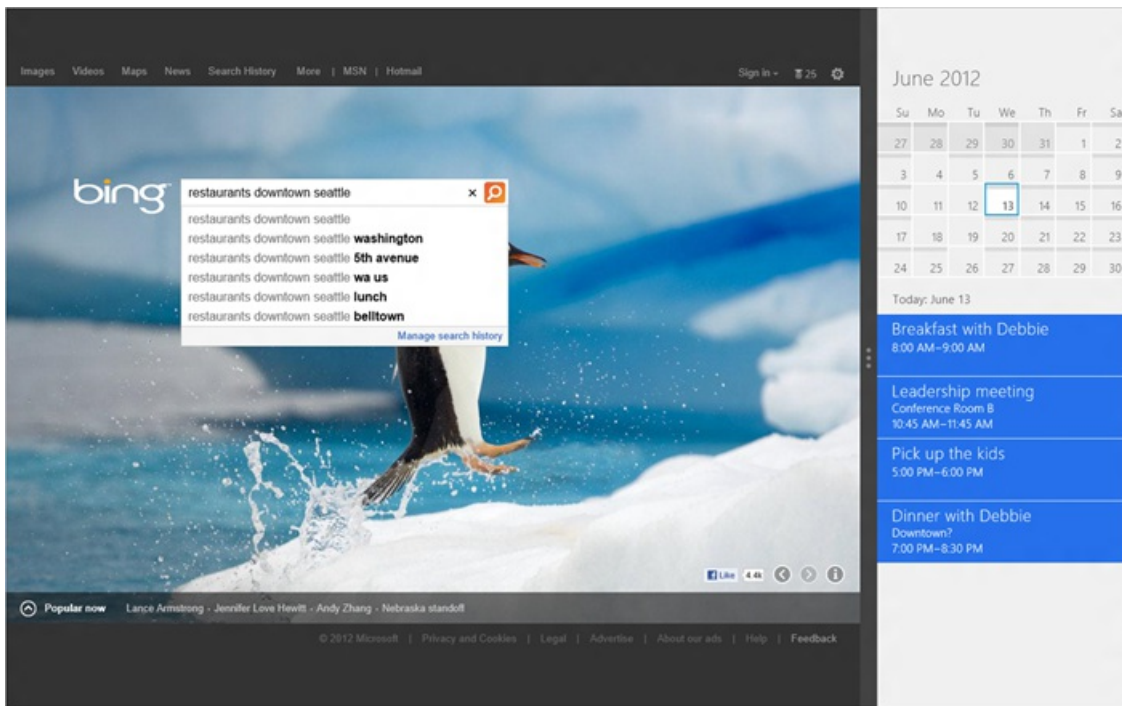
Snapping the Calendar for always-on visibility

When you're extremely busy on a particular day, you often want your full schedule visible at all times. With Snap, you can see your calendar while you use other apps.

Of course, the slim form factor of the snapped view presents quite a design challenge for Calendar. Some views, like Day view, fit very naturally into the more condensed space. To make it fit, we could simply show one day instead of two, and the overall model of the view would stay the same. Problem solved.

But Week and Month view are more challenging. A full week or month is quite difficult to fit in such a narrow area with any hope of showing the actual titles of each event. In addition, we want to maintain predictability in the system by avoiding situations where certain views (like Day view) look essentially the same when snapped, but other views (Month, Week) look entirely different when compared to their larger versions.

Given the challenges and goals, we designed a simple and consistent model where all views are represented by a single, snapped view. This single view maintains the context and positioning from the larger views. (That is, if you're on Wednesday in the large view, you're still on Wednesday when going to snapped view.) It's also designed to give you the same life clarity and ease of navigation: All events from all of your calendars are available in a simple list. A swipe or click to the right or left takes you forward or backwards in time.



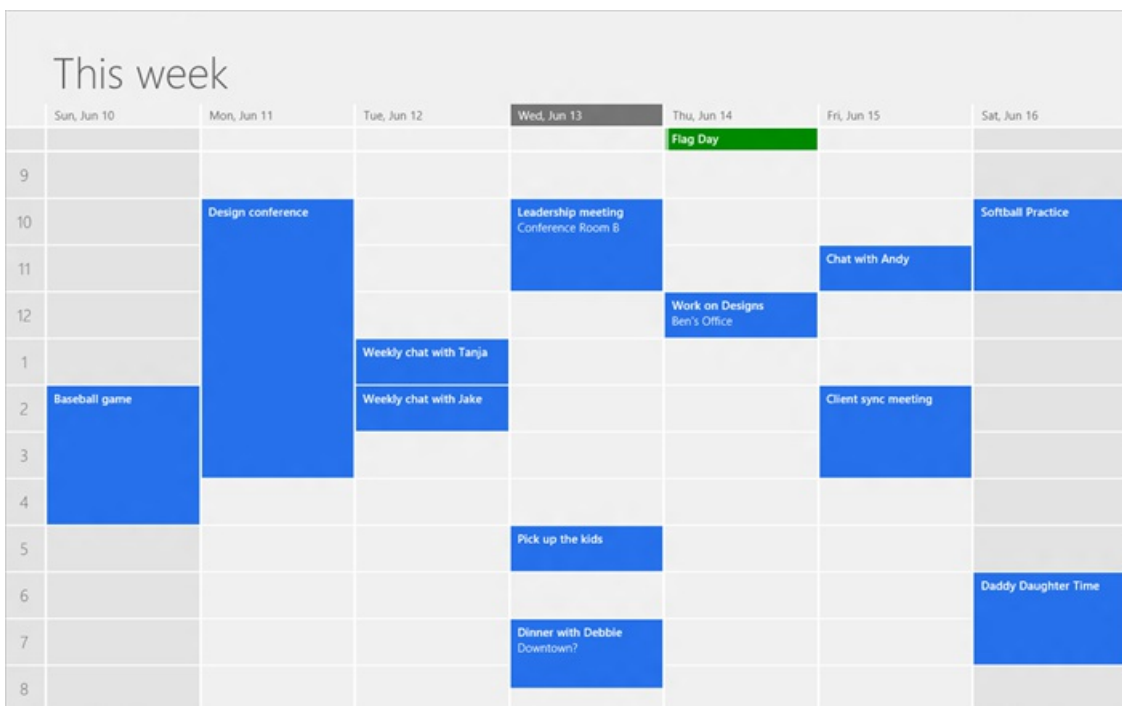
Calendar in snapped view lets you plan and browse at the same time

Ready to do more

Lastly, if we've done our job well, more and more customers will start using Calendar to manage larger portions of their lives at home and work. With that in mind, Calendar is designed to do more as your usage increases.

Week view

As you'd expect, Week view is useful when you have a large volume of appointments, and it's crafted to give you the clarity you need for a particular week:



Week view lets you see a Sunday to Saturday overview

Two-day view

We also know there are days where you're extremely busy at work, home, school, or all three. And we wanted a design that would allow even greater focus and precision. At the same time, we didn't want to zoom in so far as to create "tunnel vision" by removing too much context. To strike the right balance, we designed Calendar with a two-day view:

Today		Tomorrow	
Tuesday, August 16		Wednesday, August 17	
	Dry cleaning		
9		9	
10	Conf call to SVC	10	1:1 Meeting (Alan's office) 1:1 Meeting (Colin's office)
11		11	
12		12	
1		1	
2		2	Design Offsite
3		3	
4		4	
5	Soccer Practice Magnolia Park	5	
6		6	

Two-day view gives you the best detail about today and tomorrow

The two-day view is useful because making good time management decisions *today* often requires understanding what's coming up tomorrow. In addition, this view takes advantage of today's modern wide-screen displays without adding additional chrome or distractions just to fill up the extra space. As with the other views, it's simply the calendar grid and your events. (And here's a tip: each day scrolls independently without affecting the adjacent day. So, you can view information about tonight, while keeping an eye on your first appointment tomorrow.)

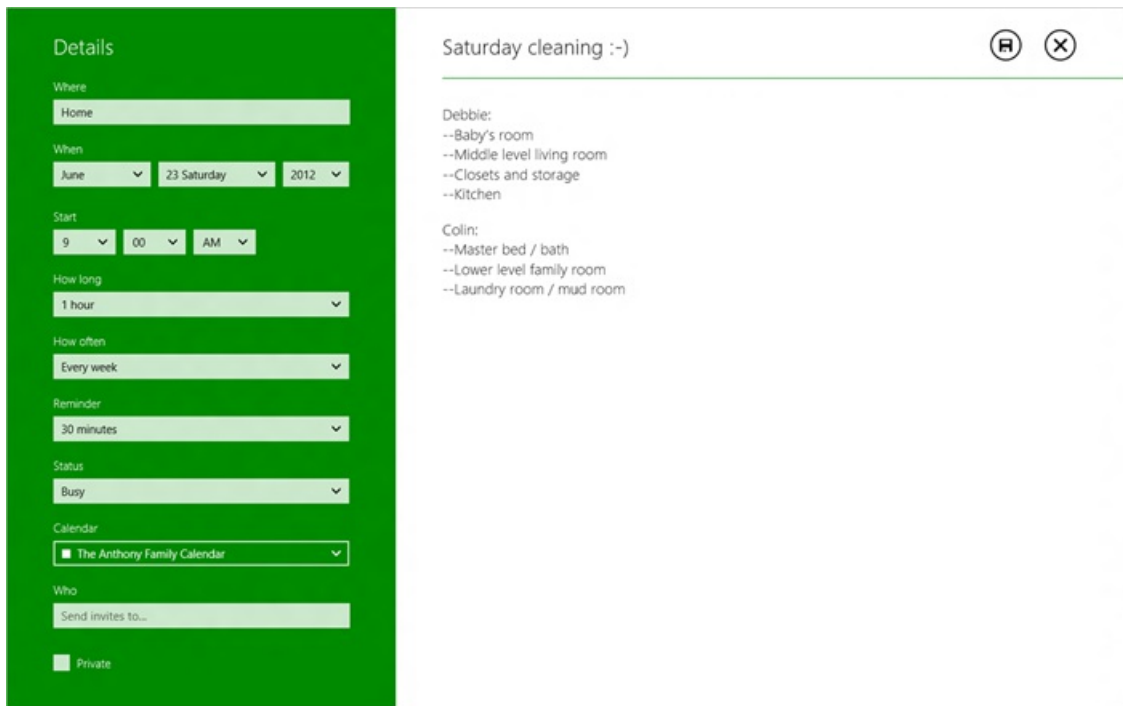
Doing more with events

Finally, we know many customers already use their calendars for quick reminders to themselves (e.g. "Pay the phone bill", "Parent/Teacher conference at noon"). As mentioned earlier, you can click or tap any place in the calendar, and simply type in the few details you need.

In addition to these typical scenarios, we designed Calendar so you can do more as your needs change—because depth and richness are a core part of what makes a digital calendar special.

For example, you may have a family trip coming up this summer. Not only can you add your trip to the calendar, but you can also add all the related information about your trip directly in the appointment—things like flight times, confirmation numbers, and day-by-day itineraries—no separate papers or messages to manage. It's all right there. You can also send the event to other members of your family, directly from the calendar, so everyone has the info they need.

As another example, on my family's calendar we have a recurring appointment that lists each person's cleaning responsibilities for Saturdays. On a paper calendar, this would be tedious to manage, or it would require a special calendar dedicated to that purpose. However, a dedicated page for each calendar event makes this level of richness possible in a single app.



The events page provides the options you need to manage your life

As you use Calendar in the Release Preview, we hope you enjoy it. The feedback we received during the Consumer Preview was extremely valuable, and we've looked at all of it very closely. Thanks! As we move towards the final release of Windows 8, we hope you'll enjoy the upcoming improvements as well.

--Colin

Introducing the Photos app for Windows 8

Steven Sinofsky | [2012-06-26T10:00:00+00:00](#)

Wrapping up our series of posts on some of the new apps in Windows 8, we take a look at the new Photos app. With this app, along with Metro style design principles, we set out to design an app that allows you to bring together photos from many different sources and to then view and share them. Brad Weed, a group program manager in the Windows Live team authored this post. --Steven

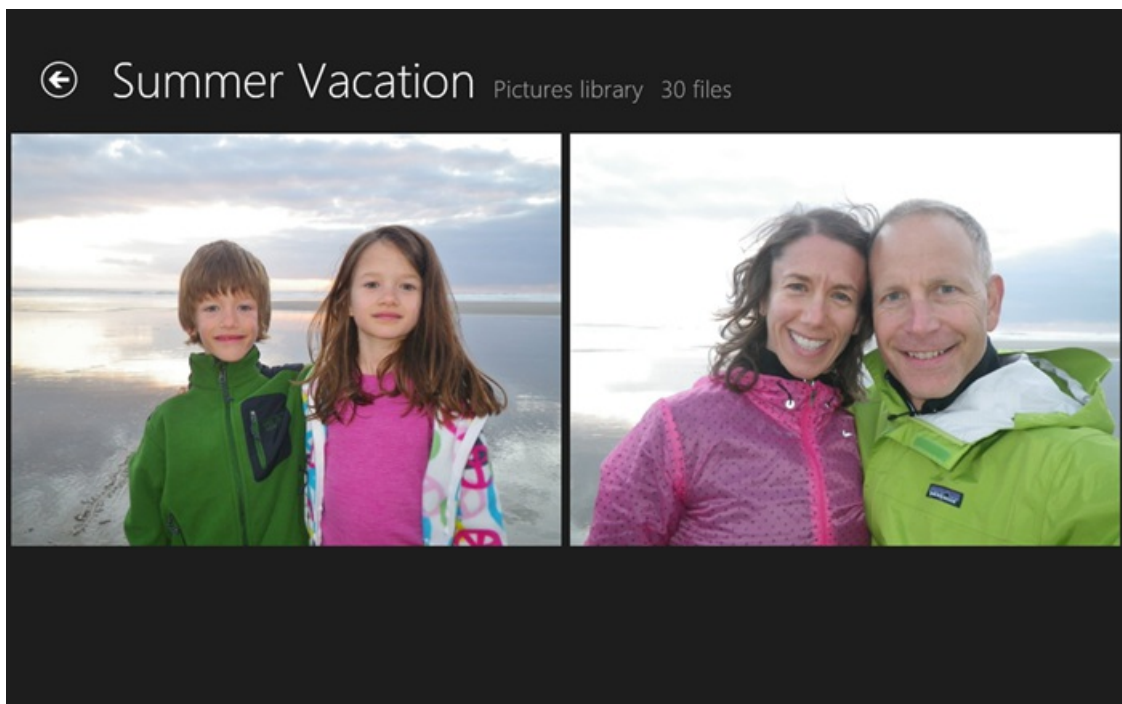
We take a lot of photos that end up in a lot of different places. Some are on our PCs, others end up on a photo sharing service like Flickr or Facebook, and even more are on our phones—sometimes indefinitely. How and where we store and share photos has changed and will continue to change as we take more photos, buy more devices, and share to more places with more people. What we need is one place where we can see, relive, enjoy, share, and immerse ourselves in all of them, all in one place.

All of your memories in one place

For the Windows 8 Consumer Preview we released an App Preview of the Photos app that introduced a new way to enjoy more of your photos. We realize the myriad places you have to go to see all of your photos, so we decided to bring them all to you in one place. Because you can [connect your Microsoft account](#) to services like Facebook and Flickr, you can get to all of your photos and all of those memories just by signing in to Windows 8 with your Microsoft account. Of course, the Photos app works best with our [SkyDrive](#) service, and with Windows Phone, you can automatically send all the pictures from your Phone to SkyDrive. This makes the Photos app in Windows 8 a great way to show off your photos without having to huddle around a phone. Even though you took your photos on your phone, you can easily enjoy them on your PC, just about as fast as you can take them.

Tell beautiful stories

Because Windows 8 is optimized for a landscape orientation, we designed the Photos app in the same way—to give you a view that tells the story best.



Our storyline view shows a photo just big enough to enjoy, but small enough to see more than one at a time. If you want to see more of your pictures at once, just pinch to zoom out, and you'll see a thumbnail view of your collection.



But notice: the thumbnail view isn't just all of your photos cropped and displayed as square thumbnails. We show your thumbnails in the way that best represents the orientation of the photo.

And of course, the best way to view a photo is in its full glory, so naturally, you can also view a single photo at a time.



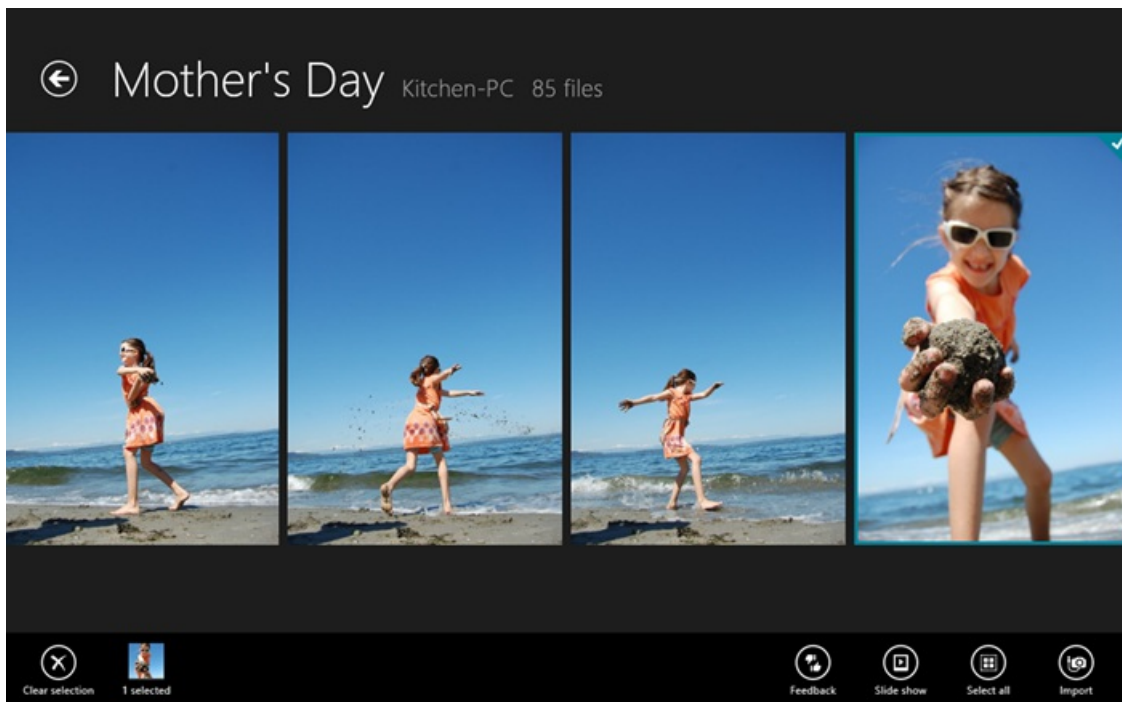
And if you don't want to swipe through your pictures to see them one or a few at a time, launch the slide show and sit back and enjoy. Better yet, use the Windows 8 Devices charm to play your slide show on your TV or any other Windows certified Play To device. Of course, this works for video as well, so you can also show off your latest Movie Maker creations this way, too.



New enhancements for Windows 8 Release Preview

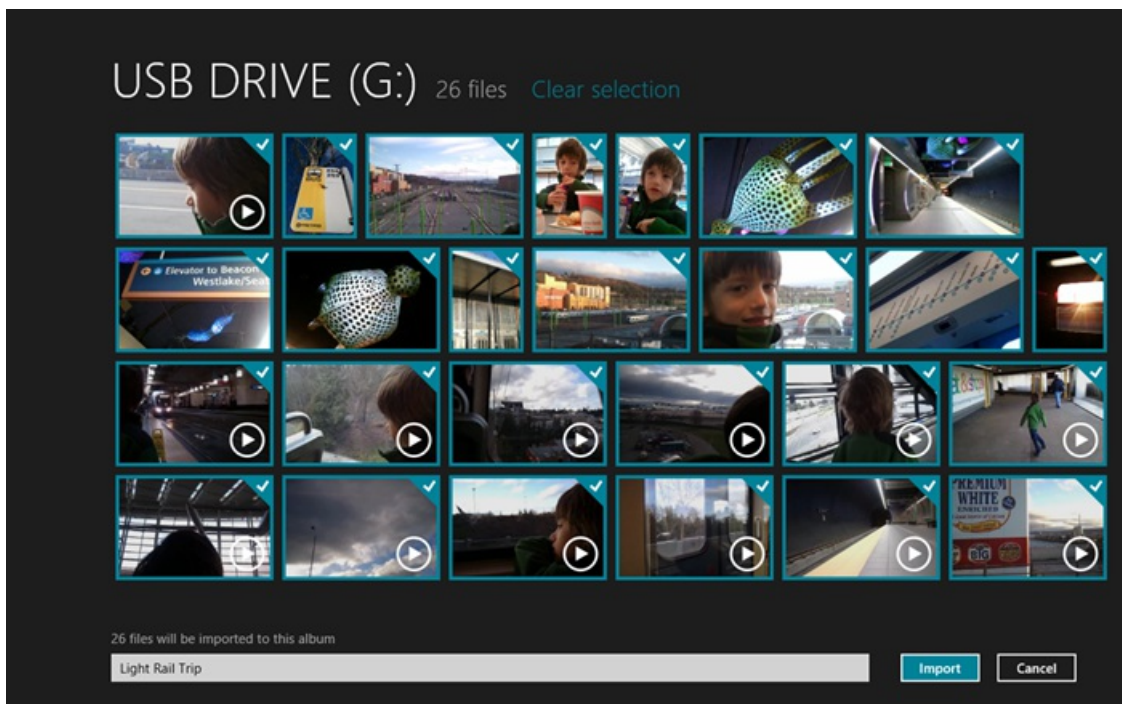
Photos from all of your devices

Since the Consumer Preview, we've been listening to the feedback and have been hard at work making improvements. We've heard from many of you that most of your photos aren't actually sitting on a web service, but they're on a PC somewhere in your home. And often they're on a PC that isn't convenient to get to or in a spot conducive to gathering your dinner guests around the monitor for "show and tell." So we've partnered with our friends in SkyDrive to make this a whole lot easier. Now, if you install the [SkyDrive desktop app](#), you can choose to have all of your photos automatically sent and saved to SkyDrive. Any PC with the SkyDrive desktop app installed will show up in the Photos app. So by simply running SkyDrive desktop app on the PC(s) where all your photos reside, the Photos app will reach back to that PC so that you can look at your old photos alongside your recent ones. You'll soon be revisiting photos you forgot you had.



Import

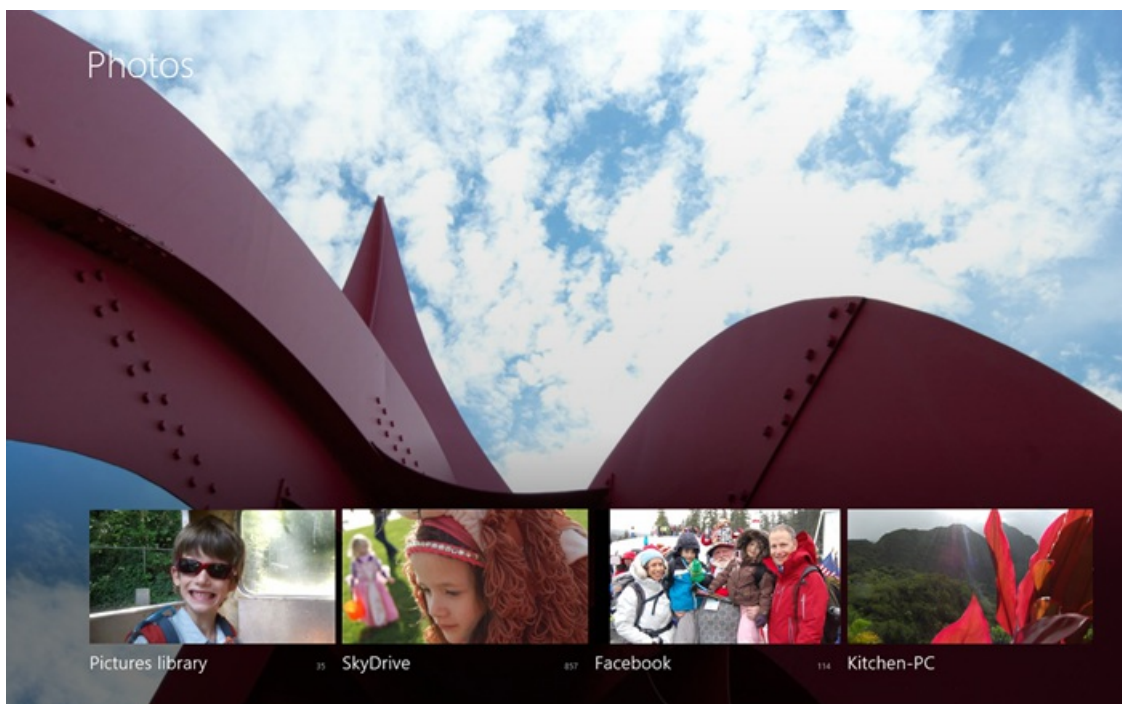
If you want to use your new Windows 8 PC as the primary place to store all of your photos, we've also added the ability to import with the Photos app.



When you plug your camera in to your PC, all you have to do is pick the Photos app as your importer, and we'll take care of the rest. You don't need any additional cables, and the quality of your photos will not be diminished. After you import your photos, if you don't have time to organize them, don't worry. All of your photos can always be viewed by date, so it's easy to browse through large collections of photos.

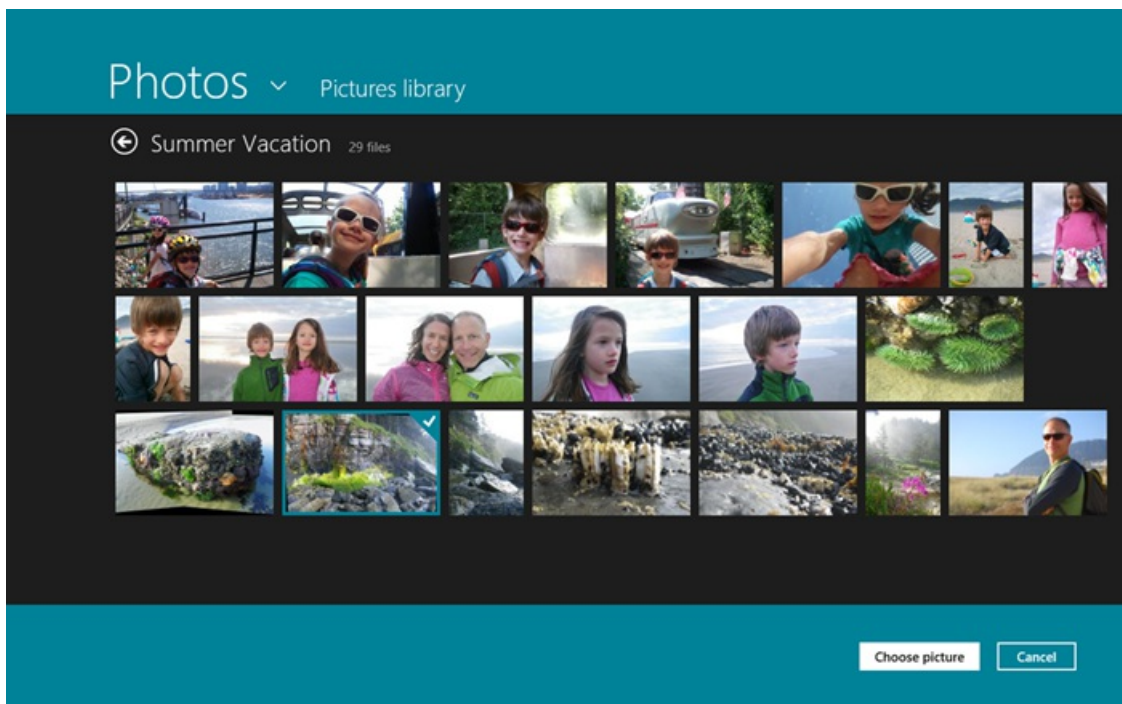
New home screen

The Photos app should scream "photos," so we've added an edge-to-edge photo that appears in the background on the home screen of the app. When you open the Photos app for the first time, you'll see a nice photo there, but you can change it to whatever photo you want, so it's one of your own. It is a personal computer after all.

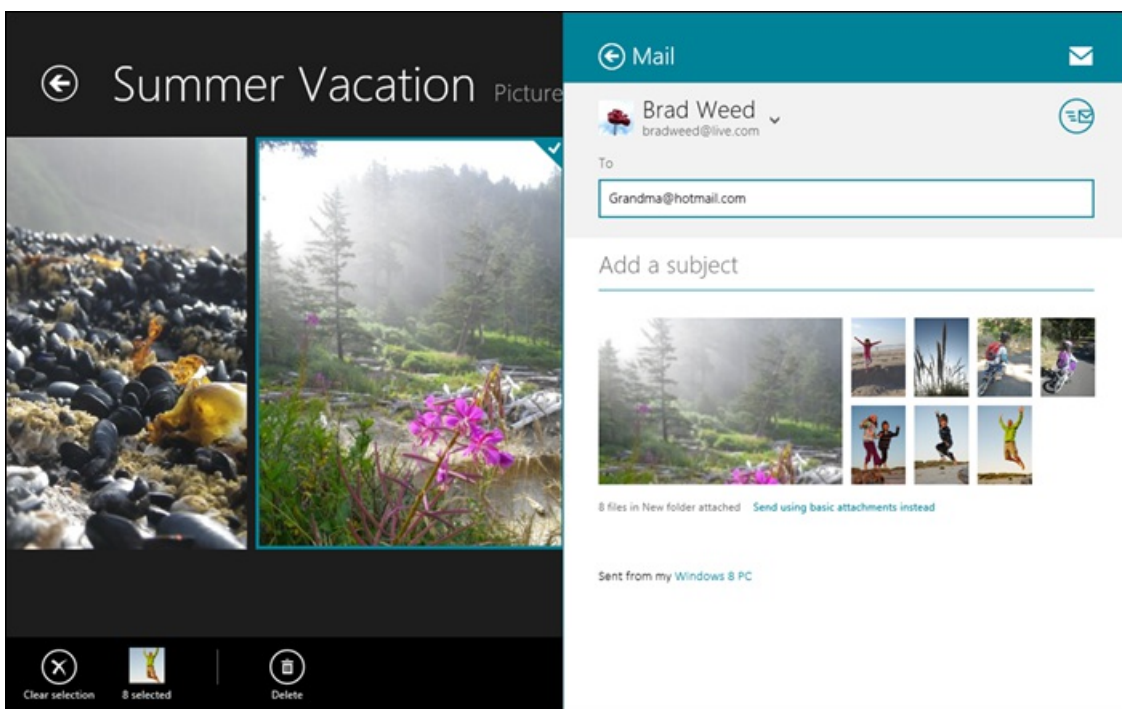


Sharing your memories

All of the rich views in the Photos app are especially helpful when you're ready to share or print your photos. Now, when you go to pick your favorite photo for your Windows 8 lock screen, or you want to share that perfect photo using the Share charm, you can grab a photo from Facebook, Flickr, SkyDrive, or any PC with the SkyDrive desktop app installed—all using the Photos app as a [picker](#).



Sometimes you stumble across a series of photos that you'd like to share while just browsing around. The Photos app makes it easy to select a set of photos, and use the Share charm to share in Mail, and off they go either as basic attachments or by sending a link to a slides show hosted on SkyDrive instead. Using SkyDrive allows you to send a lot of photos without having to worry about file size limitations.



Over the years we've all imported, shared, and saved photos to myriad SD cards, hard drives, and Internet services. It's rare that we ever go back and relive those memories, because they're in so many places that it's become too onerous. The Photos app lets you see the last photo you took on your Windows Phone, or the very first photo you shot with your very first digital camera. We've taken a lot of photos and videos in our lives and we're only going to take more. The Photos app not only brings all of those memories back, but it puts them in the palm of your hand.

Keep the feedback coming. We know there's a lot more to do. Until then, sit back and enjoy the photos you forgot you had.

Brad Weed
Group Program Manager, Windows Photos and Videos

Your browser doesn't support HTML5 video.
Download this video to view it in your favorite media player:
[High quality MP4](#) | [Lower quality MP4](#)

Readying Metro style apps for launch

Steven Sinofsky | [2012-07-03T10:30:00+00:00](#)

We know many folks are looking forward to RTM. Developers currently working on apps in the Store are especially excited. We have hundreds of apps in the Windows Store now and many more on the way. There's a broad set of developers around the world that we have been working closely with since the first Developer Preview. The WinRT platform is evolving rapidly during development based on feedback, and we have the dual task of keeping the Store up and running so we can supply apps to the millions of Preview users, while also getting ready for the next build. It means that if we change or add APIs or improve the tools, the apps will change and require an updated OS to test and verify the app. That's why we have been providing updated builds to developers who have or are committed to having apps in the Store through strong partnerships.

*This post explains the work we've been doing since September to keep developers updated with APIs and tools so that apps can stay up to date. We're doing this even after the Release Preview, just to make sure new apps are ready to go once we get to broad availability. **This post was authored by Dennis Flanagan, who leads our ecosystem outreach team.** --Steven*

As we approach the release of Windows 8, the catalog of Metro style apps continues to grow. To date, people have experienced apps that Microsoft has included with the downloaded build, and those that are offered in the Store in both the Consumer Preview and Release Preview timeframe. Many of those apps are great examples of immersive, touch-first Metro style experiences. However, like the Windows releases they run on, these apps are preview versions of the apps to come. The final versions of all Metro style apps will be available when Windows 8 becomes generally available.

Last year, we began working closely with the developer community by releasing early versions of the Windows 8 platform and tools. We decided to engage developers earlier in the engineering process so we could help them build skills in Metro style app development and give them the opportunity to influence the platform through feedback. Since September of 2011 we have released 8 developer preview versions. Some of these versions have been available to a limited developer audience. Some have been distributed broadly. All of these releases had similar goals:

- Deliver new capabilities and APIs
- Update tools to simplify Metro style app development
- Enhance performance and reliability
- Respond to developer feedback

We released our first Developer Preview version at the [//build conference](#) in Anaheim. This version introduced developers to the Windows 8 platform, tools and programming models. The WinRT platform included new APIs, and we used the conference to present literally hundreds of technical sessions and samples to give developers a basic understanding of the platform. Many developers got right to work building Metro style apps, produced some impressive early results, and provided us with useful feedback and recommendations about how to improve the platform and tools.

We made it clear that the first Developer Preview ("DP1") was an early version of the code, and we had a lot of work to complete Windows 8. DP4 and DP5, released in January and February of this year, were targeted at developers who wanted to be the first to publish applications in the Windows Store. By the time we released the Consumer Preview in February of 2012, we had added almost a thousand new WinRT APIs, and had modified hundreds of other APIs based on developer feedback.

For a detailed description of the changes that happened between //build and Consumer Preview, check out these posts on our [App Developer blog](#):

- [What's changed for app developers since //build/ \(part 1\)](#)
- [What's changed for app developers since //build/ \(part 2\)](#)

In April and May of this year, we released DP6 and DP7, which allowed developers to prepare their apps for the Release Preview. However, in close collaboration with the development community, we've continued to evolve the platform in response to their feedback. By the time we delivered the Release Preview, we had added 334 more APIs and continued to change existing APIs to address feedback.

One example of a change we made in Release Preview (RP) based on developer feedback is the HTML [ListView control](#) (in WinJS). This was an area where lots of developers had difficulties, so we overhauled it to make it easier to work with and to allow a much more extensive degree of performance tuning.

We also made lots of improvements to developer resources, such as templates in [Visual Studio](#). We even added a new template that makes it easier for developers to start a new project and get a great app up and running in very little time.

[Design tools](#) were another focus area for improvements. Metro is a design-forward experience, which means the app's user interface is one of the key ways developers get their apps noticed and differentiate them. We did a lot of work to make it as easy as possible for developers to integrate all the new Metro style design concepts into their apps.

For a complete overview of the changes between CP and RP, see [What's changed for app developers since the Consumer Preview](#).

Our next major milestone is the release to manufacturing (RTM). When the code reaches this milestone, the platform is complete for general availability (GA), and so we won't have interim updates for developers.

When Developers get the RTM version, they will continue enhancing the features, capabilities and performance of their apps. Some of the apps you've already seen will look and perform differently when you download the final released version. There are also many more apps in development that haven't been released to the Store yet. Many of those developers are waiting for RTM to put the finishing touches on their apps.

The release of Windows 8 will be a great milestone for app developers, but it is really just the beginning. A great benefit of the built-in Windows Store and update mechanism is that they provide developers with the opportunity to gain wide distribution for new apps and continuously improve apps that they've already released. As the app developer community evolves, we expect app developers to take advantage of this and provide regular updates to apps.

--Dennis

Protecting user files with File History

Steven Sinofsky | [2012-07-10T09:00:00+00:00](#)

*Backing up your critical files is something we all know we should do. Even with everything in SkyDrive, it is still something we need to do. With Windows 8, we took a new look at the way backup can work and set out to solve the perennial problem of not just restoring all your files but restoring a previous version of a critical file you have been editing through the course of a day. To achieve this, we're introducing a new feature in Windows 8, **File History**. **Bohdan Raciborski, a program manager on the Storage team authored this post.** --Steven*

Note: Comments have been off topic. Please maintain community standards and focus on the topic at hand.

What is File History?

File History is a backup application that continuously protects your personal files stored in Libraries, Desktop, Favorites, and Contacts folders. It periodically (by default every hour) scans the file system for changes and copies changed files to another location. Every time any of your personal files has changed, its copy will be stored on a dedicated, external storage device selected by you. Over time, File History builds a complete history of changes made to any personal file.

It's a feature introduced in Windows 8 that offers a new way to protect files for consumers. It supersedes the existing Windows Backup and Restore features of Windows 7.

What is unique about this approach compared to a more traditional backup and restore?

Regretfully, backup is not a very popular application. Our telemetry shows that less than 5% of consumer PCs use Windows Backup and even adding up all the third party tools in use, it is clear nowhere near half of consumer PCs are backed up. This leaves user's personal data and digital memories quite vulnerable as any accident can lead to data loss. In Windows 8 Microsoft is actively trying to accomplish the following:

1. Make data protection so easy that any Windows user can turn it on and feel confident that their personal files are protected.
2. Eliminate the complexity of setting up and using backup.
3. Turn backup into an automatic, silent service that does the hard work of protecting user files in the background without any user interaction.
4. Offer a very simple, engaging restore experience that makes finding, previewing and restoring versions of personal files much easier.

While designing File History we used learnings from the past and added requirements to address the changing needs of PC users.

- PC users are more mobile than ever. To address that, we optimized File History to better support laptops that constantly transition through power states or are being connected and disconnected from networks and devices.
- PC users create more data and are more dependent on it than ever before. So we do not only protect what's currently on the system drive but also any work they have done and data they have created in the past.

When a specific point in time (PIT) version of a file or even an entire folder is needed, you can quickly find it and restore it. The restore application was designed to offer engaging experience optimized for browsing, searching, previewing and restoring files.

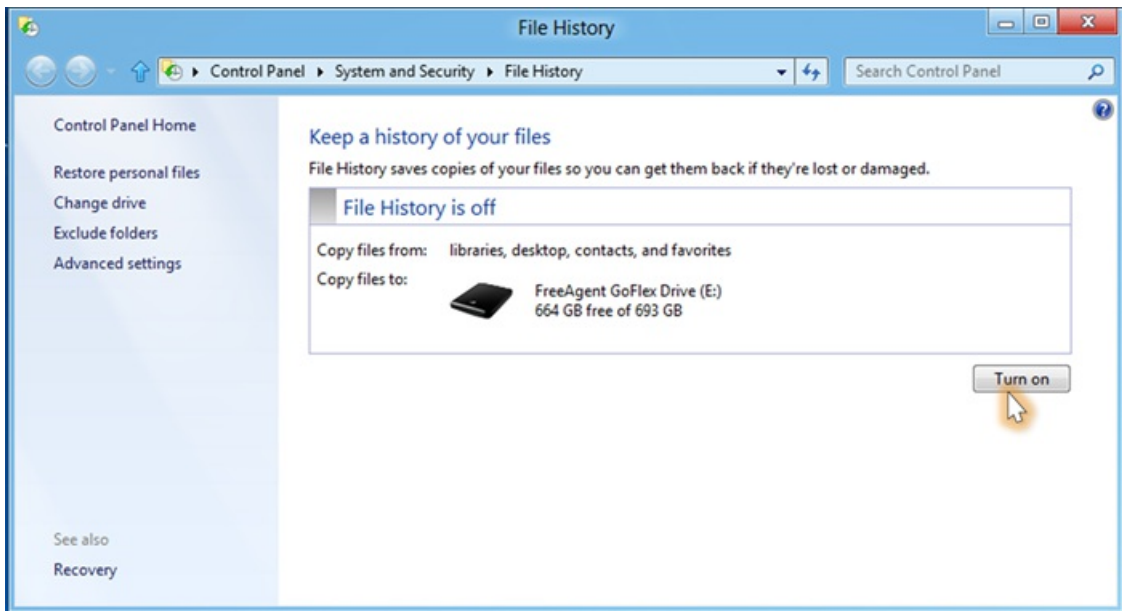
Setting it up

Before you start using File History to back up your files, you'll need to set up a drive to save files to. We recommend that you use an external drive or network location to help protect your files against a crash or other PC problem.

File History only saves copies of files that are in your libraries, contacts, favorites, and on your desktop. If you have folders elsewhere that you want backed up, you can add them to one of your existing libraries or create a new library.

To set up File History

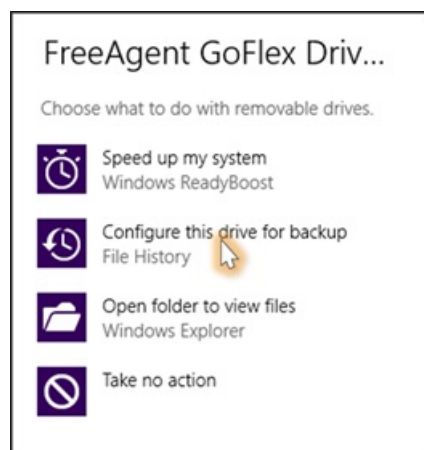
1. Open **File History** control panel applet.
2. Connect an external drive, refresh the page, and then tap or click **Turn on**.



You can also set up a drive in AutoPlay by connecting the drive to your PC, tapping or clicking the notification that appears...



... and then tapping or clicking **Configure this drive for backup**.



That's it. From that moment, every hour, File History will check your libraries, desktop, favorites and contacts for any changes. If it finds changed files, it will automatically copy them to the File History drive.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:
[High quality MP4](#) | [Lower quality MP4](#)

Restoring files

When something bad happens and one or more personal files are lost, the restore application makes it very easy to:

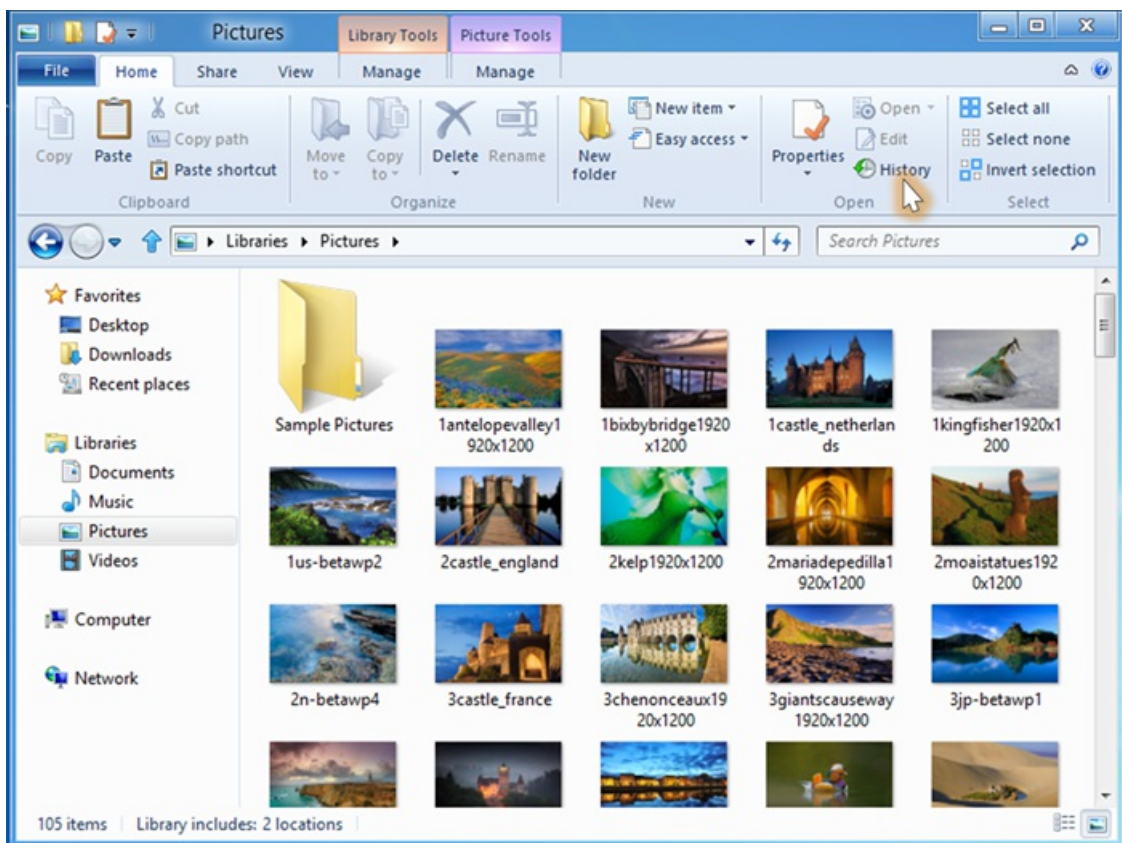
- Browse personal libraries, folders and files in a way very similar to Windows Explorer.
- Search for specific versions using keywords, file names and date ranges.
- Preview versions of a selected file.
- Restore a file or a selection of files with one tap or a click of a mouse.

We designed the restore application for wide screen displays and to offer a unique, engaging and convenient way of finding a specific version of a file by looking at its preview.

With other backup applications you would have to select a backup set that was created on a specific date. Then you would have to browse to find a specific folder, and then find the one file you need. However at this point it is impossible to open the file or preview its content in order to determine if it is the right one. You would have to restore the file. If it is not the right version, you'd have to start over.

With File History, the search starts right in Windows Explorer. You can browse to a specific location and click or tap on the History button in the explorer ribbon in order to see all versions of the selected library, folder or an individual file.

For example, when you select a Pictures library and click or tap on the History button...



... you will see the entire history of this library.



When you click on a specific file, you can see the entire history of the selected picture.



In this example, the selected picture has 4 versions. You can easily navigate to the desired version by clicking on the Previous/Next buttons or by swiping the screen. Once you have found the version you were looking for, you can click the Restore button to bring it back. The selected version will be restored to its original location.

Continuous, reliable protection

File History, instead of using the old backup model, takes a different approach to data protection.

Protect only what is most important

Instead of protecting the entire system (operating system, applications, settings and user files) File History focuses only on user personal files. That's what is most precious and hardest to recreate in case of an accident.

Optimized for performance

In the past, most backup applications used brute force method of checking for changes in directories or files by scanning the entire volume. This approach could significantly affect the system performance and requires an extended period of time to complete. File History, on the other hand, takes advantage of the NTFS change journal. The NTFS change journal records any changes made to any files stored on an NTFS volume. Instead of scanning the volume, which involves opening and reading directories, File History opens the NTFS change journal and quickly scans it for any changes. Based on this information it creates a list of files that have changed and need to be copied. The process is very quick and efficient.

File History was designed to be easily interrupted and to quickly resume. This way, File History can resume its operation, without the need to start over when a system goes into sleep mode, a user logs off, the system gets too busy and needs more CPU cycles to complete foreground operations, or the network connection is lost or saturated.

File History was designed to work well on any PC including small form factor PCs with limited resources and tablets. It uses system resources in a way to minimize the impact on system performance, battery life and overall experience.

1. File History process runs at low priority, uses low priority IO and low priority memory.

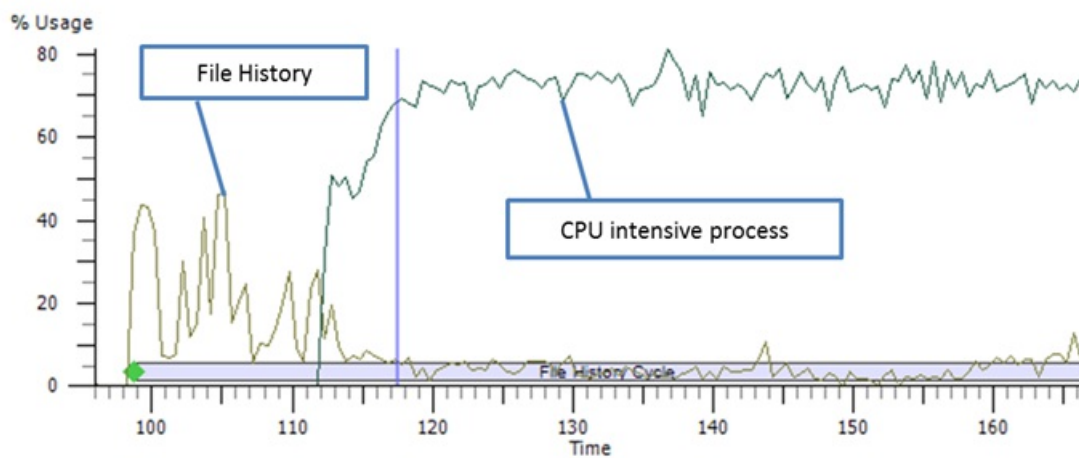


Figure 1: File History reaction to an increasing workload.

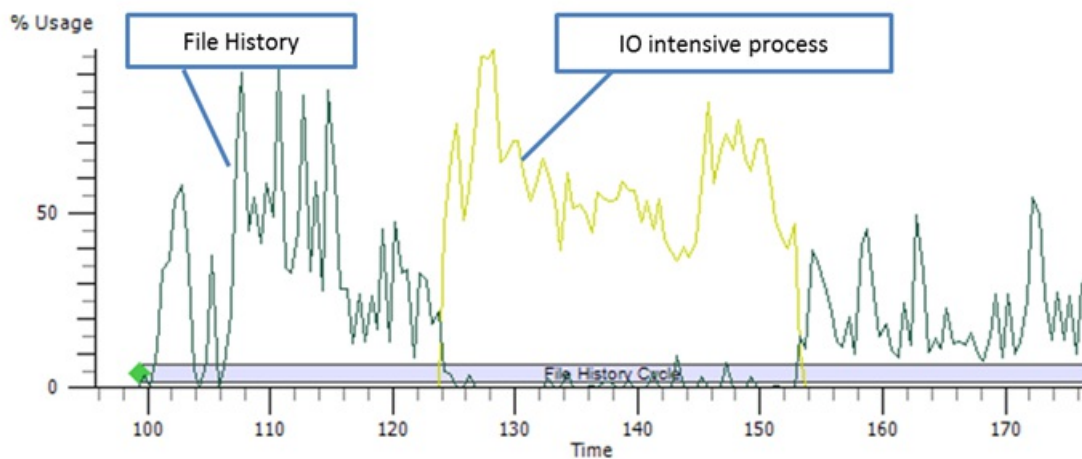


Figure 2: File History disk utilization in presence of other processes with normal priority I/Os.

1. When File History service is idle, it uses an average of 10 MB of working set. When it runs, it uses little memory and only for a short period of time. The chart below shows the working set histogram collected using a simulated workload over a period of 4 hours with File History cycle frequency set to one hour.



Figure 3 Working set size per backup cycle, sampling every 3 min.

1. The amount of data being backed up in one cycle is throttled.
2. Any activity is suspended when the machine is sleeping.

File History takes into account:

- If the user is present, i.e. logged on and actively using the system.
- If the machine is on AC or battery power.
- When the last backup cycle was completed.
- How many changes have been made since the last cycle.
- Activity of foreground processes.

Based on all of these factors, which are re-checked every 10 seconds, it determines the optimal way to back up your data. If any of those conditions change, the service makes a decision to reduce/increase quota or suspend/terminate the backup cycle.

Optimized for mobile users

When File History is running, it gracefully handles state transitions. For example, when you close the lid of your laptop, disconnect an external drive or leave home and take your laptop out of the range of the home wireless network, File History takes the right action:

- Lid closed - When a PC goes into sleep mode, File History detects the power mode transition and suspends its operation.
- Lid opened – File History resumes its operation at a priority that makes sure files are protected without impacting overall system performance, even for gamers. It also waits for all post “lid open” activities to complete so that we do not affect the system while it is coming back out of sleep.
- Dedicated storage device disconnected – File History detects that the storage device is not present and starts caching versions of changed files on a system drive.
- Dedicated storage device re-connected – in the next cycle, File History detects that the storage device was reconnected, flushes all versions from the local cache to the external drive and resumes normal operation.

Simplicity and peace of mind

We designed File History with two objectives in mind; 1) offer best possible protection of user personal files and 2) offer ease, simplicity and peace of mind.

If you want to take advantage of File History, you have to make only few, simple decisions. In most cases it will be limited to only one – which external drive to use. The rest is taken care of by Windows. The operation of File History is transparent and doesn't affect the user experience, reliability or performance of Windows in any way.

Full control

Most backup applications, including the Windows Backup and Restore that shipped in Windows 7 require administrator privileges to set up and use. This means that standard users have to ask the administrator to set it up and every time they need to restore a file, or to grant them administrative privileges. Not so with File History. File History offers full control to each individual user. Now users can decide if and when to turn File History on and which external drive to use. In fact, each user can select a different location to store their file history. And they do not have to ask for the administrator's help to restore a file.

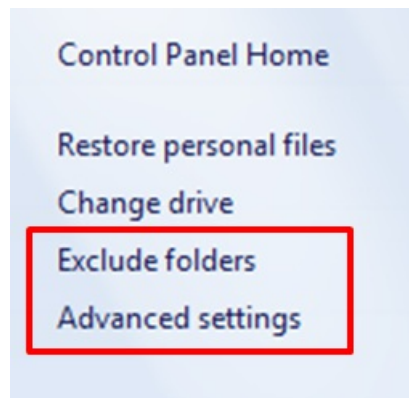
Enthusiasts and experienced PC users can use advanced File History features to control many aspects of its operation, like:

- **How often you want to save copies of your files:** The frequency of backups can be changed from 10 minutes to 24 hours. Higher frequency offers better protection but consumes more disk space.
- **How long you want to keep saved versions:** Versions can be stored forever or as little as one month. This setting is useful when the File

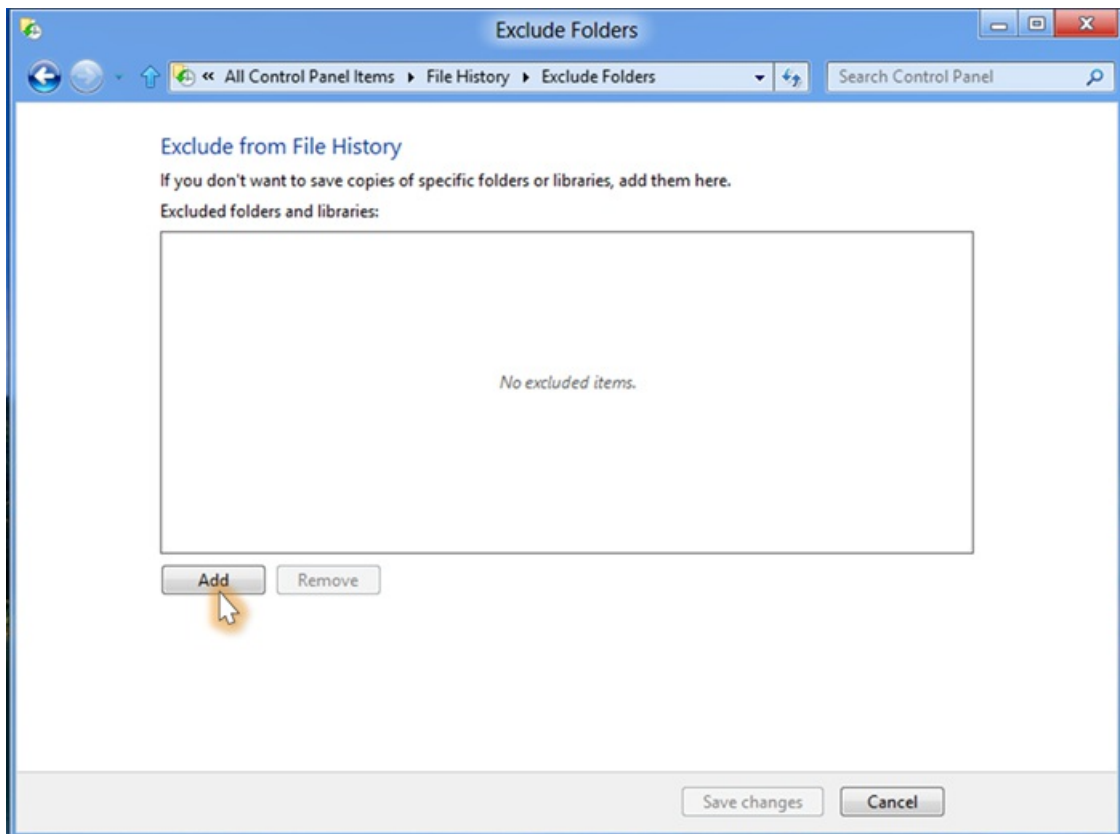
History drive fills up to fast. You can slow down this rate by reducing the time versions are stored.

- **Changing the size of the local cache:** File History uses a small amount of space on the local drive to store versions of files while the File History target drive is not available. If you create a lot of versions of files while disconnected or stay disconnected for longer periods of time, you may need to reserve more space on the local drive to keep all versions. Note that the versions stored in the local cache are flushed to the external drive when it becomes available again.
- **Excluding folders that you do not want to back up:** Some folders may contain very large files that do not have to be protected because they can be easily recreated (like downloaded high definition movies or podcasts). These files would quickly consume all of the File History drive capacity. This setting allows you to exclude such folders.
- **Recommend a drive to other HomeGroup members on your home network:** This setting is covered in more detail in the File History and HomeGroup section below.
- **Accessing the File History event log:** The event log contains records of events that may be useful while troubleshooting File History. It may be particularly useful if you want to identify files that File History could not access for any reason.

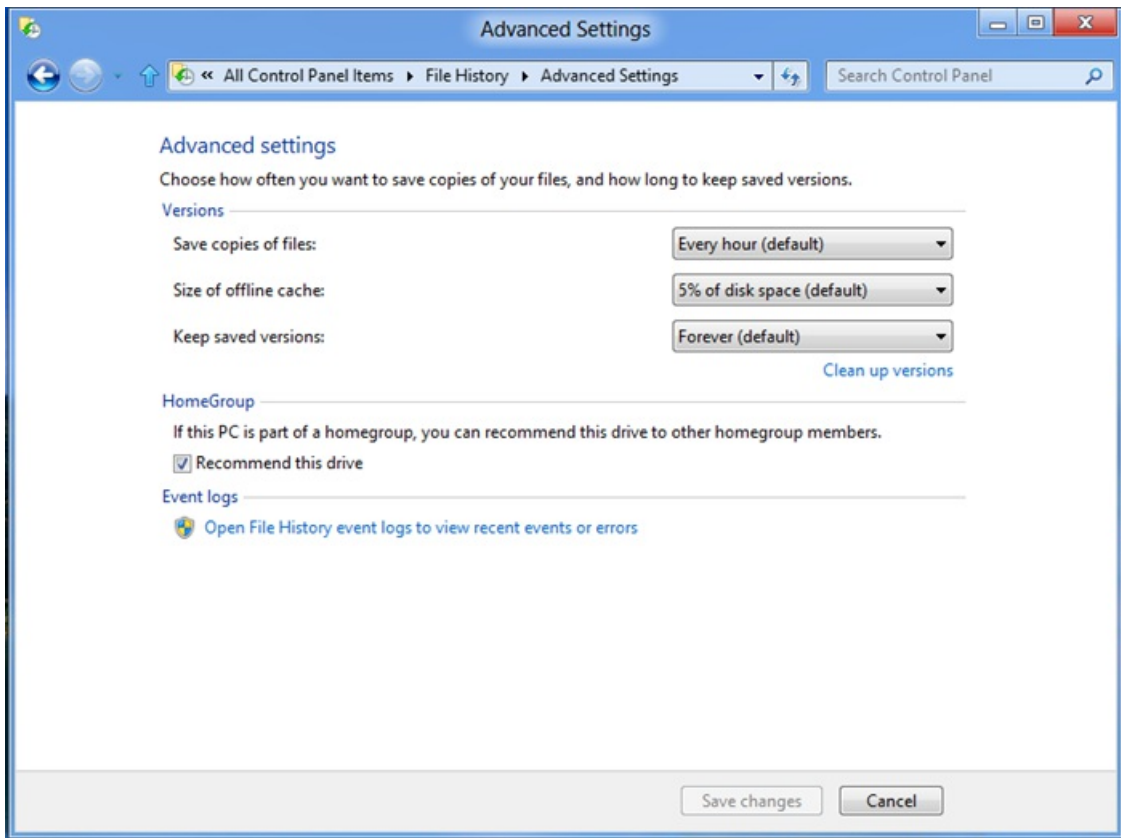
Advanced settings can be accessed from the File History control panel applet.



To exclude a folder, select **Exclude folders**. Next, click on the **Add** button, browse to the folder you want to exclude and select it. Files in this folder will not be backed up starting with the next backup cycle. To start backing it up again, simply remove the folder from the list.



Other advanced settings are available on the **Advanced Settings** page.

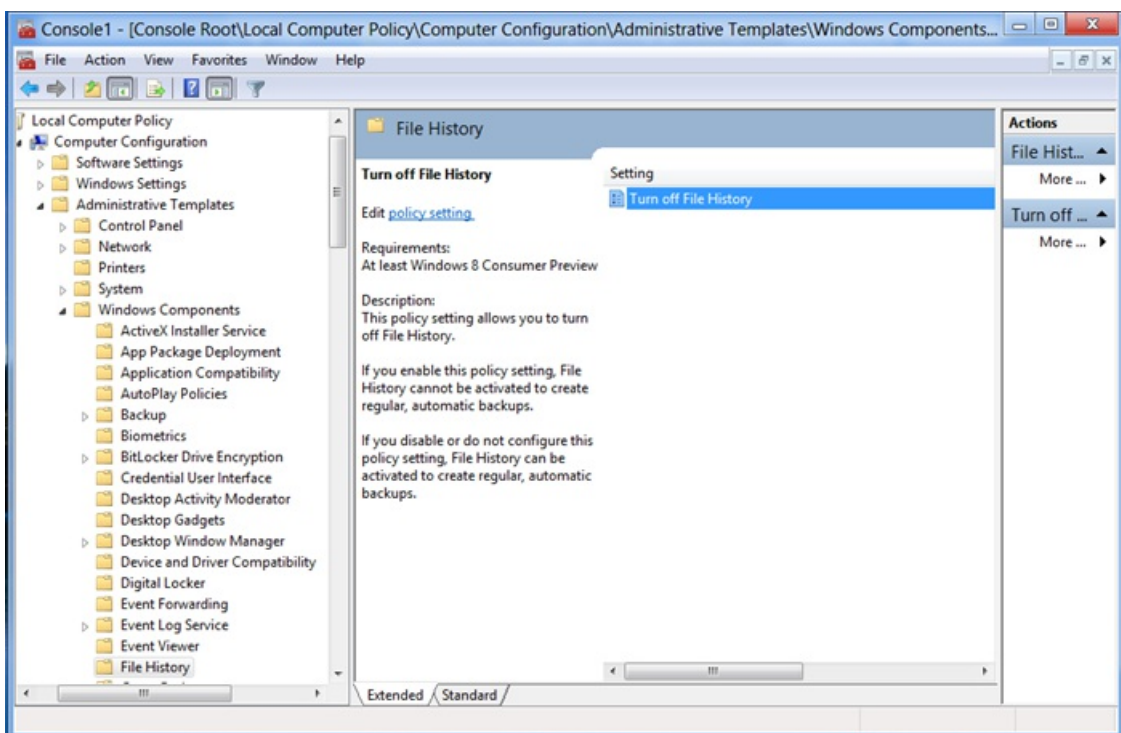


File History also supports new storage features introduced in Windows 8. Users who have lots of data to back up can use **Storage Spaces** to create a resilient storage pool using off-the-shelf USB drives. When the pool fills up, they can easily add more drives and extra storage capacity to the pool. You can find more about Storage Spaces in [this blog post](#).

Users who use **BitLocker** to protect the content of their personal files can also use File History as it seamlessly supports **BitLocker** on both source and destination drives.

File History was designed for consumers but could also be used by enterprise customers. In some cases, File History may conflict with the enterprise policies (like retention policy). To prevent such conflicts, we added a group policy that gives enterprise IT administrators the ability to turn off File History on managed client PCs.

You will find the File History policy setting in the Group Policy Object Editor under Computer Configuration, Administrative Templates, Windows Components, File History.



Minimal setup

File History is part of Windows so you don't need to install any additional software. However, File History has to be turned on, which typically requires only one click.

As described above, to start protecting your libraries, you need to attach an external drive or select a network location. File History will store versions of your files on this device.

File History automatically selects an external drive if one is available. If more than one drive is available, one with the most free storage capacity is selected.

No schedule

File History wakes up once an hour and looks for personal files that have changed. Versions of all files that have changed are replicated to a dedicated storage device. This approach eliminates the need to set up a schedule and leave a computer idle for an extended period of time. One hour frequency offers a good balance between the level of protection and amount of storage space consumed by file versions. Enthusiasts can change the frequency from 10 min to 1 day in order to increase the level of protection or reduce storage consumption.

No maintenance

File History runs silently in the background and doesn't require any ongoing maintenance. The only time when it will ask you to intervene is when the external drive is full. At this point you will be asked to either replace the drive with a bigger one or change a setting that tells File History how long to keep file versions around. By default, we keep versions of user personal files forever, but if storage is an issue, it can be reduced to a period of time that best suits your needs.

File History and HomeGroup

File History was also integrated with HomeGroup to make it easier for someone to set up backup for all members of a home network. Here is how it works.

1. Jane wants her entire family to have their personal data automatically protected. She knows she can do this with File History.
2. Jane creates a HomeGroup on the family's home network.
3. Jane turns on File History on a computer that has a large external drive.
4. File History control panel detects the HomeGroup and asks if Jane wants to recommend this backup destination to other HomeGroup members.
5. Jane selects this option and File History uses HomeGroup to broadcast the recommendation to all HomeGroup members.
6. Each HomeGroup member can now accept the recommendation. If they do, their libraries, desktop, favorites and contacts are automatically backed up to a network share on Jane's computer.

File History and SkyDrive

File History doesn't back up your files to the cloud. While the cloud is great for storing files you'd like to access on-the-go, or for sharing files with others, backing up terabytes of data to the cloud requires a specialized service. Many cloud services today support local synchronization, where the data in the cloud is mirrored in your local file system. Sync solutions by their very nature copy changes immediately to all locations, which means accidental deletes or inadvertent changes or corruption to files will be synchronized as well. The best way to address this problem is to couple your sync service with a point-in-time backup solution like File History.

In the blog post, [Connecting your apps, files, PCs and devices to the cloud with SkyDrive and Windows 8](#) we discussed how SkyDrive will integrate with Windows Explorer and the file system. File History takes advantage of that integration. If your SkyDrive is synced to your file system, File History will automatically start protecting the files stored in your local SkyDrive folder. This is a great example of local backup plus reliable anytime, anywhere access. You can access your files in SkyDrive through your PC, your phone, or the web and you'll also know that File History is providing fast local backup and instantaneous access to all versions of those files.

Full system backup

Usually a full system backup is used to protect your PC against complete system loss, for example when a PC was stolen or lost or the internal hard drive stopped working. Our research showed that only a small number of users are concerned about losing the operating system, applications or settings. They are by far more concerned about losing their personal files. For these reasons, File History was designed specifically to protect user personal files.

File History doesn't offer the ability to do a full system backup but for those users who may need a full system backup it offers a good compromise. Together with other features introduced in Windows 8 it provides protection against such disasters.

If you want to prepare for a disaster, we recommend a following strategy:

1. Create a recovery drive to be used when you need to refresh or restore your PC. You can find more about it in [this blog post](#).
2. Connect to your Microsoft account
3. Configure your PC to sync your settings

4. Load apps from the Store
5. Turn on File History

When your PC is replaced or needs to be reinstalled:

1. Use the recovery drive to restore the operating system
2. Connect to your Microsoft account
3. Configure your PC to sync your settings – this will bring your settings back
4. Go to the Store and reinstall your modern apps
5. Reinstall legacy apps
6. Connect your old File History drive and restore everything – this will restore your personal files

It may require more steps than a file or image restore but has some clear benefits:

- You do not restore any “no more desired” software or settings that were on your system
- You do not restore sources of some problems that you might have (or create new problems if you restore to different hardware)
- You do not restore settings that may cause your system to perform badly or fail

Those who need a full system backup can still use Windows Backup to create a system image.

Requirements

File History requires:

- Windows 8 Client operating system
- An external storage device with enough storage capacity to store a copy of all user libraries, such as a USB drive, Network Attached Storage device, or share on another PC in the home network.

FAQ

What happens when you upgrade to Windows 8 from Windows 7?

If Windows 7 Backup was active, i.e. it was scheduled and the schedule was active, then it will continue running as scheduled after the upgrade. File History will be disabled by default and users will not be able to turn it on as long as the Windows 7 Backup schedule is active. To turn it you will have to first disable the Windows 7 Backup schedule.

Can Windows 7 users use File History?

Windows 7 users cannot use File History. However, they can restore files from a drive used by File History by browsing the volume in the Windows Explorer and selecting a specific file. Files on the File History drive are stored in the same relative location, and use the same name. The specific version can be identified by the time stamp appended to the file name.

Does File History protect the operating system and applications?

File History only protects user libraries, desktop, favorites and contacts. Other files, such as operating system files, applications, and settings, are not backed up.

Can File History be used with cloud storage?

No. File History is designed specifically for consumers and does not support cloud storage in this release. Windows 8 Server offers a backup feature that can back up files to a cloud. This feature is available on the Server version of Windows and is designed for small and medium businesses.

Can File History be used by enterprise customers?

Yes. However, enterprise customers should be aware that File History may not comply with their company security, access, and retention policies. For that reason, we offer a group policy setting that allows enterprise administrators to disable the feature for an entire organization.

Will File History protect files stored on a file share?

No. File History only protects file stored on a local drive.

- If you use offline folders and folder redirection, your folders (like My Documents or My Pictures) are redirected to a network share and will not be protected.
- If you add a network location to any of your libraries, this location will not be protected.

In closing

File History silently protects all of your important files stored in Libraries, Desktop, Favorites and Contacts. Once turned on, it requires no effort at all to protect your data. When you lose a file or just need to find an original version of a picture or a specific version of a resume, all versions of your files are available. With the File History restore application you can find it quickly and effortlessly.

--Bohdan Raciborski

Designing the Windows 8 touch keyboard

Steven Sinofsky | [2012-07-17T08:00:00+00:00](#)

Starting with the earliest TabletPC enhancements to Windows, we have been working on “on-screen keyboards.” With Windows 8, we started fresh and took a “first principles” approach to developing the touch keyboard. Given the amount of experience many of us have with touch keyboards for phones, and the myriad of touch devices we interact with these days, we set a very high bar for the quality of the experience and effectiveness of input with the new Windows 8 touch keyboard. In this post, Kip Knox, a member of the Windows User Experience program management team, details this work. – Steven

When we began planning how touch and new types of PCs might work on Windows 8, we recognized the need to provide an effective method for text entry on tablets and other touch screen PCs. Since Windows XP SP1, which had Tablet PC features built in, Windows has included a touchable on-screen keyboard. But those features were designed as extensions to the desktop experience. For Windows 8, we set out to improve on that model and introduce text input support that meets people’s needs, matches our design principles, and works well with the form factors we see today and expect to see in the future.

I’m writing this blog post on our Windows 8 touch keyboard using the standard QWERTY layout in English. As I look at it, the keyboard seems very simple and sort of obvious. This comes partly from having worked on it for a while, but also because keyboards are familiar to us. But there is more here than meets the eye (or, fingertips).

We started planning this feature area with no preconceived notions. As we do with all our features, we began the text input design project with a set of principles or goals. On a Windows 8 PC using touch, we want people to be able to:

- Enter text quickly, reasonably close to the speed with which they type on a physical keyboard
- Avoid errors, and be able to easily correct mistakes
- Enter text comfortably, in terms of posture, interaction with the device, and social setting

You might note that none of those goals explicitly assumes a keyboard. And when we started the project, we cast a broad net across possible approaches to text input. We found that of all the methods of text input we considered, none met the goals above as well as a keyboard. The majority of people are simply faster, more accurate, and more comfortable typing than they are writing any other way. Windows has highly accurate handwriting recognition in several languages, as well as advanced speech recognition, for example. But without a great touch keyboard, we were not going to be able to fulfill people’s needs and expectations for touch-screen devices running Windows. So we set out to create the best touch keyboard on any device.

Optimizing for comfort and posture

There are many ways to imagine touch keyboards on a tablet, and we sketched a lot of them—large keyboards, tiny keyboards, floating keyboards, circular keyboards, swipe keyboards. But our initial design process was grounded in research we did into the ways that people interact with tablets. Our researchers conducted an in-depth study in which they observed people “living with” tablets over a period of time. Through these observations and interviews, we saw a set of three postures that are most common among people using tablets:

1. One hand holding the device, with one hand interacting with the user interface
2. Two hands holding the device, with thumbs interacting
3. Resting the device on table, lap, or stand, and interacting with both hands



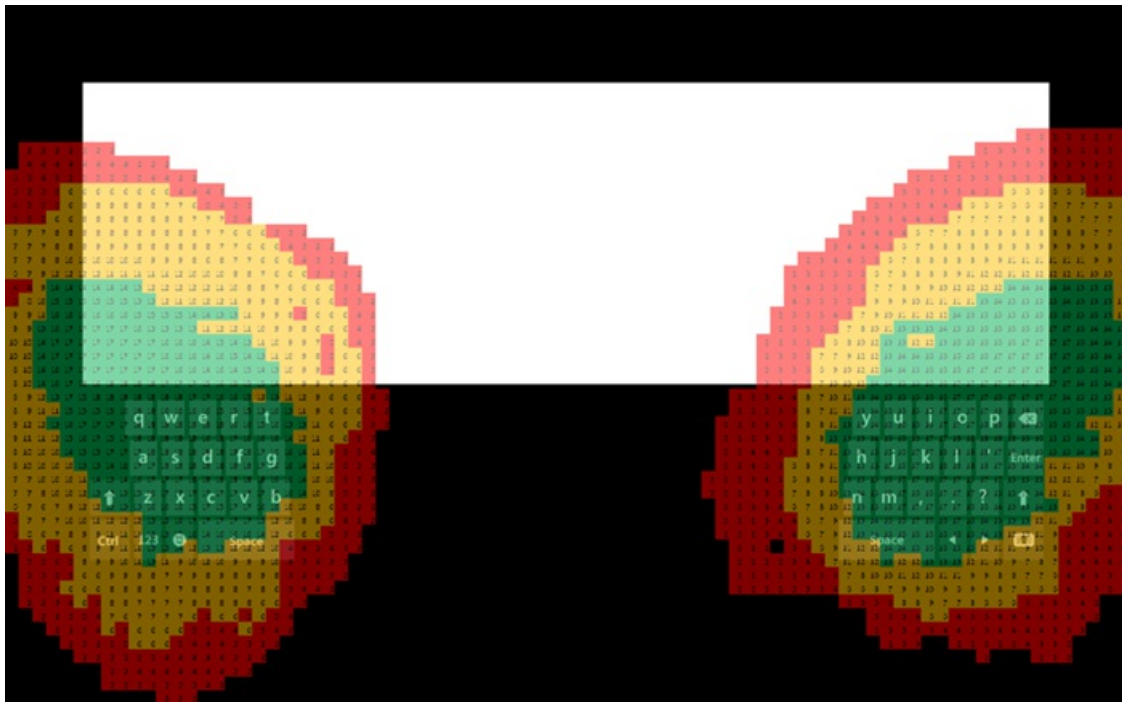
Research into people “living with” tablets revealed three common postures.

In these postures, people felt most natural and most likely to use the tablet for longer periods of time. We’ve made many design decisions in Windows 8 to optimize for these postures, and that includes how people intuitively input text. When typing on a tablet, most people either set it on their lap or a table and multi-finger type, or hold it in their hands and type with their thumbs, or hold it with one hand and “hunt and peck.”

Our standard touch keyboard layout is optimized for laying the tablet down and multi-finger typing, and also works well for typing with one hand. We also introduced a new layout we call the thumb keyboard (which we showed for the first time at our very first preview of Windows 8 about a year ago), which is designed for holding the tablet with two hands and typing with your thumbs. This keyboard is adjustable in size, to accommodate different hand sizes. An interesting observation from our posture research is that people frequently switch postures, and that posture

switch is often seen as a positive thing, as we move about to remain comfortable. So in our keyboard layouts we also considered what it would be like to type for a period of time—say, an email to your mom—and switch postures while you do it. You might start by typing with the tablet lying on the coffee table, for example, but then you might tire of that posture and pick up the tablet, lie back on the couch, and interact with two thumbs.

Further research into posture and comfort helped us to understand how people hold tablets, and how far our thumbs typically reach. In a follow-up study, we had a wide selection of people with different hand sizes use a tablet with sensors that would indicate where their thumbs could reach most comfortably, where they could extend to, and where reach was just uncomfortable. These results helped us optimize the use of the system with thumbs, and helped shape the thumb keyboard layout.



This heat map illustrates the typical reach of people's thumbs, overlaid on the thumb keyboard layout. Green is very comfortable, yellow can be reached, and red is typically uncomfortable.

Typing on glass

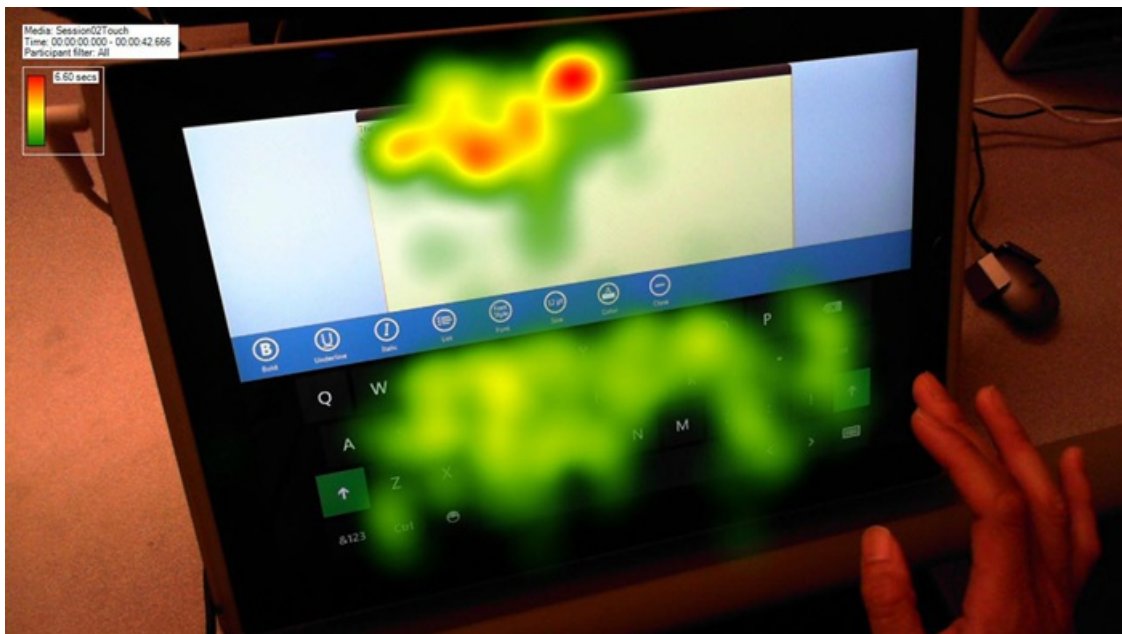
The next challenge we considered was the experience of typing on the glass display of a tablet. At least one of the key postures—laying the tablet down—is analogous to typing on a physical keyboard. So unlike typing text on a phone, we were faced with direct comparisons with the physical keyboard experience. When you type on your laptop or desktop, you enjoy some real benefits. You get a lot of sensory feedback as you type. First, you can position your hands quickly on your home keys, and most keyboards have small bumps on the J and F keys (in English QWERTY keyboards) to confirm that position. Then, as you type, the shape of the keys reinforces where your fingers are as they move about. The keys have “travel,” or small up-and-down movement, which confirms that you struck them. And because the keyboard is mechanical, there is a tapping sound that confirms your key strikes (perhaps to your chagrin, if your colleagues are checking email during meetings J).

If you lay down a piece of glass and type on it, you get no feedback; there is no indication for where to position your hands, and there is no indication of whether you've hit a target or not. Recognizing this, we made a few decisions. We needed to provide some type of feedback, and we needed to recognize that people will be more “sloppy” when typing on a touch keyboard. But we also observed that a touch keyboard can do things that a physical keyboard can't, and we should bring those functions out.

The feedback you see in the touch keyboard comes in two forms—the keys change color when you touch them, and they trigger a subtle sound. This is similar to what you see on most phone touch keyboards. We considered other forms of feedback, but ruled them out as too disruptive or unnatural. For example, we explored haptic feedback (a vibration of the device based on input) which you also find on many phones. But most people find the current state-of-the-art haptics somewhat irritating when typing pieces of any length and a buzz can feel as much like a punishment as a reassurance.

Our two forms of feedback—visual key changes and sounds—are not without controversy either. Visual key changes are not always ideal when you are entering a password, for example, and for that reason we enable you to suppress feedback in these cases. Some people have argued that key press sounds are irritating and artificial. But user testing confirmed our assumption that people clearly find the sounds reassuring and confidence-inspiring when typing on glass. The specific sounds we use (which are very similar to those on the Windows Phone) are designed to be “residual,” where you quickly forget that they are there, but would notice if they were turned off.

Both forms of feedback may be used more when people are first getting used to the experience. We have done eye-tracking studies in the lab, which showed that as people become more proficient with the touch keyboard, they spend more time looking at the input field, and less time looking at the keyboard itself. So the appearance of each character becomes the best feedback when you are typing efficiently. I'll tell you a little more about these eye-tracking studies later in this post.

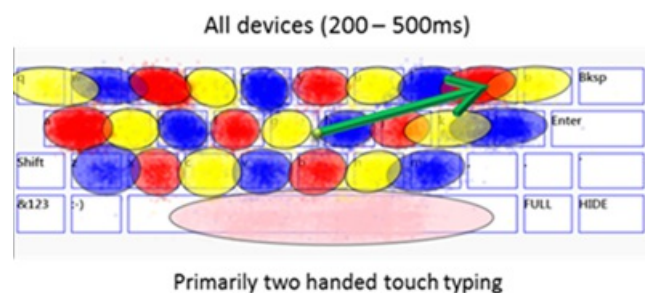


As people spend time with the touch keyboard, their focus moves more consistently to the input field, as this heat map from an eye-tracking study shows.

But even when you “get good” at typing on a touch keyboard on glass, you will still be sloppier and slower than you would be with a physical keyboard. The Windows 8 touch keyboard has some special accommodations to address this reality. The most interesting one is what we call the “touch model.”

When you tap a key on the touch keyboard, we detect the coordinates of your touch, and we can map it to the geometry of the keys. But as your fingers move about across the glass, your press is likely to migrate outside the boundaries of the key you intended to touch. If we relied simply on the geometry mapping of the keys, you would see a lot of errors. To account for this, the key press is first compared against a model that assesses the likelihood that you intended to strike that key or a key near it. This processing is informed by two things. First, we use data from many people’s typing *pangrams*, or phrases that use every letter of the alphabet, recording trends where peoples bias their touch away from the intended target. For example, they might intend to type a **p**, but often strike the **o**, because most people’s fingers curve inward. Based on a set of characteristics, including typing speed, the model weights the likelihood that you intended to type one key over another. Secondly, we use lexical data representing letters and words that are likely to be strung together in writing. This is the same system that enables spelling correction—the system “knows” what you probably intended to type even if you made a mistake.

Based on the touch model, the keyboard is often able to quietly correct cases where you intended to type a **p** for example, but inadvertently struck the **o**, on a QWERTY layout. Or consider the example where you are typing the word “the.” If you type **t** then **h** and then touch between the **e** and **w** but slightly more on the **w**, the touch model adjudicates this, knows that **t-h-e** is the common character combination in English rather than **t-h-w**, and appropriately outputs the **e**. But if you touch the **w** fully, the keyboard respects that input and assumes you know best. This all happens while you are typing, so the right character goes into the input field and doesn’t require further correction. When this works best, you don’t realize it’s even happening, increasing your confidence in typing on glass.



This map from a report on touch model data illustrates biases that people show toward certain keys when typing on a touch keyboard.

Great for typing

Once we accounted for feedback and provided “guard rails” for inevitable mistakes, we still had to determine the specific keyboard layouts—what keys go where. Key positions have a big influence over typing speed and accuracy, and people have very strong—and often conflicting—opinions about keys. But the design problem broke down logically, based on our observations of interaction and some physical realities. For example, we confirmed our assumptions that:

1. Most people have developed very strong habits based on the conventions of physical keyboards. When you break these conventions, it

slows their typing down appreciably. This even applies to very young folks or dedicated T9 typists, for example, as most of us learn to touch-type in some form at a young age.

2. There are optimal targetable sizes of keys. The extensive research Microsoft has done into physical keyboards applied here too. For example, the letter keys on our touch keyboard are 19mm wide, the same as on most physical keyboards, because people showed faster typing speeds with targets of that size (rather than smaller or larger).
3. The more keys you include, the more likely people are to make mistakes. This is partly because more keys mean the keys need to be smaller and there's a greater likelihood of hitting a key you didn't intend. More keys also create visual clutter and distraction and slow your ability to scan and find a key.
4. You don't want to obscure more than half the display with a keyboard. A too-large keyboard creates a claustrophobic experience and you lose context. However, there is a counter rule that says obscuring about half the display works fine. This is because entering text is most often a "modal" activity, where your focus is very much on typing something and not on the periphery. Your area of focus outside the keyboard is relatively small, and directed toward the characters you're typing. Our eye-tracking studies, illustrated in this post, demonstrate this.
5. People use some keys more than others. We deduce this from analyzing passages of text written in real-world circumstances. There are clear patterns of frequency in the use of letters and symbols.
6. People will learn to do new things—and learn quickly—if they don't interfere with habits.

So in the end, the layout of a touch keyboard in any language becomes a balancing act of the different factors. You want to reduce the number of keys in the default layout, for example, but if you remove a key people rely on in typing every day, you will frustrate them. The layout needs to be big enough to support accuracy, but not so big it obscures the application.

There was one more overall rule or principle that we applied to the keyboard layouts specifically: *They must be great for typing*. That seems obvious but it's clarifying when you recognize that keyboards are used for a lot of things other than writing words—shortcuts to UI, for example, or sending commands, or entering codes. Our keyboard is optimized for typing, because that is its primary purpose and it must do it well above all other things. Let's take a look at a few of the decisions we made that fit within these parameters.

Numbers

We get a lot of questions about why we don't include a number row in the default keyboard layout. We use numbers frequently in our jobs, and we're used to finding number keys on the top of our physical keyboard. The Windows 7 on-screen keyboard has a number row, for example. This is consistent with the overall design of that keyboard—it is essentially a software emulation of a physical keyboard. It has not been optimized for a world of touch.



The Windows 7 on-screen keyboard emulates a physical keyboard and isn't optimized for touch or typing.

Some of our early designs and prototypes had a number row too. But when we brought these designs in front of people, the feedback was strong that the keyboard felt "cramped" compared to what they were used to. We observed frequent errors and accidental invocation of keys, especially around the perimeter of the layout. This resulted in a number of changes, and it confirmed the decision to not include a number row. Here's why: Including a number row meant adding a fourth row of character keys. When we optimize for keys with a targetable size, that means the keyboard must be that much higher. On a typical tablet device (say with a screen size of 10.6 inches) adding a number row would mean that more than half of the display would be covered by the keyboard. When we combined this with the observation that numbers are typed less frequently than most letters and common symbols, and you recognize that the extra keys are causing accidental key presses, we settled on including numbers on the separate number and symbol view.

That settled, we still had debates about whether to display numbers as a row across the top of the numbers and symbols view, or to display it as a numeric pad. We chose the numeric pad for a few reasons:

1. People often enter multiple numbers at once.
2. It's easier to scan an organized group than a long row.
3. People type number sequences much faster when the numbers are clustered.

We also decided to include the numbers in 1,2,3 order from the top, rather than 7,8,9, as it appears on many extended computer keyboards or cash registers. This is an interesting case where the physical keyboard convention didn't matter as much, because people have become familiar and very comfortable with the order of number pads on phones, ATMs, remote controls, and other modern devices. 1,2,3 order is simply easier for the eyes to scan and the brain to process than any other order.



The number and symbol view includes a numeric pad that reflects modern layouts we find on phones, ATMs, and remote controls.

Tab key

The tab key has a similar story. It's a key we use a lot—for formatting documents, but also for things like navigating input fields on a webpage. For that reason, we included it in one of our early touch-optimized layouts, after we had removed a lot of other keys typically found on physical keyboards. It looked like this.

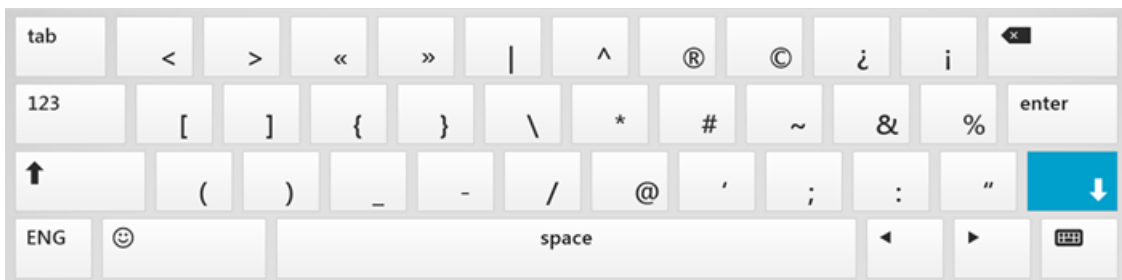


An early layout of the keyboard had extra keys that interfered with accuracy and speed.

You might observe that on the right and the left, there are borders of keys that aren't letters or symbols. This layout yielded the results described above—people experienced a cramped feeling. And worse than that, they frequently missed character keys and inadvertently touched one of the border keys. When we removed them, people raved about the openness and comfort of the layout, their errors went down, and their speed went up. With the Tab key on the numbers and symbols view, it was harder to reach—but the keyboard was better for typing, and so the Tab key's peregrinations were over.

Downshift: a mistake to learn from

The last example we'll share involves a feature we had in the product and have subsequently cut. This is a feature inspired by our desire to make punctuation easier to get to, without a complete view switch. In this design, the left shift key acted as the shift key does today—it enabled capital letters and access to alternate symbols from the default view. We used the right shift key differently—it provided a "peek" into frequently-used symbols or punctuation. The idea was that you would "downshift" briefly to select punctuation, for example, but not lose the context of the main view, and thus be faster. We theorized that this was a place where we could deviate from convention and provide value you could only get with software. Here's a picture of the "downshift" keyboard.



The downshift design was intended to provide fast way to access symbols, but interfered with expectations for shift behavior.

Suffice to say this prototype did not succeed in the lab. Participants continually struck the right shift key for the usual reasons you'd use a shift key. And when the keyboard showed the "peek" to symbols, they were confused and their typing came to a halt. So this was a case where we had to stick with the convention of a physical keyboard.

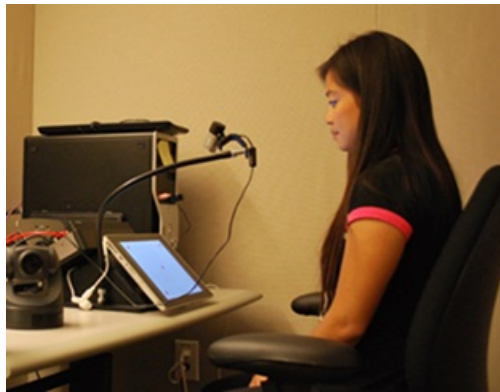
There is an interesting counter example in press-and-hold behavior. On a physical keyboard, when you press and hold a character, it repeats. On our touch keyboard when you press and hold, we show alternate characters or symbols. This is something a touch keyboard can do well and a physical keyboard can't. If you don't know the specific key combination to show ñ or é or š, for example, it's painful to type on a physical keyboard. It's easy to find on the touch keyboard. Practically no one has complained about this departure from convention. We built on it, in fact. You might discover that you can simply swipe from a key in the direction of the secondary key, and that character will be entered, without an explicit selection from the menu. So if you use accented characters a lot, you can get pretty fast with this. Try it out!



When you press and hold a key, it reveals related keys. If you swipe quickly toward the secondary key you want, you can select it quickly.

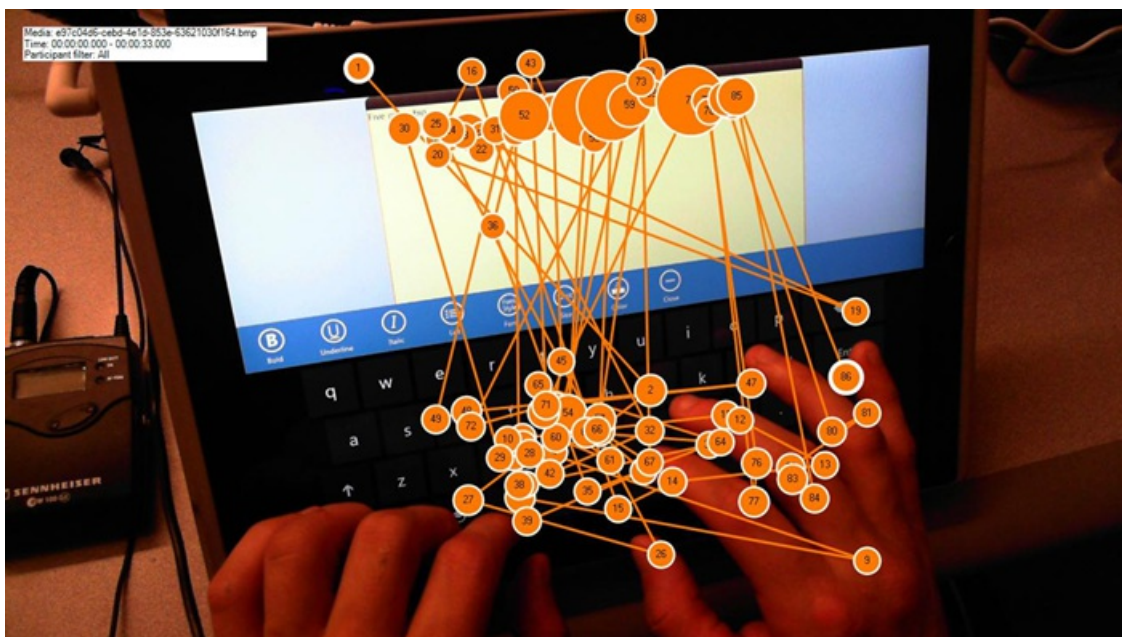
Testing and validating

We've been conducting a series of eye-tracking studies, where cameras record the direction of the participants' gaze as they are interacting with the system. These studies help us determine a few things: Where do people look when typing on a touch keyboard? Does visual gaze change over time? Are these patterns consistent across different views or layouts? And is visual gaze correlated to speed of typing?



An eye-tracking study participant begins the session.

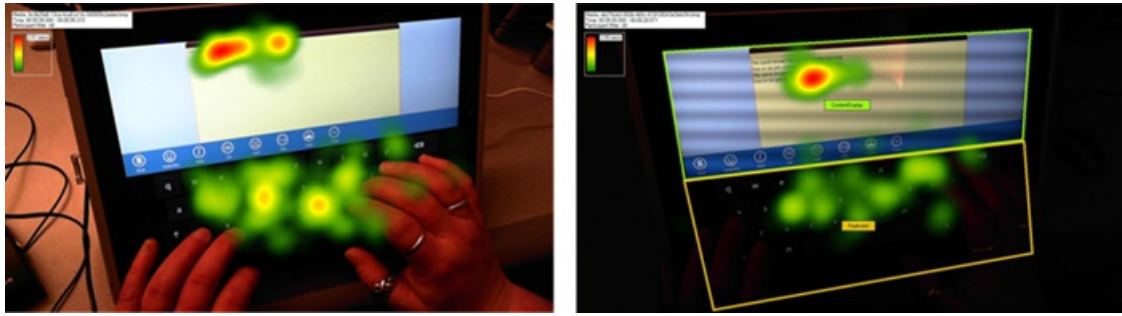
We've found very consistently that people primarily look at the text field where their characters appear, and they look at the keyboard. This is so consistent that we designed our text suggestion experience to optimize for this tendency. Text suggestions (words that are predicted as you type) appear right by the cursor in the text field, and you insert them by touching the "Insert" key on the touch keyboard. This is optimized for where we saw people putting their attention as they typed. It is notably different, for example, from text suggestion UI you see on many phones, where there is a band of possible words that run across the top of the keyboard. On a PC with a full-sized keyboard, people just don't look there, and they don't want to stop typing and change their posture to select these words.



Individual fixations, or recordings of a stabilized retina, show that people look either at the keyboard or at the text field. We do not typically look in between the two. Our text prediction UI appears near the caret for this reason.

We also found that our gaze does change over time, and as the gaze changes, we type faster. You can see this very clearly in the gaze plots of the

eye-tracking studies. A full range of people show this tendency—from slow typists unfamiliar with tablets to skilled typists who spend a lot of time with tablets. In all cases, at first, there is more attention on the keyboard, and the speed is slower. Over time—say, about 90 minutes over a few days—there is markedly less attention paid to the keyboard, more to the text field, and words per minute go up significantly.



We can see in lab studies that the focus of our gaze changes over time. The left hand image shows a typist after just a few minutes. The right image shows the gaze plots after about 90 minutes. You can see that focus moves to the text field. This typist doubled her speed during the session.

Continued refinement

Lastly, below is a picture of the current English QWERTY layout, which we have in the Windows 8 Release Preview. It is intentionally spare and open, and the keys that remain are there for explicit reasons. Each of these has its own story, but we can call out a few highlights:

- The **backspace** key is there because it's used very frequently on physical keyboards and touch keyboards. If we removed it, you would find your finger groping for it repeatedly.
- The **mode switch** key is essential to moving between views and languages and for hiding the keyboard. IME users will find that this is how you switch to Windows IMEs, which also feature touch-optimized keyboard layouts.
- The **CTRL** key and the right and left arrow keys are intended for text editing operations. You can move your input cursor and cut, copy, and paste without moving your hands from the keyboard. (Note that the CTRL key works just as it does on a physical keyboard—so any supported combination will work. We include labels for things like cut, copy, paste, and bold, because they are related to text editing. The touch keyboard is not intended for “commanding,” which is why you don't see things like the Windows key or function keys. That is a deliberate decision to stay focused on the goal of being really great for typing.
- The **space bar** is centered and wide. Physical keyboard research shows that about 80% of strikes on the space bar occur on the right (if you look at older keyboards, you will notice the wear on that side). This holds for touch keyboards too, where people will miss the spacebar if it's not ample-sized, and this creates errors that are hard to recover from.
- The **“emoji”** or emoticon key switches you to emoji view, where we support a full set of Unicode-based [emoji characters](#). The use of emoji continues to grow worldwide, and has become a part of how people write and express themselves.
- We also include an **option for a standard keyboard layout**, which can be useful on a PC without a keyboard when using desktop software that requires function keys or other extended keys. This is easily enabled from the settings Charm, in the General Settings section of PC Settings.

As you use the keyboard, we hope you also discover some extra features we've added to make things easier. For example, if you hold down the **&123** key, you can select symbols or numbers with your other hand, and when you release, you return to your original view. The team calls this “multi-touch view peek.”



The current touch-optimized layout reflects decisions about each of the keys based on a series of studies.

These optimizations apply across the input languages we have in Windows, as we support a touch-optimized typing experience worldwide. We expect to make a few more improvements to the typing experience, and we are really grateful and delighted by the feedback we've received so far. Thanks!

Kip Knox

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

Using the language you want

Steven Sinofsky | [2012-02-21T00:01:00+00:00](#)

*Since its introduction in Windows 2000, Multilingual User Interface technology, or MUI, has allowed customers to install additional display languages on their Windows PCs and to switch between them. But for the majority of users, the language you got when you booted up your Windows PC for the first time was likely the one you were stuck with. For Windows 8, we have reimagined the display language experience, **focusing on making additional display languages available to all Windows users**, making them super easy to find and install, and allowing users to switch between them. This blog entry unveils the changes we've been making in Windows to achieve this.*

February 21 marks UNESCO's International Mother Language Day. This year's theme is "Mother tongue instruction and inclusive education" and we think making sure Windows can be used in the language you want is one way we can contribute to this goal. For more information from UNESCO please see the full [site](#).

Ian Hamilton, a program manager on our Windows International team, authored this post.

--Steven

With Windows 8, we've changed how we think about languages from a "local-market feature" to a "feature for everyone everywhere," and have made it a priority for you to be able to work in any language you want, from any Windows 8 PC. If you can't read the text that Windows presents to you, you can't use Windows to its fullest potential. That's why we are so excited to bring powerful, easy-to-use language features to more users than ever in Windows 8.

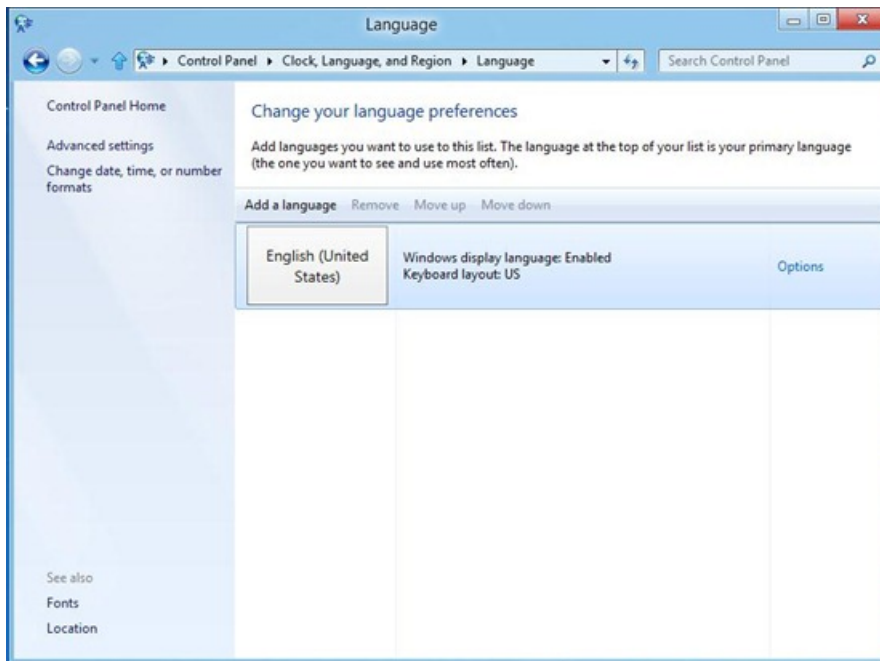
In some countries, people can purchase PCs with a variety of languages preinstalled. With Windows 8, users will be able install additional display languages beyond those preinstalled languages. This means that the language of the PC no longer needs to be a major consideration when deciding on which model to buy. If the language you want is not preinstalled on the PC you like, you can now install the one you want.

But for some families, allowing the installation of an additional display language might not be enough, as they also need the ability to switch between languages. To illustrate the point, let's look at the United States (where historically we have been less sensitive to these issues than in most other places around the world). We know from [2009 census data](#) that 80% of Americans speak English at home. The other 20% speak something other than English. Not surprisingly, 35,468,501 (12.41% of the total) speak Spanish at home. Some PCs sold in the US have had English and Spanish preinstalled on them. On those PCs, the user picks one language or the other, and the one not chosen is wiped off the hard drive after first run. Feedback showed that customers loved having a Spanish language PC, but what they really needed was Spanish *and* English, and the ability to switch between them. A subsequent study by an outside firm confirmed these results. In many cases, parents in the home spoke Spanish, and their children were speaking English. The ability to have a Spanish user account for the parents, and an English one for the kids—or at least the ability to switch a single account's display language back and forth between English and Spanish—was the way to delight these customers.

New, easier way to get languages

The new Language preferences section in Control Panel is the new one-stop place to find all Windows display languages in Windows 8. In the past, some languages were available through Windows Update, and others were distributed through the Microsoft Download Center.

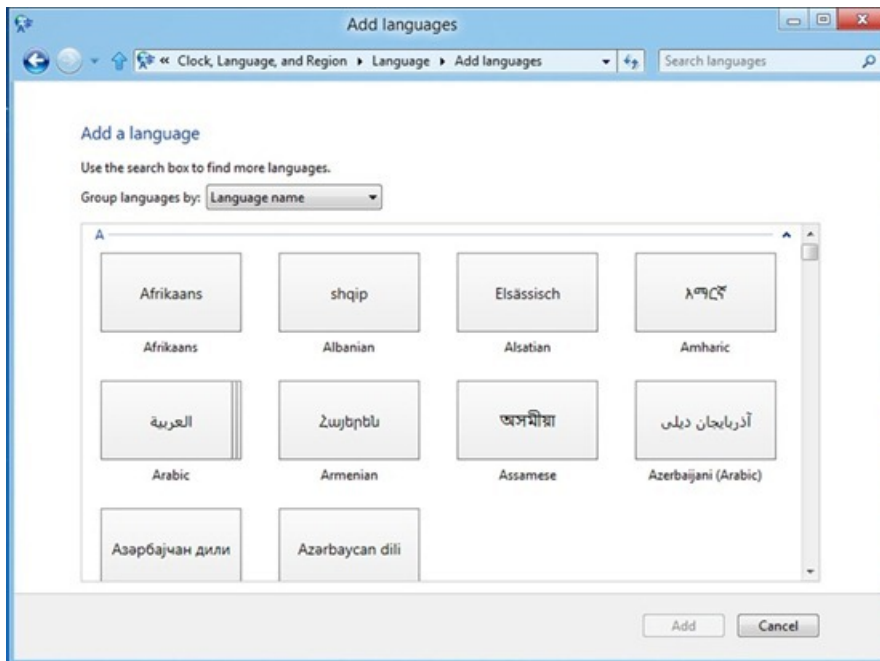
The reasons for separating the languages into two groups and their separated distribution channels made no sense to our customers. It wasn't their fault. This classification of languages only made sense to our internal teams. This confusion was a great motivator for re-imagining Language preferences in Control Panel. We will no longer ask customers to understand these nuances. Looking at the end-to-end experience, it made sense to build an entirely new experience around the acquisition of new languages. Here's what that looks like in Windows 8:



Language preferences in Control Panel

The main view of Language preferences shows you which languages are enabled on your system. You can see that on this system, English (United States) display language is installed and enabled. The keyboard layout is also US. Language preferences is the one place to go to add or change display languages, input language, and other functionality. We'll be talking more about that in future blog posts.

To add another language to your Windows, simply click the "Add a language" link above the first tile to bring up this list.



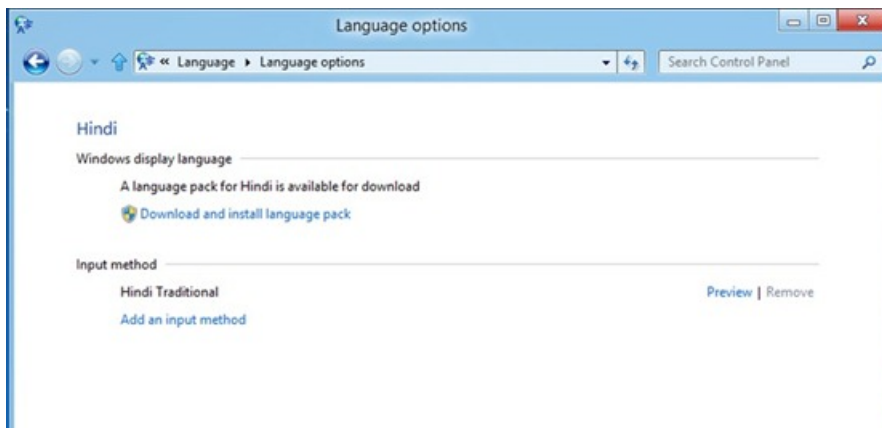
List of languages you can add to Windows

Select the language you want from the list. In these screenshots, I'm selecting Hindi. This list is long. Luckily, it's filterable. Just type the first few letters of the language you want into the search box, and the list is narrowed for you. This search filter works in both the native script as seen on the tile, and the localized name of the language.



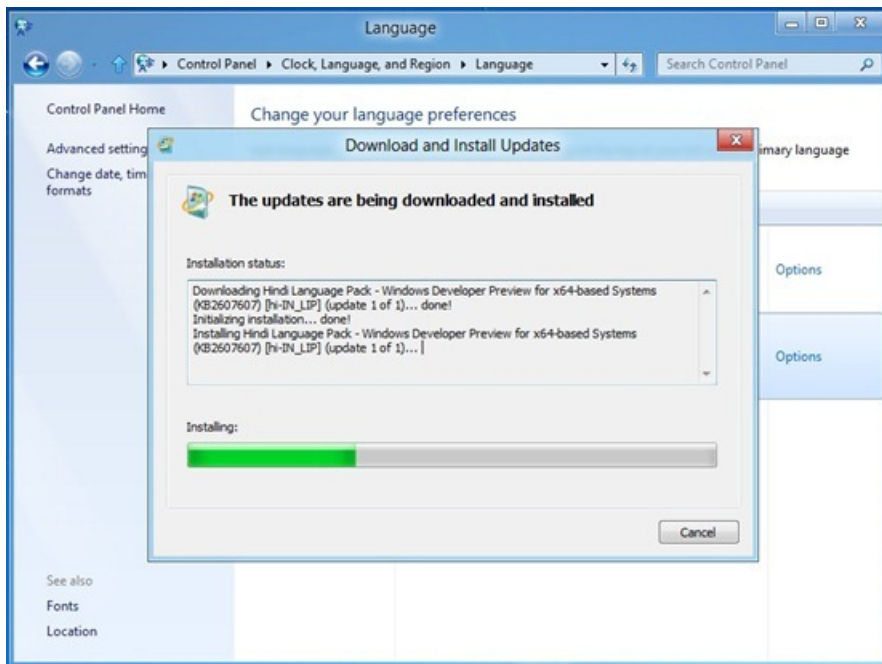
Hindi language has been added

Once selected, the language is added to your language list, but does not download and install the display language until you choose to do so. To add it as a display language, click Options.



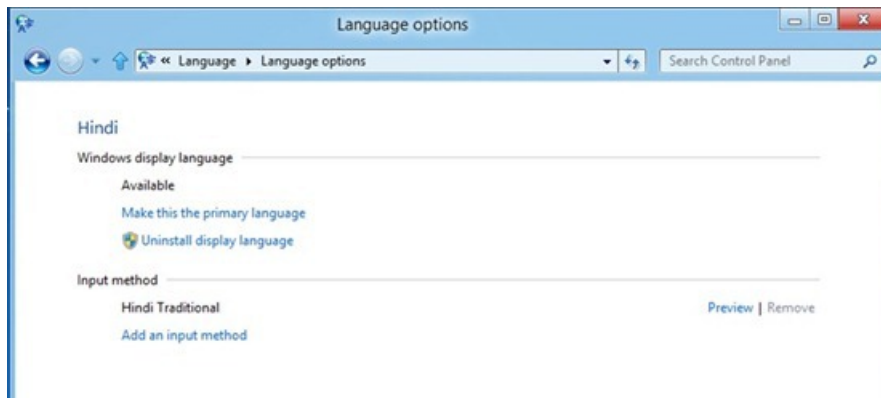
The Options page for this language shows the status of the language pack

If a language pack is available for your language, you will see the link to “Download and install language pack.”

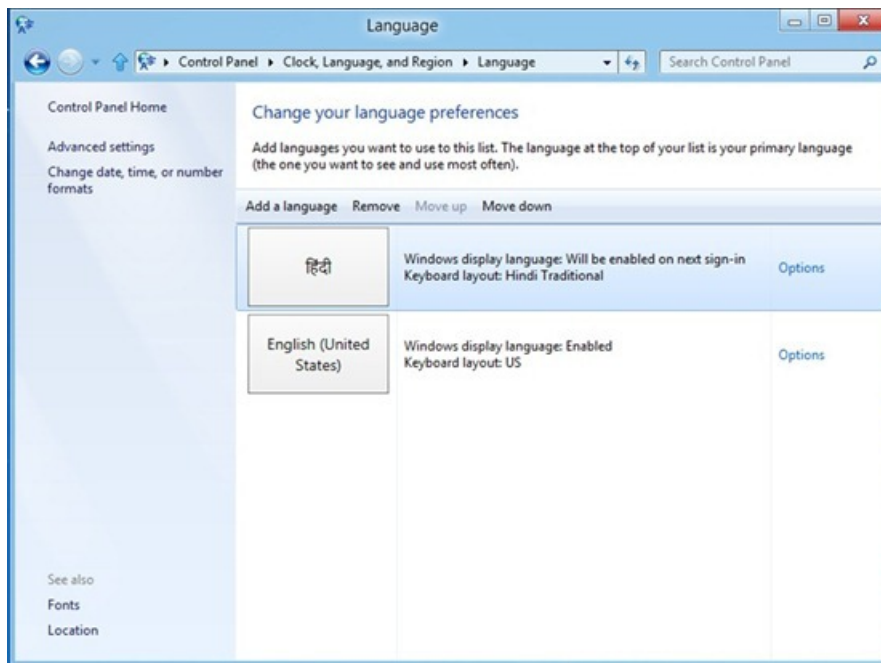


You can monitor progress of the download and installation

To switch to the newly installed display language, you’ll need to make it your primary language, by clicking “Make this the primary language,” as seen in the next screenshot.



Make your new language the primary language on the PC



Hindi is now the primary language on this PC

It's as simple as that. Pretty cool, huh? No more hunting around on websites looking for the languages you want. They're right here. If you are currently using Windows Vista or Windows 7 Ultimate, you probably see 34 or 35 languages as optional updates in your Windows Update UI. These won't show up there anymore in Windows 8. Instead, we've consolidated the languages in one place for you: Language preferences in Control Panel. Language preferences will be a clean, unified control for all Windows display languages moving forward.

More languages than ever before

Microsoft will continue to be a market leader in language support with an additional **14 new display languages** for Windows 8, bringing the total to 109 languages. (For reference, here are the [95 languages in which Windows 7 is currently available](#)). With these additional languages, Windows will provide a native language version of Windows for over 4.5 billion people.

It is important to note that a display language in Windows is a massive undertaking—Windows needs to support the fonts, localized text, and input methods to support a user experience that encompasses almost two million words. That's roughly the same number of words contained in two full sets of the Harry Potter series of books.

We are proud to announce the addition of English for the United Kingdom to the list of Windows display languages. We admit that this is something we should have done a long time ago. Windows users in the UK have gotten by with the US English version of Windows, and while we Americans knew this was not their *favourite*, that is clearly no *defence*. We believe that this version of Windows will also be widely used in India, Australia, South Africa, New Zealand, the Republic of Ireland and many other places.

We are releasing English for the United Kingdom as a standalone language. Standalone languages contain all the user interface components needed to be independent versions of Windows. Standalone languages can be used by OEMs to image a PC, or can be purchased as boxed software.

The release of English for the United Kingdom is also a trial run for us. Adding a second language under an already existing primary language code—ISO 3166-2 EN—poses some engineering challenges for us (which is why this took us so long to do). We have had to pay attention to the language fallback chain, for instance. If there are no localized resources available at any time, we fall back to secondary choices and then to English. That used to be English US. But, now there's English UK as well. Which do we fall back to? So far, planning for these scenarios is looking good.

We are also continuing to broaden our language support with the addition of **13 new Language Interface Packs (LIPs)**. Language Interface Packs install over the top of a standalone Windows display language. These lightweight packs contain localized user interface elements for the most commonly-used Windows features. The new languages offered include Punjabi (Pakistan), Sindhi (Pakistan), Central Kurdish (Iraq), Uyghur (People's Republic of China), Belarusian (Belarus), Kinyarwanda (Rwanda), Tigrinya (Ethiopia), Tajik (Tajikistan), Wolof (Senegal), K'iche' (Guatemala), Scottish Gaelic (United Kingdom), Cherokee (United States), Valencian (Spain).

This set of languages includes language coverage for emerging markets that are experiencing great growth in PC usage: **Punjabi (Pakistan), Sindhi (Pakistan), Central Kurdish (Iraq), Uyghur (People's Republic of China), Belarusian (Belarus), Kinyarwanda (Rwanda), Tigrinya (Ethiopia), Tajik (Tajikistan), Wolof (Senegal), K'iche' (Guatemala)**; and a few languages that are preferred by groups of customers in developed markets: **Cherokee (United States), Scottish Gaelic (United Kingdom), Valencian (Spain)**.

While these packages remain different in how they're installed, users will not need to understand those differences. Language preferences in Control Panel is the one place where they'll go to get new Windows display languages, and it handles download and installation seamlessly.

Display languages are just the beginning

Most of this post has focused on Windows display languages—the language of the Windows user interface on your computer. We have focused our language efforts in Windows 8 on:

- Enabling more users than ever to install additional languages on their Windows PCs and switch between them.
- Building a Language preferences area in Control Panel that is an easy-to-use central location for all display languages.
- Making significant additions to our language list by adding one standalone language and 13 Language Interface Packs (LIPs).

We're super excited about these improvements in Windows, and we hope you'll like them as well.

But display languages are only one part of the overall language story for Windows 8. In a future blog entry we will tell you about improvements in text entry, locale support, and other critical pieces of the Windows 8 story. Let us know if there are other language-related topics you'd like to hear more about.

Thank you,

Ian Hamilton

Using your feedback to make Narrator work better with touch

Steven Sinofsky | [2012-07-18T06:00:00+00:00](#)

*Shortly before we released the Windows 8 Consumer Preview in February, we [blogged about our work to make Windows 8 more accessible](#) to people with disabilities. This included our work on Narrator to enable customers who are blind to use Windows 8 on touch screens. This work has continued to evolve in the Release Preview, and will also improve as we move toward the final release of Windows 8. This post details some of the work we have done to improve Narrator when using a touch-enabled PC. **This post was authored by Doug Kirschner on our Accessibility team.** –Steven*

First off, we would like to thank all the people who have given us feedback; there has been a lot of positive reaction—people are excited that Windows 8 touch screens will include basic screen reading support by default. We've gotten a tremendous amount of constructive feedback on things we could do to make Narrator work better on touch screens and easier to use on the web. We've listened. Your suggestions, combined with suggestions from usability testing on visually impaired users here at Microsoft, have resulted in some important changes that we think you'll really like.

Listening to the accessibility community

When the Developer Preview build was released, we took the opportunity to reach out and gather feedback on Narrator from as many people who require visual assistance tools as we could. To start with, we worked with the community of folks inside Microsoft (we are fortunate to have a significant and organized community that is engaged in the accessibility of all Microsoft products) to install Windows 8 and send us their impressions, and we held internal accessibility events where people could come and try it out in person. We also held usability studies where we invited people to Microsoft's campus to experience Narrator on a touch screen and walk through common tasks to see where we could improve. Millions of you downloaded the Developer and Consumer Previews, and many of you tried out Narrator and sent us some great feedback. We followed up with a number of people who contacted us via [@BuildWindows8](#). Lastly, we attended the [CSUN conference for Technology and Persons with Disabilities](#), where we were lucky to have the chance to sit down with people one-on-one as they tried out the Windows 8 Consumer Preview for the first time on touch screens.

There were a couple of key scenarios we wanted to validate. In particular, we wanted to make sure touch users could get up and running using Narrator on a new PC, right out of the box. That includes finding and installing accessible apps from the Store, and accomplishing basic everyday tasks like sending email, reading webpages, and listening to music. The excitement around the work we'd done so far was overwhelming and gratifying, but it was clear that we still had more work to do to make touch Narrator even better.

Thanks to all of your constructive feedback, we identified key areas that we've improved for the Release Preview:

- **Responsiveness:** We heard that Narrator on touch screens didn't feel responsive enough.
- **Gestures:** Some people had difficulty with Narrator gestures, particularly some of the more complicated multi-finger gestures.
- **App exploration:** Finding particular elements on the screen (e.g. finding tiles on the Start screen) could be hard for people not already familiar with the particular app or UI.
- **Web navigation:** The commands available in the Consumer Preview were not extensive enough for some webpages.

We worked heavily on each of these areas for the Release Preview, and we're still working in some areas for the final release of Windows 8. We wanted to share with you some of the improvements you can already experience in the Release Preview today.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

Making Narrator feel more responsive to touch

Some people we heard from felt that Narrator touch was not very responsive. We heard various versions of this feedback—that Narrator was slow, that Narrator sometimes didn't respond, or that people just felt disconnected or disoriented—but the root cause of the issue was the same. When you touch the screen, you expect a timely response. We found two common scenarios where this problem occurred:

- **Single-finger exploration:** When people had to find an item on the screen by dragging a finger around, we observed that they would often skip right over the item they were searching for, as they moved their fingers too quickly, generally before Narrator had a chance to start reading the item.
- **Gesture response:** Some people were confused as to whether their gesture had succeeded, and would attempt to repeat the gesture several times, even though the first attempt was already successful. The problem was that there was a delay between the time Narrator recognized the gesture, and when it provided the speech response. Sometimes it was also unclear from the response whether Narrator had done what the user wanted, or was just reading something similar but unrelated.

In each case, the blue, visual highlight rectangle that moves to whatever Narrator is currently reading was quick to jump to the appropriate item, indicating that Narrator had registered the user's movement and was responding appropriately. However, the problem was in the actual speech process. The text-to-speech (TTS) synthesis is fast, but even at high speeds, it takes a while for the system to read the response back; moreover it

took additional cognitive time to process the language and to understand what they were hearing. To complicate matters, the speech response time varied widely, depending on context, which made it hard for the user to discern whether the intended gesture was the one that Narrator had recognized. Each of these minor delays added up; people would skip over items altogether or repeat successful gestures, thinking that their first attempt was not successful.

Audio cues

For users with full vision, even if an action takes a few more milliseconds to complete, visual feedback such as highlighting a button or animating a flyout help indicate immediately that the system is responding. These cues are not only aesthetically pleasing, but also functionally important to understand how your touches are influencing the system in real-time.

As we dug into some of the feedback around responsiveness, we realized that Narrator could make more effective use of audio cues. In the Release Preview, we have started to add audible cues; each gesture now has an associated sound that plays when the gesture is performed. These cues were designed to be quick, short and easily distinguishable, allowing you to instantly recognize whether your gesture is successful and if your action has been taken. Here are some examples:

- Moving to the next item plays a “tick.”
- Activating plays a “click.”
- Scrolling plays a sliding sound.
- Selecting plays a “thud.”
- Narrator errors play a “bloop” sound that is easily distinguishable from the system error "ding."
- Explore the screen with a single finger, and Narrator makes a tick with each new item that you touch, so you know if you passed over an item too quickly to hear what it was.

We had a lot of fun designing and implementing these sounds!

Making interactions easier

The next step was to tune Narrator's touch interaction model. Some people told us they found it difficult to use multi-finger gestures. In particular, we saw people struggle with the two-finger swipe for next and previous item, and even more so with the four-finger swipe to scroll. We also observed people accidentally triggering the commands lists (available item commands, search window, etc.), which consequently caused them to lose their context in an app.

In response, we've made it easier to interact with touch Narrator. The system is now more forgiving, with a simpler gesture model that is easier to remember. Single-finger taps and flicks now carry out a majority of the common tasks in Narrator. The revised interaction model is easier to perform, and it groups gestures more logically, so that command lists and windows don't pop up when you're trying to perform an unrelated gesture.

The table below outlines the new interaction model:

Touch gesture	Command
Tap or drag	Read item under finger
Double-tap OR	
Hold with one finger and tap anywhere with a second	Do primary action
Triple-tap OR	
Hold with one finger and double-tap with a second	Do secondary action
Flick left or right	Move to previous/next item
Flick up or down	Change move increment

Hold with one finger and 2-finger-tap with additional fingers	Start dragging or extra key options
2-finger tap	Stop speaking
2-finger swipe	Scroll
3-finger tap	Show/hide Narrator settings window
3-finger swipe up	Read current window
3-finger swipe down	Read from current location in text
3-finger swipe left or right	TAB forward and backward
4-finger tap	Show commands for current item
4-finger double tap	Toggle search mode
4-finger triple tap	Show Narrator commands list
4-finger swipe up or down	Enable/disable semantic zoom (semantic zoom provides a high-level view of large blocks of content)

Improving Narrator’s exploration model

As we collected feedback from people who were using the Developer Preview, we reviewed the exploration model in Narrator. One of the things we heard clearly was that people wanted an easy way to find all of the controls on the screen like buttons, labels, text fields, list items, etc. without having to manually touch around the whole screen. One user who was blind gave the analogy that when he enters a hotel room, his first task is always to walk around the room and locate the door, dresser, beds, and bathroom in order to understand the layout of the room before doing anything else. Similarly, when exploring a new app, users want to know what's on the screen before deciding what to do next.

One of the ways we made all elements on the screen accessible in Developer Preview was to use horizontal swipe gestures to move between items in a container, and vertical swipe gestures to move into and out of containers. This was a powerful model—you could find all accessible items on the screen—and it was a true representation of how graphical UI is constructed. However, it wasn't intuitive. Having to navigate into and out of containers made it difficult to discover all of the interesting elements on the screen.

Changing our default cursor mode

In response to the feedback, we made some changes to the way navigation works by default in Release Preview. The navigation gestures, which are now all single-finger flicks left and right, move you through all of the items on the screen. You no longer need to know how the UI is constructed in order to navigate it; all you need to do is flick to get to the next and previous items, and Narrator presents you with a linear ordering of the important items on the screen.

This allows you to learn about all of the interesting items in an app in an easy step-by-step manner, and interact with any item as you go. If you just want to hear all of the items in an app without flicking each time, you can swipe up with three fingers and Narrator will read through all of them in order, without stopping.

(Note: This is the new default mode of navigation, which allows you to explore apps by flicking left and right to find all of the interesting items. If you prefer the old way of moving through the multiple layers of UI manually, you can change the Narrator cursor movement mode to “Advanced” in the Narrator settings).

Improving web navigation

In Windows 8, Narrator has made reading the web much easier. It has various features that are optimized for web reading, such as the “start reading” command, which reads out continuous sections of webpages without stopping, and search mode, which provides a list of various types of controls on a page. After we released the Developer and Consumer Preview builds, we heard from users that although these features were helpful, they did not enable them to accomplish some common tasks on the web, such as quickly scanning news headlines, doing a quick search, or checking stock quotes.

So we revisited this feature, and as we dug further and gained a better understanding of these scenarios, we found ways to improve them in the Release Preview. For news reading in particular, we heard people saying they wanted to jump to various points in the page (e.g. headings, links), and then subsequently to be able to read line-by-line and even letter-by-letter. Many users wanted Narrator to provide these commands for them to navigate the web with more precision.

In response, we added the concept of views to Narrator’s navigation commands. The new views are available in default navigation mode whenever you are on a webpage or other accessible text area, such as in the Mail app. The default Item view moves through the items on the page, and works the same way as item navigation throughout the system. But for accessible text areas such as webpages or Mail, Narrator now supports seven additional views:

- Headings
- Links
- Tables
- Paragraphs
- Lines
- Words
- Characters

You can easily change the view by flicking up or down, and then flick left or right to move through the items in that view. These commands are also available with a keyboard by using Caps Lock + Arrow keys.

With the new views, web reading is more powerful in the Release Preview. The views work with other Narrator reading commands as well. For example, if you find an interesting news headline and want to hear more, you can swipe down with three fingers and Narrator will start reading all of the page content until you tell it to stop.

Finishing the job

These examples represent some of the major work we’ve done in response to feedback from people who tried Narrator touch in the Developer Preview and Consumer Preview. We’ve made many more improvements based on your feedback—including reading out touch hints that teach you how to activate items, improving the Narrator settings UI to be easier to use with touch, and adding a new setting that makes it easier to type on the touch keyboard. While we believe Narrator is feature complete at this point, we’re still fixing bugs and fine-tuning it before Windows 8 is complete.

It’s been fantastic and humbling to hear from so many of you who have had the chance to try out Narrator. We’ve thoroughly enjoyed working one-on-one with users through our usability studies, at the CSUN conference, and within the Microsoft community. Thanks to all of the great constructive feedback we’ve received, we’ve made these important changes to Narrator for the Release Preview to make it a much better feature.

While we work towards shipping this product soon, we’d love for you to download and install the [Release Preview](#) for yourself, and try out Narrator.

Note: The touch features described in this blog require touch screens supporting at least four contact points. Windows 8 certified touch hardware will universally meet this requirement, but some current Windows 7 hardware may not (see [this post](#) for more info). If you do not have a touch screen supporting four contact points, you can still run Narrator using the keyboard.

Thanks!

-Doug Kirschner

Hardware accelerating everything: Windows 8 graphics

Steven Sinofsky | [2012-07-23T09:00:00+00:00](#)

*With Windows 8 we set out to enable all applications to have the beautiful and high-performance graphics enabled by modern graphics hardware. This work builds on the well-established foundations of DirectX graphics, which have been providing an increasing breadth of APIs and capabilities. In Windows 7, we expanded the capabilities of DirectX to provide a common hardware-accelerated graphics platform for a broader range of applications. Whereas previously, DirectX mainly provided 3-D graphics, we added functionality for what we call “mainstream” graphics. Mainstream uses center on the typical desktop applications most people find themselves using every day, including web browsers, email, calendars, and productivity applications. Windows 7 added two new components to DirectX: Direct2D for two-dimensional graphics (shapes, bitmaps, etc.) and DirectWrite for handling text. Both of these additions not only focused on performance but also on delivering high-quality 2-D rendering. With these additions, DirectX became a hardware-accelerated graphics platform for all types of applications. Indeed, we showed what a typical application could achieve by using DirectX when [Internet Explorer 9](#) brought hardware-accelerated graphics to the web. WinRT bring these capabilities to the full range of new Windows 8 applications. In this post, **authored by Rob Copeland the group program manager on our Graphics team**, we look at the details behind the scenes in enabling this new class of graphical application. --Steven*

In computer graphics, high performance is a guiding principle. In the early days of personal computing, discrete, add-on graphics cards were mostly focused on specialized applications such as CAD/CAM and gaming. Even early on, there was a view that all of this graphics horsepower could be used for more: notably a better user interface and experience. One of the first graphics cards for a PC was called a “[Windows Accelerator](#)” from S3 Graphics, which focused on the user experience by moving windows around the screen faster. As graphics hardware evolved, so, too, did the methods that developers use to interact with that hardware.

DirectX is the part of Windows that provides a common application programming interface, or API, that allows developers to use the graphics hardware in the PC to draw text, shapes, and three-dimensional scenes, and display them on the screen. DirectX has also evolved over time in both capabilities and performance characteristics. In the early years, DirectX was focused mainly on games. As applications evolved to provide richer and more graphically-intense user experiences, many of them started to use DirectX as a way to get better performance and richer visuals.

Enter Windows 8

When we started to plan the work we’d undertake for graphics in Windows 8, we knew that we would be creating a new, visually rich way for users to interact with apps and with Windows itself. We also knew that we’d be building a new platform for creating Metro style apps, and that we’d be targeting a more diverse set of hardware than ever before. While we had a great graphics platform to start with, there was more work to do in order to support those efforts. We came up with four main goals:

1. Ensure that all Metro style experiences are rendered smoothly and quickly.
2. Provide a hardware-accelerated platform for all Metro style apps.
3. Add new capabilities to DirectX to enable stunning visual experiences.
4. Support the widest diversity of graphics hardware ever.

While each of these focus on different aspects of building Windows 8, they all depend on great performance and capabilities from the graphics platform.

Planning for performance

Graphics performance on Windows depends on both the operating system and the hardware system, comprised of the CPU, the GPU (graphics processing unit), and the associated display driver. To ensure that we could deliver a great experience for new Metro style apps, we needed to make sure that both the software platform and the hardware system would deliver great performance.

In the past we’ve used many different benchmarks and apps to measure the performance of DirectX. These have been largely focused on 3D games. While games are still very important, we knew that many of these existing ways to measure graphics performance did not tell us everything we needed to know for graphics-intensive, 2D, mainstream apps.

So we created new scenario-focused tests and metrics to track our progress. The metrics we use are as follows:

1. Frame rate

We express frame rate in frames per second (FPS). This metric is widely reported for gaming benchmarks, and is equally important for video content and other apps. When something is animating on the screen, a rate of 60 FPS makes the animation appear smooth. We target that rate

because most computer screens refresh at 60 hertz. With that frame rate, Windows can provide very smooth animations with “stick to your finger” touch interactions.

2. Glitch count

While frame rate is an important metric, it doesn't tell the whole story. For example, running a benchmark for 10 minutes and getting 60 FPS on average sounds perfect. But, it doesn't tell us how low the frame rate might have dropped during the test. For example, if the frame rate dips down to 10 FPS momentarily during demanding parts, the animations will stutter. The glitch count metric looks for the total number of times that rendering took more than 1/60 of a second, thus resulting in a reduced frame rate. It also looks at the number of concurrent frames missed. The goal here is to have no missed frames during animations.

3. Time to first frame

Most people expect their apps to launch quickly, so initializing DirectX needs to be fast. “Time to first frame” tells us how much time it takes from the moment you tap or click to launch an app until you see the first frame of the app on the screen. To measure this, we created simple apps to help analyze and optimize the graphics system for the time it takes to initialize a graphics device, allocate the required memory, and so on. This helps us ensure that the work to set up DirectX takes very little time.

4. Memory utilization

The more memory our graphics components use, the less memory is available for apps. By ensuring that most of the system's memory is available for apps, you get the best app performance, and more apps can run at the same time. Apps use a mix of system memory and GPU memory. GPU memory is mostly used for rendering operations such as drawing images, geometric shapes, and text. Additionally there are graphics operations that use the CPU and therefore use system memory.

In order to characterize memory utilization, we measure the memory used by the system for the following scenarios:

- **The app is idle.** That is, it is not doing any work and is not rendering or displaying new information to the screen.
- **The app is displaying information to the screen.** This represents the base memory cost of a simple drawing.
- **Texture creation.** This represents the memory used for creating a large number of image objects on the GPU.
- **Vertex buffer creation.** This represents the memory overhead of creating geometric shapes.
- **GPU data upload.** This measures memory overhead involved in uploading data to the GPU.

Measuring memory usage across many types of apps and these various scenarios has helped us further optimize DirectX and the display drivers.

5. CPU utilization

Most graphics operations utilize the CPU in addition to the GPU. For example, when an app is figuring out what it's going to draw, it typically does these calculations on the CPU. CPU utilization is important to understand because the higher the percentage of the CPU used by a task, the fewer cycles the CPU can devote to other tasks. For good graphics performance and overall system responsiveness, it is important to effectively balance work between the CPU and the GPU.

These benchmarks and metrics help us ensure that the experiences and apps are smooth and have great performance. They play a big role in our understanding of mainstream apps. Of course, we still utilize industry benchmarks, games, and other ways to measure our overall performance.

Hardware accelerating mainstream graphics

There are many ways to look at mainstream graphics. To ensure that our work would give users the right performance and the right experiences we studied many examples of both Metro style and desktop apps to understand how they used the graphics hardware. In particular, [Internet Explorer 9](#), [Windows Live Mail](#), and [Windows Live Messenger](#) make excellent use of DirectX. Because these apps have done great work utilizing DirectX, they're good examples of what other apps might do. This led to a number of investments to ensure mainstream apps were fast and looked great.

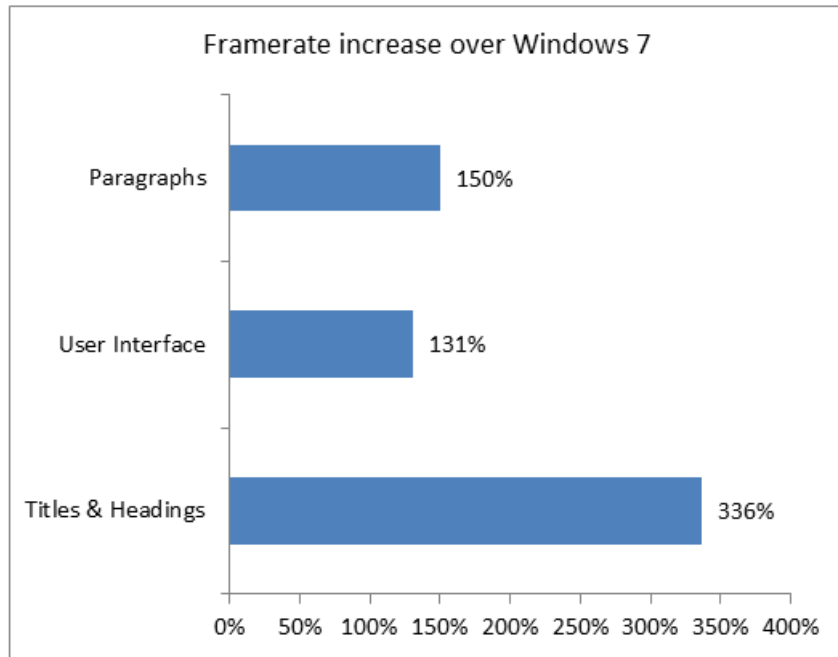
Improving text performance

Text is by far the most frequently used graphical element in Windows, so improving text rendering performance goes a long way towards creating a better experience. Web pages, email programs, instant messaging, and other reading apps all benefit from high-quality and high-performance text display.

The Metro style design language is typographically rich and a number of Metro style experiences are focused on providing an excellent reading experience. [DirectWrite](#) enables great typographic quality, super-fast processing of font data for rendering, and provides industry-leading global text support. We've continued to improve text performance in Windows 8 by optimizing our default text rendering in Metro style apps to deliver better performance and efficiency, while maintaining typographic quality and global text support.

The bar chart below illustrates the performance improvements that result from this work. It includes measurements for the following text scenarios:

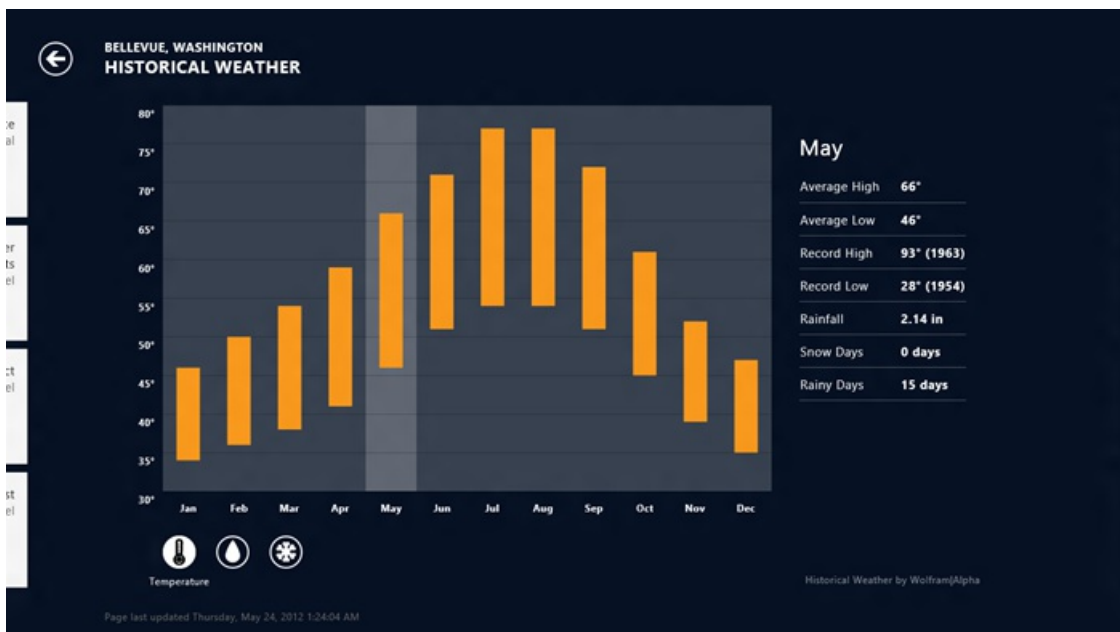
- Rendering a screen full of reading-size text formatted as **paragraphs** as you would find in a web page or Word document
- Rendering a screen full of small chunks of text at reading sizes as you would find in **user interface** controls such as button labels or menus
- Rendering a screen full of small chunks of heading-sized text as you would see in **titles & headings** in Metro style apps and as headlines on blog posts and news articles on the web.



The most noticeable performance improvement can be seen when scrolling through a long document on a touch screen. The reduction in time required to render the characters frees up CPU cycles to handle other tasks like processing high-frequency touch input, or displaying more complex document layouts.

Improving geometry rendering performance

Along with text, we also made dramatic performance improvements for 2D geometry rendering. Geometry rendering is the core graphics technology that is used to create things like tables, charts, graphs, diagrams, and user interface elements, as shown in the example below. For Windows 8, our improvements in this area have primarily focused on delivering high-performance implementations of HTML5 Canvas and SVG technologies for use in Metro style apps, and webpages viewed with Internet Explorer 10.



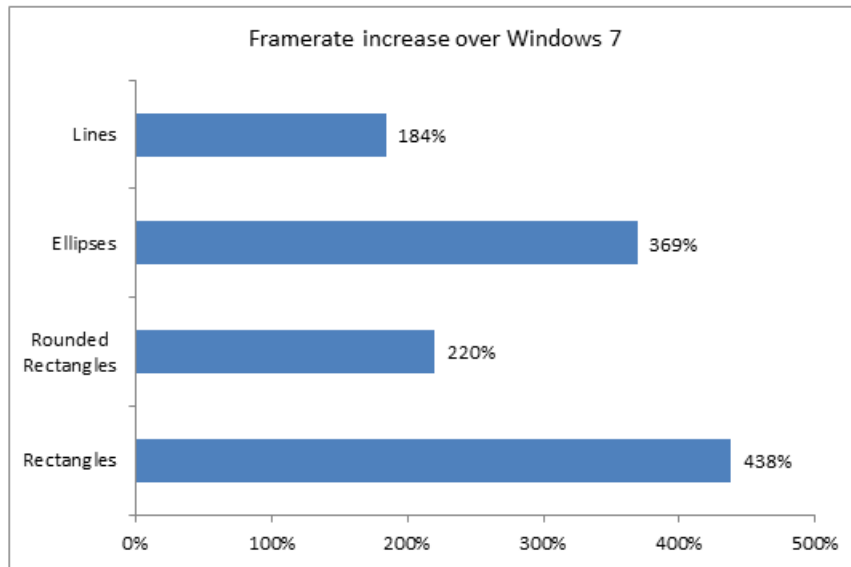
The Weather app in Windows 8 uses geometry to display a graph of historical temperature and precipitation data

When Direct2D draws geometry, it takes instructions from the app about what to draw in the form of 2D figures (e.g. rectangles, ellipses, and paths), the size and location of the figures, and specifics about the style of rendering, including brush color and stroke style. Then it converts those instructions into a set of triangles and commands that it sends to Direct3D to generate the desired output. We call this conversion process

tessellation.

To improve geometry rendering performance in Windows 8, we focused on reducing the CPU cost associated with tessellation in two ways.

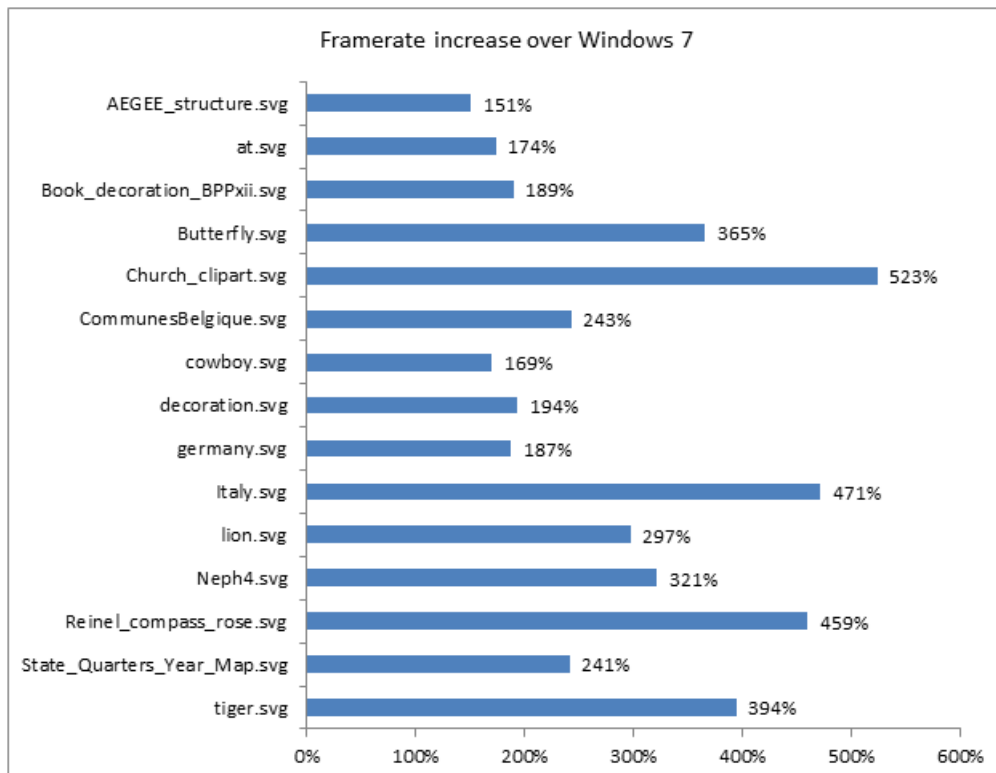
First, we optimized our implementation of tessellation when rendering simple geometries like rectangles, lines, rounded rectangles, and ellipses. Below is a chart showing the impact of these improvements.



Second, to improve performance when rendering irregular geometry (e.g. geographical borders on a map), we use a new graphics hardware feature called *Target Independent Rasterization*, or TIR.

TIR enables Direct2D to spend fewer CPU cycles on tessellation, so it can give drawing instructions to the GPU more quickly and efficiently, without sacrificing visual quality. TIR is available in new GPU hardware designed for Windows 8 that supports DirectX 11.1.

Below is a chart showing the performance improvement for rendering anti-aliased geometry from a variety of SVG files on a DirectX 11.1 GPU supporting TIR:



We worked closely with our graphics hardware partners to design TIR. Dramatic improvements were made possible because of that partnership. DirectX 11.1 hardware is already on the market today and we're working with our partners to make sure more TIR-capable products will be broadly available.

Rendering images

Images are widely used in a variety of scenarios including displaying user interfaces, webpages, and other app content. Websites commonly use JPEGs for pictures and PNG and GIF files to efficiently store user interface elements such as button graphics.

Working with digital photographs is also a very common activity on Windows. The number of digital photographs that Windows customers view and manipulate on their PCs continues to grow at an incredible rate.

We've made several performance improvements for working with images and photographs using the JPEG, GIF, and PNG formats.

For JPEG, improvements include:

- Faster image decoding by expanding [SIMD usage](#) on all CPU architectures
- Faster [Huffman](#) decoding and encoding

For PNG, improvements include:

- Faster image decoding by expanding SIMD usage on all CPU architectures
- Faster image encoding and decoding by optimizing our [zlib](#) implementation

In addition, we've improved pixel format conversion as well as image scaling. This results in faster decoding and rendering of images for all apps.

The video below uses a test app to measure the decoding and rendering time for a set of images. Windows 8 takes 40% less time than Windows 7 to render 64 images (4.38 seconds vs. 7.28 seconds)

Rendering and displaying

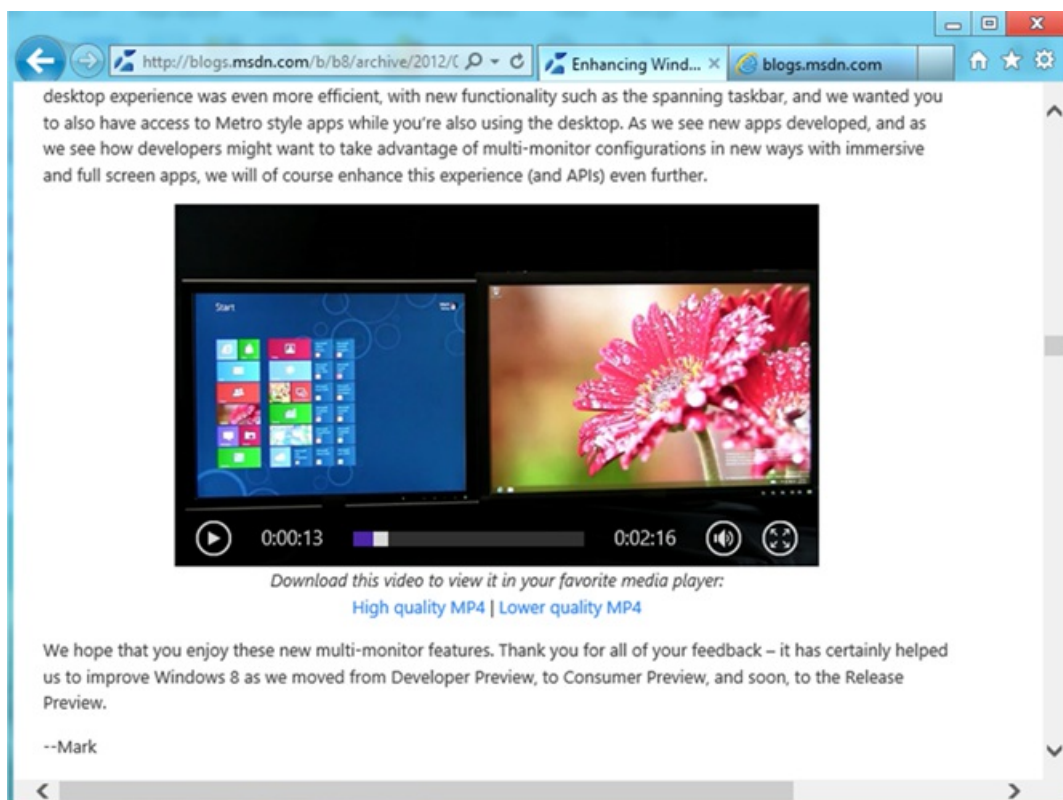
As we evolve DirectX to support more mainstream scenarios, another area we've invested in is optimizing how apps render and display their content. There are some big differences in how a 3D game draws its content and how a mainstream app such as Internet Explorer draws its content. For example, consider the video of the game below. In games like this, the entire scene changes rapidly. As the "camera" moves around the vehicle, the clouds move across the sky, and smoke billows up from the engine, the app must redraw the entire scene in each frame in order to achieve a life-like and engaging experience.

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

Now consider the webpage below. It has both a text article and a video. While the video plays, the browser must update the portion of the window containing the video but not the text. Additionally, if the user scrolls the page up, then we only need to render the new text at the bottom of the page. The rest of the text has already been rendered and simply needs to be moved.



To improve apps that don't need to redraw the entire screen for each frame, we optimized how DirectX deals with redrawing just portions of the screen and how it scrolls. This work not only improves app efficiency and performance, but since it reduces redundant drawing and reduces the

number of times graphics data needs to be copied in memory, it also reduces power consumption, thus increasing battery life.

Making the entire platform great

All of these changes help Windows render experiences very quickly and smoothly. While we've talked mostly about features in DirectX, the great thing is that all of this work contributes to making our entire platform hardware-accelerated by default. Since we built the Metro style platform on top of DirectX, all apps take full advantage of the graphics hardware on the system, regardless of the programming language and framework the developer chooses.

Creating stunning visual experiences with Direct2D and Direct3D

Direct2D Effects

Stylistic effects applied to images are becoming more common in modern user experiences. They can help highlight an area of an app, draw your attention to a specific part of the screen, or just make things look better. As we planned the graphics capabilities for Windows 8, we wanted to make it really easy for developers to apply these types of effects in their apps. We looked at two main areas where image processing would be useful:

- **User interface images**

The Metro style experience uses dynamic visuals. We wanted to enable Metro style apps to do image processing in real-time. This can range from 3D transition effects to perspective transforms, blurs, and highlights on user interface elements.

- **Photos**

Apps that deal with photographs often want a rich set of image processing features. Effects such as adjusting exposure, brightness, and contrast, applying vibrancy and clarity, working with advanced curves, and applying lens corrections all allow these apps to enhance your digital memories.

To enable these types of experiences, we added "Direct2D Effects," a new set of APIs that enable high-quality, hardware-accelerated effects to be applied to any image. Direct2D Effects have the following benefits:

- They provide optimal-quality renderings of image effects to suit the needs of wide variety of apps.
- The effects are hardware-accelerated and work on a wide variety of graphics hardware.
- A simple API enables great effects with minimal programming.
- They provide many built-in effects.
- They support large image sizes and up to 32 bits per channel.
- Custom effects can be combined with built-in effects or other custom effects.

Direct2D Effects power some of the new user experiences in Windows 8. For example, when tapping on a tile on the Start screen, the tile uses the 3D perspective transform effect to "tilt" in the right direction. They also power the rest of our platform. For example, SVG filter effects and CSS 3D transforms are implemented using Direct2D Effects.

Direct3D 11.1 as a common foundation

While adding new features like Direct2D Effects is a great way to help developers deliver new experiences, we also looked at ways to make it easier to use existing DirectX features.

Over years of development, we've added various different features to DirectX. Hardware acceleration of video decoding came alongside programmable [shaders](#) in DirectX 9. In Windows 7, we added Direct2D and built it on top of DirectX 10. At that time, we also created DirectCompute, a new system for high-performance computation on the GPU that became part of DirectX 11. One result of all these updates is that DirectX has a very comprehensive set of features around graphics and GPU computation, but as a side effect, it has also become increasingly difficult to create an app that uses video, 2D graphics, 3D graphics, text, and DirectCompute together.

In Windows 8, the new DirectX 11.1 API is the foundation for hardware acceleration of 2D graphics and text, image processing, 3D graphics and computation, and video. The new API makes it much simpler to mix different types of content in a single scene because that single API now manages all of the GPU resources associated with rendering. This also reduces memory usage by eliminating the redundancy involved in creating multiple graphics device-management objects in app code. In addition, DirectX 11.1 provides a uniform way for apps to access the various capabilities of different graphics hardware. It provides mechanisms for the app to determine what features are available, and then only uses those capabilities. This enables apps to make maximum use of the GPU's capabilities, whether the GPU was designed for long battery life on a tablet, or high-end gaming on a desktop PC.

Diverse graphics hardware

Historically, the expectations for each successive release of Windows have been that both the graphics platform and the graphics hardware capabilities will become richer and higher in performance. This is still true, as the graphics hardware industry continues to develop faster, more powerful GPUs. But in Windows 7, we started to see an inflection point in these assumptions, as the diversity of the hardware broadened with the introduction of mobile, low-power devices.

With Windows 8, this trend towards diverse hardware types is continuing and accelerating, both with new, high-performance graphics cards, and with an increasingly wide range of low-power mobile devices. The diversity of the hardware for Windows 8 will span a broader range than ever before; from graphics hardware that consumes on the order of 1 watt in always-connected tablets all the way up to high-end systems with multiple graphics cards that use a total of 1,000 watts or more. This broadening diversity brings with it new design considerations.

Our goal remains to provide visually compelling, high-performance experiences. With highly mobile devices, the primary power source is a battery, so we also need to maximize battery life. To meet both the performance and power consumption requirements of these new form factors, many of our graphics hardware partners have employed new GPU architectures.

Low-power systems

One of the graphics architectures commonly used in low-power system designs to achieve performance along with great battery life is called “tile-based rendering.” The general concept of a tile-based rendering approach is to have a very high performance (but small) memory cache that the graphics engine uses for rendering. The GPU then renders the screen in sections (or tiles) by repeatedly processing the same set of commands on each tile, rather than the whole screen at once. The intent is to minimize operations that use memory off-chip, therefore keeping power consumption low and performance high. Repeatedly accessing memory off-chip is expensive both in terms of time and power consumption.

To increase the efficiency of these tile-based architectures, we added a number of flags, hints, and new APIs that can minimize the number of times the tiles are rendered. We have incorporated the use of these into the Metro style app development platform to ensure greater efficiency in apps running on graphics hardware that uses a tile-based rendering architecture.

Another way for graphics hardware to reduce power consumption while still achieving great performance is to perform graphics rendering calculations using fewer bits of precision. This allows the GPU to more efficiently structure its data so that it can process more data simultaneously, thus reducing the power needed. For Windows 8, we added new mechanisms for apps to specify the amount of precision needed in their graphical calculations. For example, when doing custom blending of multiple images where the image data is 8 bits per component, the blending computations could be done with 10 bits of precision rather than the default of 32 bits. The reduced precision doesn’t impact image quality, but does reduce power consumption.

Great performance, smoothly rendered

Your browser doesn't support HTML5 video.

Download this video to view it in your favorite media player:

[High quality MP4](#) | [Lower quality MP4](#)

As you can see, we’ve done a lot of work to enable a very fast and smoothly animated user experience in Windows 8. From new ways to measure our progress, to optimizations for mainstream uses of our graphics platform, and new hardware features, we’ve created the best Windows graphics platform yet. And of course, we continue to push the envelope on immersive, three-dimensional gaming, with great performance and new features such as stereoscopic 3D.

From high-end gaming rigs to light-weight, always-connected tablets, Windows 8 supports the broadest range of graphics hardware ever in a single operating system. We hope this post has helped explain some ways in which this work enables a whole new set of rich experiences.

- Rob Copeland

P.S. Thanks to Sriram Subramanian, Dan McLachlan, Kam VedBrat, Steve Lim, and Jianye Lu, for their substantial contributions to this blog post.

Signing in with a picture password

Steven Sinofsky | [2011-12-16T10:30:00+00:00](#)

Picture password is a new way to sign in to Windows 8 that is currently in the Developer Preview. Let's go behind the scenes and see how secure this is and how it was built. One of the neat things about the availability of a touch screen is that it provides an opportunity to look at a new way to sign in to your PC. While many of us might prefer to remove the friction of getting to a PC by running without a password, for most of us, and in most situations this is not the case or is at least unwise. Providing a fast and fluid mechanism to sign in with touch is super important, and we all know that using alpha passwords on touch-screen phones is cumbersome. This post is authored by Zach Pace, a program manager on our You Centered Experience team, and looks at the implementation and security of picture password in Windows 8. Just as a note, you can also use a mouse with picture password too, just by using some click and/or drag actions.

—Steven

The experience of signing in to your PC with touch has traditionally been a cumbersome one. In a world with increasingly strict password requirements—with numbers, symbols, and capitalization—it can take upwards of 30 seconds to enter a long, complex password on a touch keyboard. We have a strong belief that your experience with Windows 8 should be both fast and fluid, and that starts when you sign in.

Other touch experiences in the marketplace have tried to tackle this problem, with the canonical example being a numeric PIN. A PIN is a great solution: Almost everyone has seen or used one before, and a keypad is simple to use with touch. We knew though, that there was room to improve.

A numeric combination often presents a problem for people because the sequences easiest to remember are typically the least secure. Common number sequences—like 1111, or 1234—are troublesome, but PINs that are composed of common well-known personal dates can also be deduced if an attacker has personal knowledge of the person (much of which is not hard to obtain). In such a case, the number being personal to a person can work against its security. We set out to change the paradigm here: we designed a fast and fluid touch sign-in experience that is also personal to you.

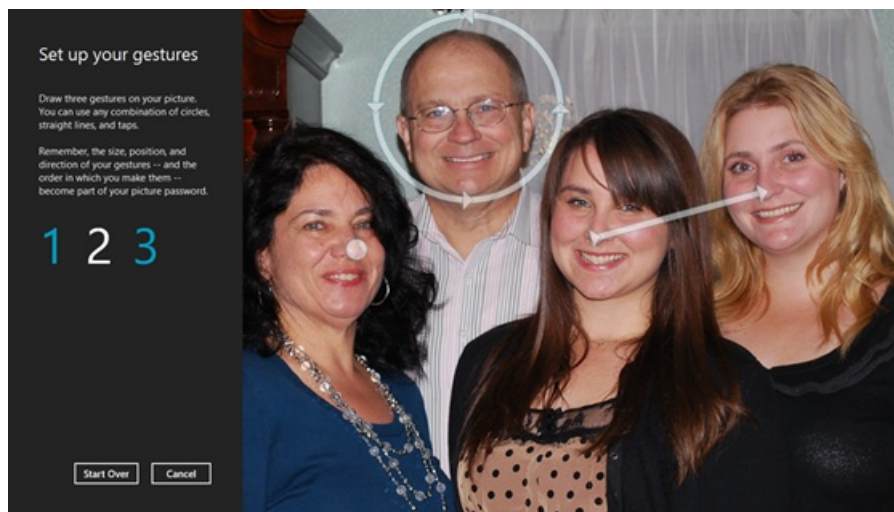
A personal sign-in experience

At its core, your picture password is comprised of two complimentary parts. There is a picture from your picture collection and a set of gestures that you draw upon it. Instead of having you pick from a canned set of Microsoft images, you provide the picture, because it increases both the security and the memorability of the password. You get to decide the content of the picture and the portions that are important to you. Plus, you get to see a picture that is important to you just like many people do on their phone lock screen.



At its core, the picture password feature is designed to highlight the parts of an image that are important to you, and it requires a set of gestures that allow you to accomplish this quickly and confidently. In order to determine the best set of gestures to use, we distributed a set of pictures to a set of study participants and asked them to highlight the parts of the image that were important to them. That's it, no additional instructions. What we found were people doing three basic things: indicating location, connecting areas or highlighting paths, and enclosing areas. We mapped these ideas to tap, line, and circle, respectively. It's the minimal set of gestures we found that allowed people to signify the parts of the image most important to them.

There's also an attribute inherent to circle and line gestures that adds an additional layer of personalization and security: directionality. When you draw either a circle or a line on your selected picture, Windows remembers how you drew it. So, someone trying to reproduce your picture password needs to not only know the parts of the image you highlighted and the order you did it in, but also the direction and start and end points of the circles and lines that you drew.



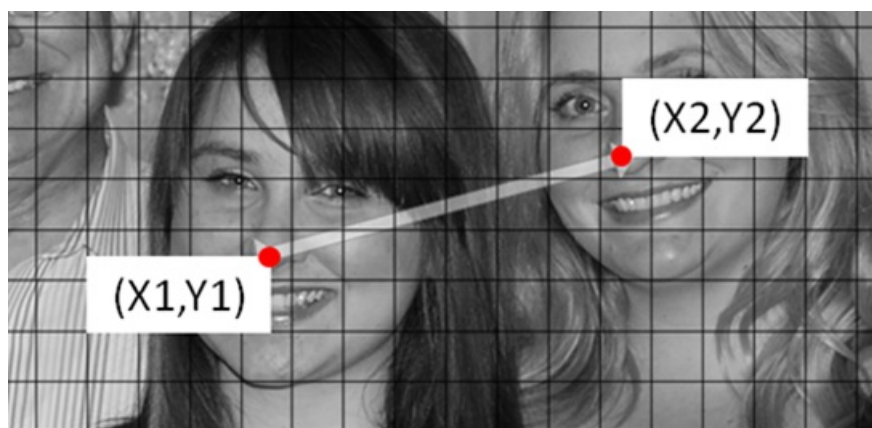
We also researched using freeform gestures. When we explored the concept, both with design iterations and research, we found the major pitfall of such a system: the time it takes to sign in. As I mentioned above, we wanted a solution that was faster than a touch keyboard. Throughout the evolutionary process of this feature we used the time taken to sign in using a touch keyboard as a benchmark to judge the success of our methods. We found that when people were allowed to use freeform gestures, it took them consistently longer to sign in. They were slowed down by the concept, feeling that they needed to be unnecessarily precise and trace fine details in an image.

Because people were highlighting areas instead of fine detail, we found that using a limited set of gestures was on average more than three times as fast as the freeform method. We also found that with repeated use, people using the gesture set were consistently able to complete the task in under four seconds, compared to an average of 17 seconds for the freeform model. After continued use of the freeform method, we found many participants asked to change their freeform gestures, picking simple lines and locations instead.

How it works

Once you have selected an image, we divide the image into a grid. The longest dimension of the image is divided into 100 segments. The shorter dimension is then divided on that scale to create the grid upon which you draw gestures.

To set up your picture password, you then place your gestures on the field we create. Individual points are defined by their coordinate (x,y) position on the grid. For the line, we record the starting and ending coordinates, as well as the order in which they occur. We use the ordering information to determine the direction the line was drawn in. For the circle, we record a center point coordinate, the radius of the circle, and its directionality. For the tap, we record the coordinate of the touch point.



When you attempt to sign in with Picture Password we evaluate the gestures you provide, and compare the set to the gestures you used when you set up your picture password. We take a look at the difference between each gesture and decide whether to authenticate you based on the amount of error in the set. If a gesture type is wrong—it should be a circle, but instead it's a line—authentication will always fail. When the types, ordering, and directionality are all correct, we take a look at how far off each gesture was from the ones we've seen before, and decide if it's close enough to authenticate you.

As an example, let's take a look at the tap gesture. The tap is the least complex of the three gestures both in number of unique permutations and in the subsequent analysis. When considering whether the spot that you've tapped matches a reference spot, our scoring function compares the distance between the gesture you recorded as part of your picture password and the one that you just performed. The score decreases from 100% for a perfect match to 0% when sufficiently far away. Points match when the score is $\geq 90\%$. Here is a visual representation of the scoring function for a point in the immediate vicinity of a 100% match:

70%	77%	82%	85%	86%	85%	82%	77%	70%
77%	84%	89%	92%	93%	92%	89%	84%	77%
82%	89%	94%	97%	98%	97%	94%	89%	82%
85%	92%	97%	100%	100%	100%	97%	92%	85%
86%	93%	98%	100%	100%	100%	98%	93%	86%
85%	92%	97%	100%	100%	100%	97%	92%	85%
82%	89%	94%	97%	98%	97%	94%	89%	82%
77%	84%	89%	92%	93%	92%	89%	84%	77%
70%	77%	82%	85%	86%	85%	82%	77%	70%

The area that is scored a match is a circle of radius 3. For any specific tap, a total of 37 (X,Y) locations will return a match. We perform similar calculations for the variables associated with lines and circles.

Security and gesture count

When we took a look at the number of gestures that would be required to use picture password we considered security, memorability, and speed. We sought to balance these often competing attributes to achieve an optimal user experience that would also be secure to use. In order to determine the appropriate gesture count that would meet our security goals, we compared picture password with different authentication methods, namely PIN and plain text password.

The analysis of the number of unique PINs is trivial. A 4-digit PIN (4 digits with 10 independent possibilities each) means there are $10^4 = 10,000$ unique combinations.

When looking at alphanumeric passwords, the analysis can be simplified by assuming passwords are a sequence of characters comprised of lower case letters (26), upper case letters (26), digits (10), and symbols (10). In the most basic case, when a password is comprised strictly of n lower case letters, there are 26^n permutations. When the password can be any length from 1 to n letters, then there are *this* many permutations:

$$\sum_{i=1}^n 26^i$$

For instance, an 8-character password has 208 billion possible combinations, which to most people would seem amazingly secure.

Unfortunately, the way most users pick passwords is far from random. Left to their own devices, people use common words and phrases, names of family members, and so on.

In this scenario, let's assume the user composes their password from all but two lower case letters, one upper case letter, and one digit or symbol; however, the upper case letter and digit/symbol can appear in any position of the password. The number of unique passwords is then:

$$26^{n-1} \cdot 20 \cdot \frac{n!}{(n-2)!}$$

The following table illustrates how the size of the solution space varies with password length and various character set assumptions.

Password length	Unique passwords
1	n/a
2	n/a
3	81,120

4	4,218,240
5	182,790,400
6	7,128,825,600
7	259,489,251,840
8	8,995,627,397,120

When considering picture password, we can conduct a similar analysis for each of the gesture types. The information in the tables below accounts for both unique gesture positions and the leniency of our recognition algorithm.

For the simplest gesture, the tap, the number of unique gesture sets as a function of number of taps is as follows:

# of taps	Unique gestures
1	270
2	23,535
3	2,743,206
4	178,832,265
5	15,344,276,658
6	1,380,314,975,183
7	130,146,054,200,734
8	13,168,374,201,327,200

The circle gesture has more complexity than a tap, but less than a line. In an attempt to quantify the relative security of a circle, we can assume that an attacker knows the radii is guaranteed to be between 6 and 25 (reducing the work to guess a circle gesture), we will further assume that both X and Y coordinates are known to be between 5 and 95. This makes the potential solution space for a hacker to explore to be as follows:

$$(95 - 5 + 1)^2 \cdot (25 - 6 + 1) \cdot 2 = 331,240$$

As a function of number of circles, the number of unique gesture sets is as follows:

# of circles	Unique gestures
1	335
2	34,001

3	4,509,567
4	381,311,037
5	44,084,945,533
6	5,968,261,724,338
7	907,853,751,472,886

The most complex gesture of the three is the line. A line is comprised of two points on a normalized 100 x 100 grid, and an ordering of those points. This nominally results in 100 million possible lines; however, lines must be at least 5 units long, so the number of unique lines is actually 99,336,960. Unlike attempts to guess circles where hackers can make simplifying assumptions that significantly reduce the solution space, there are not any similarly obvious reductions for lines. Lines could just as easily go from edge to edge of the screen as they could be very short segments. The number of matches in the case of the line is as follows:

# of lines	Unique gestures
1	1,949
2	846,183
3	412,096,718
4	156,687,051,477
5	70,441,983,603,740

Now that we understand the security of individual gestures, this data can be combined to assess sets containing multiple gestures. This can be done by summing up the unique gestures for all three gesture types for the specific gesture length n and raise it to the n^{th} power. This results in the table below, which compares picture password to both PIN and alphanumeric password methods.

Length	10-digit PIN	Simple a-z character set password	More complex character set password	Multi-gesture picture password
1	10	26	n/a	2,554
2	100	676	n/a	1,581,773
3	1,000	17,576	81,120	1,155,509,083
4	10,000	456,976	4,218,240	612,157,353,732
5	100,000	11,881,376	182,790,400	398,046,621,309,172

6	1,000,000	308,915,776	7,128,825,600
7	10,000,000	8,031,810,176	259,489,251,840
8	100,000,000	208,827,064,576	8,995,627,397,120

As you can see, the use of three gestures provides a significant number of unique gesture combinations and a similar security promise to a password of 5 or 6 randomly chosen characters. Additionally, using three gestures ensures a Picture Password that is easy to remember and quick to use.

In addition to the number of unique combinations, we've increased security of the feature by introducing two safeguards against repeated trial attacks. Similar to the lock out feature on phones using PIN, when you enter your picture password incorrectly 5 times, you are prevented from using the feature again until you sign in with your plain text password. Also, picture password is disabled in remote and network scenarios, preventing network attacks against the feature.

To be clear, picture password is provided as a login mechanism in addition to your text password, not as a replacement for it. You should be sure to have a good hint and use safeguarding mechanisms for your text password, which you can still always use to sign in (the sign-in screen provides a one-click mechanism to switch between all available password entry methods).

Securing against smudges

We've also taken some practical considerations to protect you if you use Picture Password. People are often concerned with the smudges left behind on a touch screen and how easy or hard it would be to divine your password based on those markings. Because the order of gestures, their direction and location all matter, it makes the prospect of guessing the correct gesture set based on smudging very difficult even in the completely clean screen case, let alone on a screen that sees regular touch use.

The potential threat here is that smudges left from signing in may yield clues as to the authentication sequence. We can compare three way of logging in—touch keyboard, a four-digit PIN, and picture password—to compare the ease of guessing the sign-in sequence. Let's assume the worst case scenario:

1. The user cleans their screen to an absolutely mirror shine.
2. The user touches exactly the minimum places necessary to authenticate.
3. The user then walks away from their machine without touching it further.
4. The attacker steals the tablet and can with 100% accuracy see every gesture used for authentication.

Obviously, this is rather unlikely, but this scenario allows us to compare and contrast the three forms of authentication and their relative vulnerability to this sort of attack.

A PIN will leave a smudge in a known location for each digit used in the code. If there are n digits in the PIN, and all digits are unique (the hardest to deduce case), there will be $n!$ possible ways of ordering the PIN. For a typical 4-digit PIN, this is 24 different combinations.

For an on-screen keyboard, there are also $n!$ ways of ordering an n -character password. For compliant passwords, a person will typically use the Shift key (or another button) to select alternate character sets. This key press will, of course also be visible to the attacker, but it does not indicate when in the sequence the Shift key was utilized. If we make the simplifying assumption that there is only one shifted key in the password, then there are $n! \cdot n$ possible passwords to consider.

Gestures also have $n!$ orderings. For every circle and line used in the gesture set, the number of permutations increases by a factor of two. If all gestures are circles or lines, then the possible set of permutations is the same as a password that uses the Shift key, $n! \cdot 2^n$.

The following table summarizes the number of permutations for each of these methods for various sequence lengths:

Length	PIN	Password	Password with Shift	Tap-only gestures	Line and circle gestures
1	1	1	1	1	2
2	2	2	4	2	8
3	6	6	18	6	48

Again, this is assuming a completely clean screen with only the gestures visible via smudging. If we consider a scenario where the attacker cannot gain any useful information from smudging—either because the machine is very heavily used (and smudged) or because it is mouse and keyboard only—the chances of guessing the correct sequence becomes even more remote. With our three gestures types, directionality, and the requirement that the sequence be at least three gestures long, the possible number of gesture combinations sits at 1,155,509,083, as discussed above.

The final attack that we considered involves points of interest on an image, or areas that people may commonly choose when presented with an image. Even though the research we did showed this kind of attack to be extremely unreliable—the areas people chose and the kind of gestures they drew upon them correlated very poorly in the lab—we can analyze such an attack by assuming a given picture has m points of interest. If the user is free to use any combination of taps, circles, and lines, then the total number of permutations is $(m \cdot (1 + 2 \cdot 5 + (m - 1)))^n$, where n is the length of the picture password. This yields the following number of possible combinations:

Points of interest

Length	5	10	15	20
1	75	200	375	600
2	5,625	40,000	140,625	360,000
3	421,875	8,000,000	52,734,375	216,000,000
4	31,640,625	1,600,000,000	19,775,390,625	129,600,000,000

Assuming the average image has 10 points of interest, and a gesture sequence length of 3, there are 8 million possible combinations, making the prospect of guessing the correct sequence within 5 tries fairly remote.

Although we're very happy with the robustness of a picture password, we know that there are a variety of businesses for which security is paramount, and anything less than a full password is unacceptable. As such, we've implemented group policy that gives a domain administrator the freedom to choose whether picture password can be used. And of course, on your home PC, picture password is optional as well.

When we started the process of designing picture password, we knew that we wanted a sign-in method that was fast, fluid, and personal to each and every user of Windows 8, but still had a robust security promise. Through our research and refinement of both the experience and the concept, we believe we've hit on a method of signing in that's secure but also a lot of fun to use. We love picture password and the additional personal flavor it brings to Windows 8, and we hope you do too!

-- Zach

Your browser doesn't support HTML5 video.
 Download this video to view it in your favorite media player:
[High quality MP4](#) | [Lower quality MP4](#)

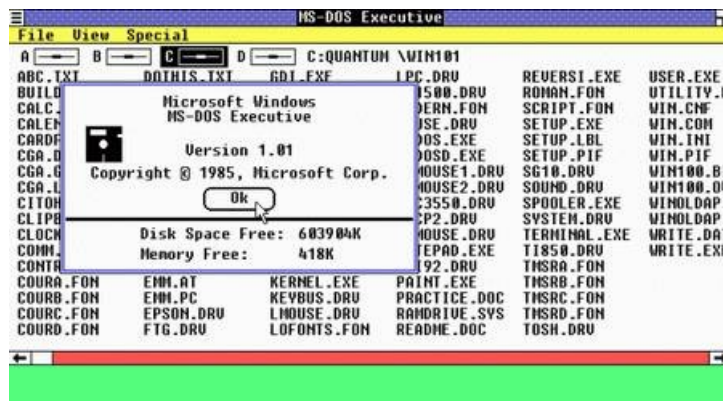
Simplifying printing in Windows 8

Steven Sinofsky | [2012-07-25T10:30:00+00:00](#)

Printing is one of the most common things we do with our PCs even as we read and work with more online resources. We set out to simplify and improve this common operation—working with partners across the ecosystem to deliver these improvements in Windows 8. This blog post was authored by Adrian Lannin, a lead program manager on the Printing team.

--Steven

Of all the peripheral devices that you can connect to your Windows PC, printers are one of the most popular, and have been supported for the longest time. In fact, Windows 1.0 (shipped in 1985) supported “a number of printers and plotters” and included a “Print Spooler [which] allows the user to work on one file while printing on another” according to the Windows 1.0 Press Kit. The screenshot of Windows 1.0 below shows the files included with that version of Windows – Epson.driv, lots of font files, and the print spooler process. Some parts of the print system are older than the people who work on it. ??



Over the years, the print system has evolved into a complex architecture that supports printing to a huge variety of printers, and can scale from a simple \$50 inkjet at home to a high-availability print server hosting thousands of print queues for hundreds of thousands of users, driving printers that cost tens of thousands of dollars each.



The print system touches many layers and facets of Windows. It shows UI, and it hosts drivers that also show UI. It performs intensive graphics operations, since printing is essentially re-drawing your on-screen content onto paper. It encompasses lower-level communications, mainly USB or network (the majority of printers bought in the US today are network-capable, but our telemetry data tells us that over 75% of the printers installed with the Windows 8 Consumer Preview are plugged into a USB port). The print system needs to scale to very large, mission-critical deployments in large businesses but also run efficiently on small systems.

In this blog post I'm going to talk about the work that we've done in Windows 8 to re-imagine how the print system can best provide [good device support](#) to our customers. I'll show you how it works on ARM-based PCs and in Metro style apps. And I'll talk about what we've done to ensure that the maximum number of existing printers "just work"—whether you're accessing them from the desktop, from a Metro style app, or on a device running Windows RT.

Reimagining the print system for Windows 8

In Windows 8 we've introduced a new printer driver architecture, which we call version 4, or v4. The v4 architecture produces smaller, faster printer drivers, and it supports the idea of a print class driver framework—a system that allows people to install their printers without having to locate a driver for that device, in many cases.

As you've probably guessed, V4 is the fourth iteration of the printer driver architecture in Windows. V3 was the architecture used from Windows 2000 to Windows 7, and it's actually still fully supported in Windows 8 for device compatibility reasons. So if you only have an existing driver available for your current printer, then it should still work in Windows 8. Versions 1 and 2 were the driver architectures for Windows 1.0 through

Windows ME.

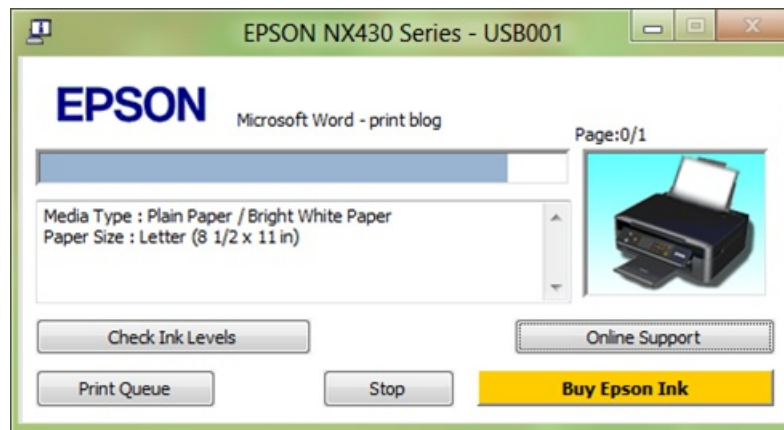
Before I explain how the print system works, I'd like to talk about some of the requirements that we worked to address with the Windows 8 print system.

Printing from Metro style apps

One of the things that we needed to figure out was how to give Metro app developers the ability to print. Printing from win32 applications requires knowledge of graphics programming, either GDI (Graphics Device Interface) or XPS ([XML Paper Specification](#)). When we looked at how we could make printing possible from Windows 8 apps, we completely reinvented how we enable printing from the Windows Runtime, and we made printing very easy to use from HTML5/JavaScript and XAML/C# apps.

Integrating printing into Metro style apps

Printing from a Metro style app should naturally be a Metro style experience. I'm sure that when most of you have printed something, you've seen a little applet pop up to tell you that the printer is out of paper, or to offer you the opportunity to purchase ink.



These pop-ups are very common with inkjet printers. Some pop up only when relevant (you have low ink), while others pop up every time you print. These pop-ups come from the printer driver software itself, and they are all desktop UI, of course. But when printing from the Metro style Photos app, for example, we don't want you to have to switch over to the desktop just to see UI that tells you that printing is in progress.

Printing in Windows RT

Printer drivers have evolved over time to include a lot of functionality—some install services, some install numerous little applications, and many are now quite large. The v3 printer driver model in use since Windows 2000 evolved into a highly complex and highly extensible model, which allowed printer manufacturers a lot of freedom in what is installed with their driver software. When we thought about how this would work on some of the devices that are going to run Windows RT, we knew that we had to make some significant architectural changes. We really wanted to ensure that we didn't negatively impact ARM systems by running unnecessary services, and we wanted to reduce system resource usage, while still providing support for as many devices as possible.

Lots of printers supported, far fewer drivers

There's a huge diversity in printer capabilities, and Windows supports a vast range of printers. In Windows 7 and earlier versions of Windows, each of these printers required a specific driver in order to work (there are some exceptions, such as universal printer drivers, but these tend to be large and resource hungry). This meant that the number of drivers that we included with Windows (we call these in-box drivers) was very large so as to provide good support. Of course, we have many more drivers on Windows Update, but we believe that it's important to have a core set of in-box drivers that support popular devices, so we can still provide a good printing experience for people who can't or won't download a driver from Windows Update. In-box drivers are essential for Windows RT—in fact, it uses only in-box printer drivers. The challenge here is to get a relevant set of printers supported, but to also reduce the resources required to accomplish this.

Another interesting challenge in supporting lots of printers is that the support gets stale over time. The set of drivers included in Windows 7, for example, provided excellent support for devices released in 2008 and 2009, but as new devices were released over the years, and time went by, the set of drivers in Windows 7 became less relevant. One big challenge in Windows 8 then, is to ensure that Windows provides a high level of support for lots of printers, including ones that haven't even been released yet.

Printer sharing

Anyone who has administered a print server can tell you that getting the correct drivers installed to support sharing is the most time-consuming part of managing a print server. Some of these difficulties you might encounter when you're trying to share a printer at home too, especially if you have both 32-bit and 64-bit versions of Windows. This becomes tricky because the print "server" (which just means the PC that the printer is connected to – not actually Windows Server) has to provide the drivers to the clients that want to print to the shared printer. In Windows 7, we

used HomeGroup to address this problem, and it works well much of the time. However, the requirement to load drivers for each Windows architecture becomes more problematic when you think about printing from Windows RT.

Although we expect that most people who print from Windows RT devices are likely to print to wireless printers, we didn't want to totally exclude the possibility of printing to a USB printer. On the other hand, we didn't want to increase the complexity of printer sharing by requiring people to add drivers for 32-bit clients, 64-bit clients, and Windows RT clients! So, with the v4 model in Windows 8, we developed a new way to share printers that doesn't rely on putting client drivers onto the print server.

The print system in Windows 8

Applications enable you to create and view content. The purpose of the print system is to provide these apps with the means to print your content to any installed printer without having to worry about what particular device is installed. I'm going to talk a little about how the app prints, and go into more detail about how we get the content onto a printed page.

Creating printable content

For apps, adding printing support is quite straightforward. The content that you want to print from an app is in a format that the app specifies. For Metro style apps, this will often be HTML5 or XAML, but for Win32 apps such as Word or Photoshop, the content is in a format specific to each particular app.

So when you want to print from an app to your printer, one of the things that the print system needs to do is to translate the content from the app's format to the format that the printer understands. Unfortunately, printers don't all understand the same formats (not even close!) so this turns into quite a bit of work.

To give a real example, an app such as Word uses the GDI graphics system to draw the content both to the screen and to the printer. When possible, the print system uses a high quality intermediate format called XPS ([XML Paper Specification](#)) as its internal content format; we convert the content from Word into XPS. We chose to use XPS as the foundation of our print system because it is a very flexible format and is just like electronic paper. It supports high-fidelity color, and since it's an XML-based description with no executable code embedded, it's great for archival purposes and it is secure compared to other options. In addition, Microsoft has worked with ECMA International ([European Computer Manufacturers Association](#)) to make it an open standard (ECMA standard TC46, OpenXPS). Both the desktop viewer and the Reader app can display OpenXPS. I "print" all my receipts from online purchases as XPS files.

Once the content is being managed by the print system, it is then converted to the format that the printer understands (if necessary; there are lots of printers that understand XPS directly) and the print system sends this to the printer with the correct options set, and the job prints.

In Windows 8, we have a distinct improvement to this story because all Metro style apps use Direct2D as their basic drawing format, and Direct2D and XPS share the same XML-based graphics "language." So in another real-world example, the Reader app uses Direct2D to render its content onto the screen. It also uses Direct2D to render the same content to the print system. Reader's content can easily be submitted to the print system as XPS, without any costly conversion from GDI.

If the app requires a print layout that is different from the screen layout, then it can do this using style sheets or XAML. This means that you don't have to "click here for a printer-friendly version of this page." If you have a printer that supports XPS, then the path from the app to the printer involves no conversions at all, and printing is extremely fast!

Now that you understand in broad terms how an app sends print information to the print system, I'm going to talk about what the system does with that, the services it provides, and what else has changed in Windows 8.

Supporting lots of printers

One of the big benefits that Windows provides to apps is that it abstracts the specific printer from the app, so that the app's programmer doesn't have to worry about what printer you've installed. Windows supports tens of thousands of printer models in total, including printers that are supported by drivers available via Windows Update or the manufacturer's website. When we see printers that don't work, this is often because the manufacturer has chosen to block the installation if they don't recognize the version of Windows that their software is being installed on. We work with printer manufacturers to get these packages updated, but this does take some time.

Ideally, when you plug a new printer into Windows, it just works, without your needing to go off and find drivers.

So how do we make that happen? In the past we've shipped a lot of printer drivers in Windows. Vista contained about 4500 drivers, and Windows 7 contained about 2100 drivers. Even though Windows 7 had half as many drivers as Vista, it provided better market coverage, by which I mean that there was a better chance that it had a driver for the more popular printers. Why is this? There is an incredible diversity of printers in use. In Vista, we supported a lot of devices that were old and no longer in popular use, and so the relevance of the set of devices supported was not as good as in Windows 7.

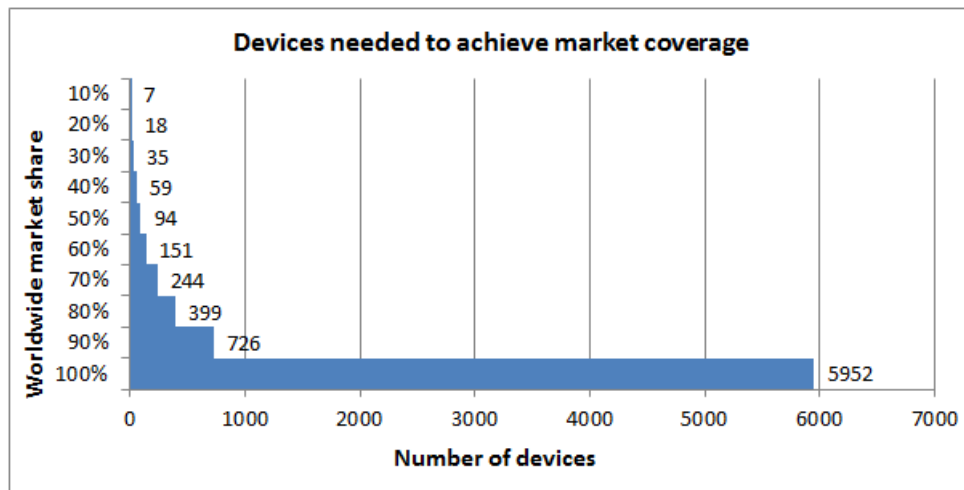
As an aside, the other thing we do when we release a new version of Windows is to take the drivers that were in the previous version and post them to Windows Update, so that even though these devices may be dropping in popularity, it's still possible for people to automatically get the device working by just plugging it in.

Here's a photo I took of one of the benches in one of the printer labs (we have several) where we test that this all works. You can see several small inkjets and laser printers from different manufacturers. Luckily for my ears, we don't test with dot matrix printers very often these days.



People tend to keep printers for 5-7 years on average, so when we want to add support, we have to think "what devices are people using? Which were the most popular devices over the past several years, and what will be the most popular in the future?" This last part is tricky because, pretty soon after we release Windows, the printer manufacturers will release devices that we didn't know anything about. This means that over time, the set of devices that we support in any particular version of Windows becomes stale.

We know that at any given moment, about 100 specific printer models make up about 50% of the installed base. If we want to support 75% of the models being used today, then we need to support about 300 models. The diagram below illustrates this.



To get to 95%, we need over 1000 models supported. But the problem is even harder because the printers that make up this set of 100, or 300, or 1000 changes all the time. The 100 printers that represent 50% of the market *today* are not the same 100 printers that will represent 50% *next week*, or next month, and especially not next year. Every day, many people buy and install new printers.

As I mentioned above, we basically took a brute-force approach to solving this in the past. We have representatives from the major printer manufacturers working directly with Microsoft, sitting in offices in Redmond, working to check their source code into Windows. They would create a completely new set of in-box drivers for each new release of Windows. This just isn't very efficient.

In Windows 8, we took a radically different approach, and have stopped shipping lots of printer drivers with Windows. Instead, we built a *print class driver framework*. This framework is extensible, as it supports printing to existing devices, but it also allows manufacturers to include support for new devices, even those that have not yet been designed.

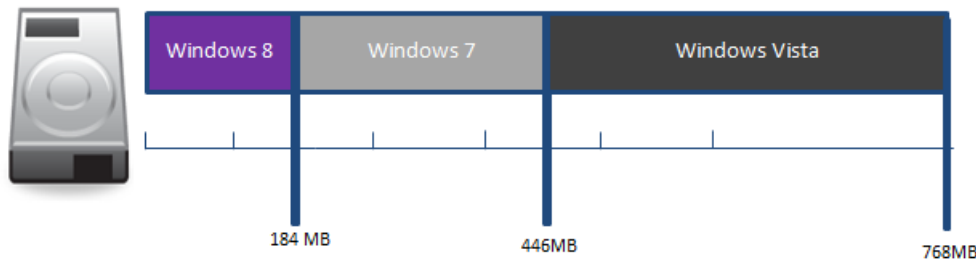
With a print class driver framework, we can get closer to giving you an experience like driverless printing, where you don't have to actually go and find a driver, but instead the printer just works with the Windows printing system. A true driverless printing experience requires changes to how most printers are designed, and the print class driver framework provides support for this idea, but we also feel that it's very important to provide

as much support for existing devices as possible.

With the ability to support new and planned printers, the number of printers that are supported by the Windows 8 print class driver framework will actually increase over time.

Besides the great progress in increasing the number of devices covered, we have also been able to reduce the resources that we use to achieve this coverage.

First, we reduced the amount of disk space needed to support printers and imaging devices from 768MB in Windows Vista, to about 184MB in Windows 8. This number is an average across different editions and architectures of Windows 8. The following graphic illustrates the reduction in space used since Windows Vista.



Comparison of disk space needed to support printers and imaging devices in Windows 8, Windows 7, and Windows Vista

In addition, the reduction in disk space used has been accompanied by an increase in the relevance of the devices supported directly by Windows. The following table summarizes how the relevance of the inbox coverage has increased, while disk use has decreased.

	Approximate number of devices supported <i>in-box</i>	Approximate installed base	Disk space used
Windows Vista	4200	55-60%	768 MB
Windows 7	2100	60-65%	446 MB
Windows 8	2500	70% at release growing to 80%	184 MB

This is a huge improvement in Windows 8, and this reduction in space used directly translates into more available storage space for users of hardware with limited storage capacity, which we expect will be a characteristic of some Windows RT computers.

The Windows 8 printer driver model also allows us to focus our manufacturing partners on a set of code that will not change as much from one version of Windows to the next, so we will be able to more usefully spend those resources on improving quality and performance, instead of constantly repopulating the driver set.

Print class driver architecture

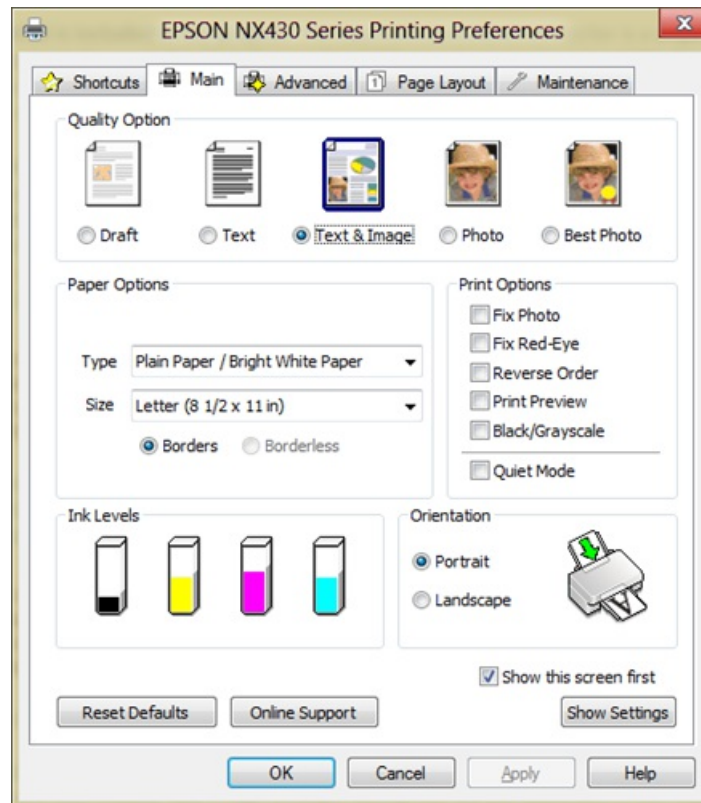
Besides wanting to create an architecture that supports the needs of Windows RT and Metro style apps, we wanted to make sure the model would also work with existing devices, and would utilize technologies that were familiar to printer manufacturers, so that it would be easier for them to implement the new driver technology.

A printer driver does several key things:

- **Configuration** allows the user to change settings, translating the intent to turn on double-sided printing (for example) into the specific command that the printer needs for this. Configuration is presented to the user through the user interface.
- **Rendering** translates the printed content from the format that Windows print system uses into the format that the printer understands. In some cases, the printer may directly understand the native Windows print format (XPS), so for those devices, there is no work to do here, unless they want to do extra rendering (doing multiple pages per physical sheet of paper is an example of this case). The part of the driver that does rendering is called the *render filter*.
- **Events** allow the printer to inform the user that something has happened – a job has completed, there has been a paper jam, or the printer is out of ink.

Configuration UI

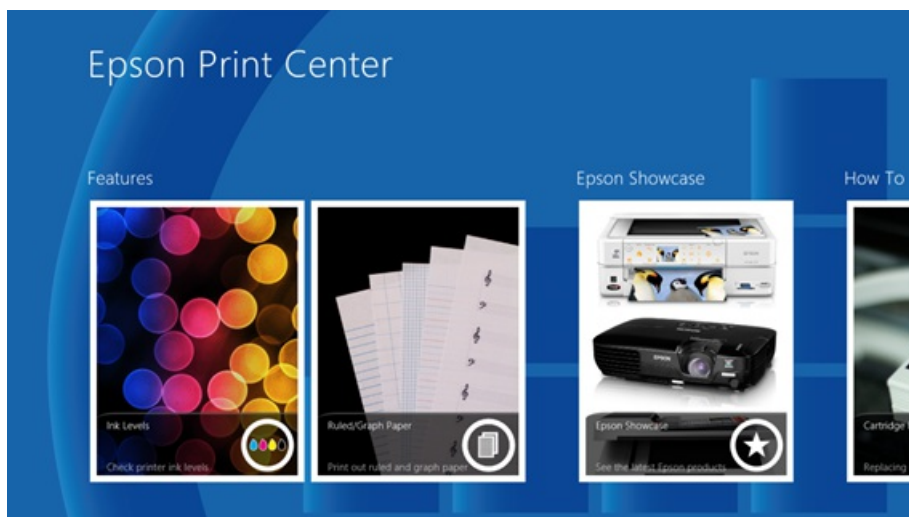
One big change between the old driver model and the Windows 8 driver model is in how the UI is provided. In the old printer driver model, the configuration UI was built into the driver. Here's an example of some typical printer UI (taken from the Epson NX430 that is currently on my desk).



As I mentioned earlier, we needed to find a way to enable the display of Metro style UI when people wanted to change printer settings.

In the Windows 8 driver model, the manufacturer's UI is completely separate from their driver. This is a much better architectural decision for many reasons: The UI to control the printer is now an app that can be invoked when printing from Metro style apps or desktop apps. This allows printer manufacturers to present you with a much richer experience – imagine access to video showing you how to set up your printer or install an ink cartridge.

Here is an example of a Metro style app that Epson has developed for the Epson NX430:



You can see that this UI has all the hallmarks of a Metro style app, but for your printer. It includes an attractive view of the ink levels of the printer, and is much easier to use, especially on touch-screen devices.

Windows will automatically show you the correct type of UI – desktop printer UI when you're printing from desktop apps, and Metro style UI when you're printing from Metro style apps.

If the manufacturer hasn't provided any configuration UI for their device, then Windows provides some standard UI that you can use with any printer. However, when the printer manufacturer has decided to invest in providing a customized experience for their device, they can provide an app that replaces the standard Windows UI. Then, when you decide to alter the configuration of the device, or when the device configuration changes during printing (e.g. paper jam), then Windows will display the manufacturer's customized app to you instead.

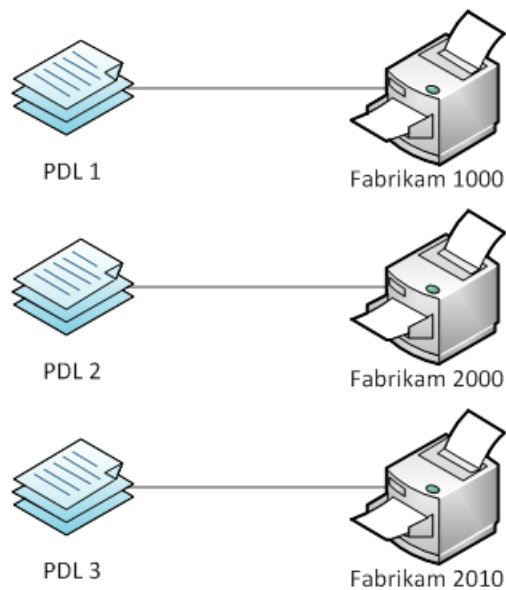
Rendering

One of the most important functions of a printer driver is to take the content that the app produces when you ask it to print, and convert that into something that the printer can understand. This was one of the most challenging areas of building the Windows 8 print class driver, so let's look at it in a bit more detail.

As described above, desktop apps like Word or Photoshop use graphics commands to draw their content onto the screen or the printer. When they do this, the print system receives the content, converts it into XPS if necessary, and then calls the printer's driver (or more specifically, the *render filter* part of the driver) to convert the content into the right format. This is sent to the printer and your file is printed out.

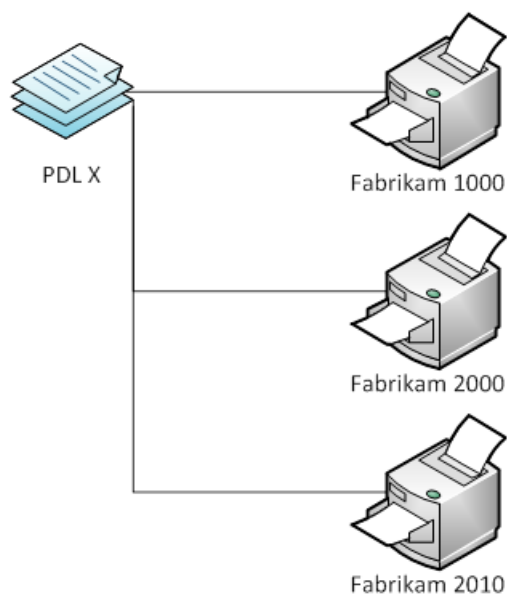
Probably one of the largest challenges in supporting a very wide range of printers is in dealing with the rendering portion of the problem. Some of the more expensive printers support standard "page description languages" or PDLs, such as PostScript, PCL, and XPS. However, less expensive, consumer-focused devices are manufactured with cost savings in mind, and many of these support proprietary methods of sending the page information to the printer. Some manufacturers may have only a few languages that they use across their product line, but others may tweak the language from one model to the next, trying to get the most out of their printer hardware.

This leads to a 1:1 mapping between printer driver and printer hardware.



Imagine each PDL as being a complete printer driver and it's easy to see that increasing support involves a steadily increasing number of drivers. Of course, this is a bit of a simplification and it is possible to create a driver that supports a number of devices, and we have often seen drivers that support a series of printers. But the key point here is that Windows 7 and earlier versions of Windows didn't do anything to support this design approach.

The printer driver model in Windows 8 supports the idea that a PDL (or driver) can be associated with multiple devices.



We've been working with our printer manufacturer partners to have them include an identifier in their device that describes how they are supported more generically. We call this a *compatible ID*. So for example, if a device has a compatible ID that says that the device supports XPS, then the

print system knows that it doesn't need to find a model-specific driver for that device, it can just install a generic XPS driver for the device. Windows understands that the device is a generic XPS printer and can treat it that way. Of course, Windows also understands that this is a Fabrikam 1000 printer (or whatever), so if there is a model-specific driver, then Windows will install it. But if there is no driver available, then Windows can still print to the printer using the class driver.

So in this example, we have a set of render filters in-box as part of the class driver model, and these can be installed for any device that implements a matching compatible ID. The logical extension of this idea is that it's quite possible for future devices, devices not yet designed or built, to be compatible with the print class driver in Windows 8. We've been working with the printer manufacturers and they all plan to implement compatible IDs in their devices (many already do). Because of this capability, the number of printers that are supported in Windows 8 will increase over time, rather than decrease, and more and more people will get the experience of being able to use their printers instantaneously from Windows 8 without the need to go and find a driver.

But what about all the devices that have proprietary rendering languages? The print class driver supports that model too, but with the disadvantage that we do need to have a separate rendering filter for each small set of models that speak each unique language. There is no way around this, and in Windows 8 we've taken a number of filters that address a set of popular models. However, once again, we've been working with the printer manufacturers to improve this position, and we expect to see manufacturers produce printers that can more easily utilize the class driver in the future.

Printing from Windows RT

The reduction in the resources used by the print class driver contributes directly to a smaller footprint for Windows, which is especially valuable on Windows RT. In addition, the V3 printer driver architecture was highly extensible and had evolved over many years into a model that encouraged the development of large, complex printer drivers. Some drivers install services that run all the time, exhausting battery power and using processor time. I've seen some drivers that support only one device but that are larger than the complete printer driver set in Windows 8!

The need to support printing in Windows RT, and a general desire to make printing more efficient, led us to develop an architecture that more tightly controls what the driver can do. I've already mentioned that the UI portion of the print experience is now a completely separate component, an app instead of being part of the driver. This means that it's also optional, and drivers will work well with the standard Microsoft printing UI. We've also simplified the driver architecture to be more power-efficient, by removing service dependencies and reducing the likelihood that additional software will be included with the driver.

With the Windows 8 driver model, we also made significant changes to how printer drivers are installed. In Windows 7 and earlier versions of Windows, all printer drivers are stored in the "Driver Store,"—sort of like a database for all types of drivers. When you plugged in a printer, we would find the correct driver in the driver store, and copy it to a special location where the spooler could use it with your printer. In Windows 8, we have eliminated this extra copying, which removed quite a bit of disk I/O. The print spooler now just knows how to find the driver in the driver store.

For a real world example, we compared the installation times for an Epson Artisan on Windows 7 versus Windows 8 (using a relatively small driver on Windows 7): the install time on Windows 7 was 14 seconds, compared to under 2 seconds on Windows 8.

Final thoughts

As you can see, the Windows 8 printer driver architecture is big step forward. It provides good support for a lot of the printers that people already own, and supports future devices with a small, fast, built-in class driver framework. The performance is great and the disk footprint is small.

We're look forward to your feedback!

-- Adrian Lannin

Releasing Windows 8 – August 1, 2012

Steven Sinofsky | [2012-08-01T09:00:00+00:00](#)

Today marks an important milestone in the Windows 8 project. The Windows 8 team is proud to share with you that a short while ago we started releasing Windows 8 to PC OEM and manufacturing partners. This means our next milestone will be the availability of exciting new models of PCs loaded with Windows 8 and online availability of Windows 8 on October 26, 2012.

Back when we first demonstrated Windows 8 in May 2011, we described it as “reimagining Windows, from the chipset to the experience,” and that is what Windows 8 (and Windows RT) represents for both Microsoft and partners. The collective work: from the silicon, to the user experience, to new apps, has been an incredibly collaborative effort. Together we are bringing to customers a new PC experience that readies Windows PCs for a new world of scenarios and experiences, while also preserving an industry-wide 25-year investment in Windows software.

We continue to be sincerely humbled by the breadth of participation in our pre-release testing. The previews of Windows 8 (Developer, Consumer, Release) have been the most widely and deeply used test releases of any product we have ever done. Over 16 million PCs actively participated in these programs, including approximately 7 million on the Release Preview that started 8 weeks ago. The depth and breadth of testing validate the readiness of Windows 8 for the market.

The openness of the previews presents a unique perspective on product development, and we’re deeply committed to the transparency of the preview process. No product used by so many people in so many different ways is developed “out in the open” like Windows 8 has been. This blog, the forums, and the preview releases form an important part of the development process. Major changes have been made at each milestone and as we promised, the final release (build 9200, for those tracking) contains many promised refinements. We are humbled by the responsibility of meeting the needs of such a diverse set of customers and enthused by the deep level of participation in the pre-release process.

While we have reached our RTM milestone, no software project is ever really “done.” We will continue to monitor and act on your real world experiences with Windows 8—we’ve used the preview process to test out our servicing and we have every intent of doing a great job on this next important phase of the product. Hardware partners will continue to provide new devices and improve support for existing devices. PC makers no doubt have quite a bit in store for all of us as they begin to show off PCs specifically designed for Windows 8.

With improvements in fundamentals, enhanced storage and connectivity, newly architected subsystems, the “fast and fluid” user experience, and the WinRT platform (to name a few), Windows 8 has literally *thousands* of new features. We did a record number of blogs posts (and videos) and did not even come close to covering the full breadth of Windows 8. There’s much left to learn about and discover in the product.

Some of the most exciting innovations with Windows 8 are yet to come—the innovations from developers building apps on the new WinRT platform. Today, the Store is [open for business](#) and we’ll rapidly expand to over 200 markets around the world. The opportunity for developers around the world to deliver innovative (and profitable) apps is unique with Windows 8. We’re excited to see the work developers will be bringing to Windows 8. We’ll also have a chance to talk more about the Windows 8 platform at the next [BUILD conference](#) recently announced.

We know there are lots of questions about how to get Windows 8 and when, and of course more questions to come about exploring and using the full set of thousands of Windows 8 features. Our [Windows Team Blog](#) today has posted a lot of new information and gathered up some important details that we hope will answer your questions. Please check our blog and stay in touch on the in-market developments of Windows 8 there.

On behalf of the Windows 8 engineering team, we want to thank you very much for your contributions throughout development and your contributions yet to come to Windows 8. THANK YOU!

Next stop, October 26, 2012 and General Availability!

--The Windows 8 team

Collaborating to deliver Windows RT PCs

Steven Sinofsky | [2012-08-13T09:00:00+00:00](#)

*Since RTM on August 1, PC manufacturers have been using the released software to ready new PCs designed for Windows 8. Collectively, we are all very excited by the innovation and creativity that will arrive in market this October. Our engineering collaboration has been better than ever as we work to bring better performance, reliability, and battery life to new PCs designed for Windows 8. We also know many are interested in how we extended this process to a new generation of PCs built on the ARM platform. This post details how we have collaborated on the development of Windows RT and new PCs designed for the operating system. **Mike Angiulo, the vice president of our Ecosystem and Planning team, authored this post.***

–Steven

Windows 8 and Windows RT each reached the RTM milestone, and we are hard at work in collaboration with ecosystem partners, including PC manufacturers, Silicon partners, and other component suppliers, to complete high quality Windows RT and Windows 8 PCs that we think you'll love. We're very excited about the designs PC manufacturing partners have built on the foundation of Windows 8 and Windows RT.

The breadth of Windows 8 Intel- and AMD-based designs from our PC manufacturing partners will continue to push the envelope with powerful computing and innovative design. You can expect to see everything from ultra-thin sleek designs with stunning high-resolution displays, to beautifully designed All-In-One PCs with large immersive displays complete with touch, to high-power towers rocking multiple graphics cards and high-performance storage arrays. In addition, this broad range of PCs will provide price and feature combinations that allow every customer to find a PC that fits their needs and lifestyle perfectly.

We are particularly excited about the new low power x86 Windows 8 PCs that will take advantage of Intel's SoC platform innovations to provide an *always on and always connected* experience (known as connected standby). Just recently, [Lenovo announced](#) the ThinkPad Tablet 2, which offers an outstanding combination of new features built on the latest Intel ATOM® processor. We'll cover the benefits of this scenario later in the post.

Microsoft has worked very hard with this release to provide the tools and support to contribute to new PCs that are more reliable, faster, use fewer system resources, and have improved software loads than comparable Windows 7 PCs. From the newest [Ultrabook™](#) to the most powerful and extensible workstations, Windows 8 PCs are on the way.

Windows RT begins a new era of ARM-based PCs, where we are working with our Silicon and PC manufacturing partners to bring a whole new set of innovations to market. In an earlier post, [Building Windows for the ARM processor architecture](#), we focused on the detailed engineering work required to create Windows RT. In the remainder of this blog I would like to provide an update on our efforts to collaborate across the ecosystem in bringing new Windows RT PCs to market. But first, let's briefly recap the key points from the previous post:

- Windows RT shares significant code with Windows 8 and has been developed for and will be sold and supported as a part of the largest computing ecosystem in the world.
- We have achieved our goal of one Windows binary for all Windows RT SoC platforms from NVIDIA, Qualcomm, and Texas Instruments, each of which has developed innovative ARM CPUs that form the basis of a complete system.
- Delivering Windows RT PCs has been about building out a new system for the first time—a completely new ecosystem of PCs providing opportunities for PC makers to bring to life a new generation of PCs with new capabilities, starting with ARM-based processors.
- Windows RT PCs are thin and light in industrial design, and have long battery life and integrated quality. These PCs have all been designed and manufactured expressly for Windows RT.
- PC makers will provide Windows RT PCs as integrated, end-to-end products that include hardware, firmware, and Windows RT software. Windows RT software will not be sold or distributed independent of a new Windows RT PC, just as you would expect from a consumer electronics device that relies on unique and integrated pairings of hardware and software. Over the useful lifetime of the PC, the provided software will be serviced and improved.

If you are following Windows RT, perhaps you have taken note of the [Asus Tablet 600 \(Windows RT\)](#) announcement or Microsoft's own [Surface RT™](#) news. Along with Asus, we are excited to share that there will be ARM-based PC designs from Dell, Lenovo, and Samsung running Windows RT.

You will need to stay tuned for more details; PC manufacturers will be unveiling their products as we approach the Windows 8 and Windows RT launch. What I can say is the spectrum of form factors and peripherals being developed to meet each unique customer's computing needs is unique in the industry.

“Dell's tablet for Windows RT is going to take advantage of the capabilities the new ecosystem offers to help customers do more at work and home. We're excited to be Microsoft's strategic partner, and look forward to sharing more soon.”

- Sam Burd, Vice President, Dell PC Product Group

The uniqueness of our approach starts with a new way of working across partners to engineer a PC—a collaboration that brings the best of all parties together to deliver end-to-end experiences that are integrated and optimized from the chipset to the experience.

It's also worth taking a moment to describe how our collaboration on these PC efforts has been different than in any other Windows release. Our engineering collaboration on these Windows RT PCs has been strong, collaborating with the PC manufacturers, Silicon partners, and Operators to focus on hardware, software and services integration. Each respective partner was committed to sharing early iterations of their products, whether it was a SoC bring-up board, early builds of Windows RT, firmware and drivers, or hundreds of pre-release PC hardware samples (such as the ones featured in earlier demonstrations and videos). Product designs were informed and revised by our collective efforts through development and testing. As a result, all of these Windows RT PCs will have consistent *fast and fluid* touch interactions, long battery life, connected standby, and are beautiful, thin, and light designs. All of these are designed to make the most of the capabilities of Windows RT.

This is a snapshot of an *actual* pre-release Windows RT PC, showing a very early engineering prototype and the evolution to its current form.



Windows helped achieve these goals by focusing on optimizing key scenarios. Taken together, these scenarios drive a new level of mobile experience and performance not possible without new technology and engineering collaboration. So let's dig into some of the specifics.

Connected standby is the scenario of having your PC be *always on and always connected* in the new connected standby state without excessively draining your battery, so that you have access to your important and up-to-date information whenever you need it. When your Windows RT PC is not in use, it will move into a new low-power mode that allows it to keep your data fresh and current while also not requiring a battery charge for days. And when you need your system, it will turn on in less than a second at the touch of a button, which is a mobile phone experience but in a full PC. Additionally, we focused on an aggressive whole system power modeling scenario that has allowed us to better inform battery capacities to deliver all-day battery life with days of connected standby in thin and light designs.

The following chart shows some of the measurement ranges we are seeing as we test early production PCs for the connected standby and power scenarios.

The measurements are based on firmware still undergoing final optimizations, and the just released Windows RT RTM code, and will only improve as the PCs move towards manufacture. To provide context on the significance of the measurement, it is important to understand how the scenario was measured. In this case, the PC was playing back in full screen a local HD video at full resolution with a screen brightness of 200 nits. It was also configured for one email account using the Microsoft network. Finally, these numbers are also influenced by the different PC form factors themselves, which include both tablets and laptops, screen sizes that vary from 10.1" to 11.6", and battery sizes spanning 25 Whr to 42 Whr.

Scenario	Early production range
HD Video Playback	8 hours to 13 hours of scenario run time
Connected Standby	320 hours to 409 hours of scenario run time

During development, further power modeling analysis at the component level allowed us to better understand where we needed to invest in design optimizations. For example, typical touch controller solutions were based on multi-chip solutions. By reducing those solutions to single-chip designs, we achieved lower power usage and reduced thermals, which translated to smaller battery sizes and thinner and lighter designs. The table

below provides an overview of the typical weight and thicknesses we have been able to achieve with our partners across the different models representing different form factors based on ARM SoCs.




System characteristics	Measurement range
Weight (g)	520g to 1200g
Length (mm)	263mm to 298mm
Width (mm)	168.5mm to 204mm
Height (mm)	8.35mm to 15.6mm

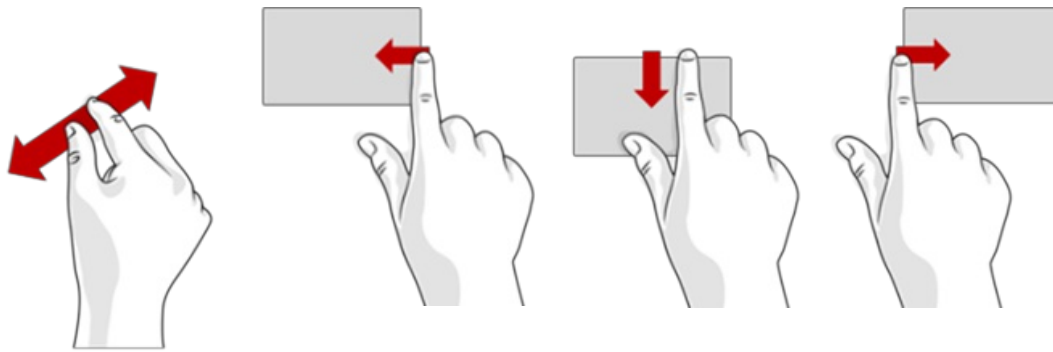
These single-chip solutions not only reduce power requirements, but they also provide performance optimizations that result in fast and fluid touch usage. The Windows RT PCs that our partners will be delivering for the upcoming launch provide sampling rates of 100Hz per finger. This not only allows for fast and fluid response, but also incredible industry-leading accuracy.

Our graphics core has also gone through extensive optimization. Besides the optimizations around power-efficient HD video playback, the core Windows RT UI animations achieve 60fps, which was our design goal.

We didn't stop at optimizing power, thickness, and weight. We also focused on enabling exciting new scenarios in these Windows RT PCs, such as sharing information intuitively and easily. You will see NFC integration in some of our Windows RT launch PCs that open up fun and interesting things like tap to share. By simply tapping two NFC-enabled Windows RT PCs together, users can easily share photos, URLs, map directions, and anything else that our software partners have designed into their Windows apps. And of course Windows RT natively supports a broad range of device scenarios such as USB mass storage, printing, audio/video peripherals, and more, along with connectivity through WWAN, Wi-Fi, Bluetooth, and USB. These build on the common foundation of Windows 8 and Windows RT, and were previously shown as early as the //BUILD/ conference.

Windows RT is not just for tablet form factors. Some of our Windows RT PCs come with full keyboard and touchpad solutions, whether removable/dockable or a traditional clamshell. Not only do these solutions provide additional battery capacity, but they also provide a new touchpad experience that incorporates intuitive Windows 8 gestures. By working closely with our touchpad component vendors, we have incorporated native support in firmware to deliver incredibly fast gesture recognition that makes interacting with Windows a breeze. The touch gestures that will be natively supported are described in these two tables:

Single-finger slide	Single/two-finger tap, double tap	Two-finger slide	
			
Mouse cursor manipulation	Primary/secondary button click, double-click at cursor location	Horizontal or vertical scroll (mouse wheel)	
Two-finger pinch	Swipe in from the right edge	Swipe down from the top edge	Swipe in from the left edge



Zoom
(Ctrl + mouse wheel)

Toggle the charms
(Windows logo key + C)

Toggle the app
commands (Windows
logo key + Z)

Switch to last app
(Windows logo key +
Ctrl + Backspace)

Finally, the wealth of hardware components and optimizations is only as rich as the applications that take advantage of them. As an ecosystem, Windows, the PC manufacturers, and the Silicon partners have been engaged with developers around the world to design application experiences that will light up the capabilities of this new PC hardware. We've purposely built thousands of reference design hardware systems to develop and test the OS and apps, collectively seeding over 1500 Windows RT reference systems to ISV and IHV companies in preparation for launch. The results are starting to show, as we've seen over 90% of the RTM applications in the Windows Store support Windows RT, as well as the Windows Hardware Certification requirements working to ensure every Windows RT PC is indeed compatible with a broad set of peripheral devices such as printers, webcams, and mobile broadband modules.

Windows RT represents a significant re-imagining of not only Windows, but Windows PCs and how we partner together to engineer them. The deep engineering collaborations from the Silicon and component manufacturers through to our PC partners and Windows engineering team have provided a compelling suite of exciting new Windows PCs that deliver on the promise of fast and fluid, always on and always connected, thin and light, and all-day battery life. We are looking forward to the exciting announcements ahead from Dell, Lenovo, and Samsung.

--Mike

Updating our built-in apps for Windows 8

Steven Sinofsky | [2012-10-04T15:30:00+00:00](#)

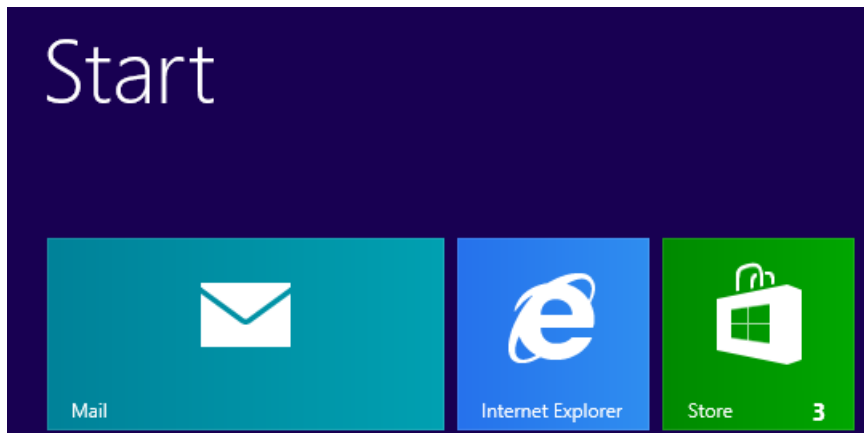
We're super excited to be approaching general availability (GA) of Windows 8 and Windows RT. With thousands of new apps in the Store, there are a lot to choose from and tens of thousands of developers have been very busy around the world creating new apps. Across Microsoft we've been busy since August adding new features and improving the apps that come with Windows and will be updating these apps before GA. We'll introduce new features, improve performance, and increase reliability. This post is authored by Gabriel Aul on our program management team and details some of the updates you will see starting in the next day or so as the updates enter the Store.

--Steven

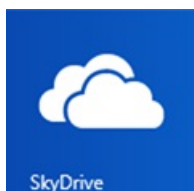
As we get close to the general availability (GA) of Windows 8, there are many things that the Windows team and other teams at Microsoft have been doing to get ready. Of course, the most important thing has been working with PC makers to help them ready the wave of amazing Windows 8 PCs that will soon be available. Some of these have been announced already and more announcements are to come. With Windows 8, we also introduced a new Store for Windows 8 apps, as well as a number of new apps that are included with Windows. We already have thousands of apps in the Windows Store, even before GA, and we're working with developers from around the world to bring more in every day. The Windows Store represents an unprecedented opportunity for developers to reach hundreds of millions of customers, and we're very pleased to see the exciting things that are showing up every day.

Of course, we are also taking advantage of the integrated way that we can deliver updates to apps through the Windows Store. Leading up to GA for Windows 8, we will be releasing updates for many of the apps that were included with the release to manufacturing (RTM) build of Windows 8 that was delivered to PC makers and to MSDN and TechNet subscribers in August. Naturally, these app updates will also be available to PC makers to include by default with their PCs shipping in the future, but for those of you who have already installed Windows 8 RTM, it is super easy to get the updates from the Store app. The Store tile will notify you when updates are available, and you can open it and click the updates link in the top right corner to see the list and install the ones you want.

The Bing app will be the first one out, available tomorrow, and more updates will roll out up until Oct 26th. You will be notified of Windows Store updates just as you have come to expect, with a count of available updates on the Store tile. You can easily choose to install the updates at a convenient time:

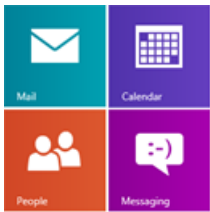


Across the board, you'll see performance and reliability improvements in the apps, but there are some great new capabilities as well. Here are some highlights of the changes you'll see:



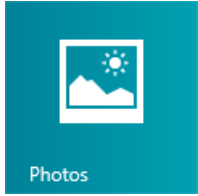
SkyDrive

- Search within SkyDrive
- Rename and move folders and files
- New first-run experience
- Custom sort order



Mail, Calendar, People, and Messaging

- Conversation view of your inbox
- Complete IMAP account support
- Accepting and declining invitations in email
- Capturing and updating your account picture
- Improved search
- Search for a contact within the Messaging app



Photos

- Crop and rotate photos
- New auto-curated collage slideshows
- View photos and videos on network locations in your Pictures Library such as Windows Home Server, network shares, and HomeGroups
- Move through photos in your Pictures Library even when you open them from the desktop



Maps

- Bird's eye view
- 3,000+ indoor venue maps
- Driving directions hints
- Improved navigation and layout
- Improved customization, including custom pushpins and roaming options
- Integration with Bing and Travel apps



Bing

- Richer search results for local content and images
- Bing rewards integration
- Use zoom on your search results to see related queries
- Use the file picker to select an image from Bing to use on your lock screen or in your other apps



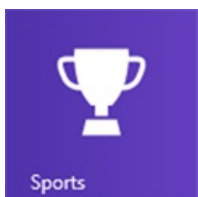
Finance

- Additional news and magazine content
- More market exchanges enabled
- Finance videos



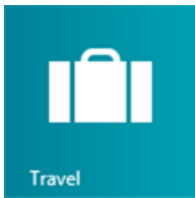
News

- Additional news content from partners such as The New York Times and The Wall Street Journal
- Enhanced article reader, including font customization, zoom, pagination, and more
- Improved offline reading experience
- News videos
- Slideshows



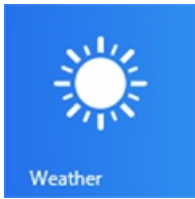
Sports

- Additional news and magazine content
- Sports videos
- Slideshows
- New soccer leagues, including MLS, J. League, and Brazilian League



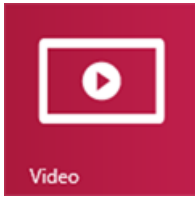
Travel

- Additional news and magazine content
- Improved flight search and new flight progress indicator
- Improved hotel listing page
- Interactive 360-degree panoramas (gyroscope supported)



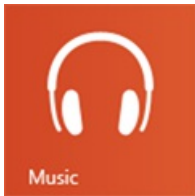
Weather

- Improved default location usability
- Hourly weather forecasts, up to 10 days
- Day & evening high/low temperatures



Video

- Purchasing in local currencies
- Closed captioning
- Search by actor or director



Music

- Expanded music services
- Rich "Now Playing" experience
- Discover more music with SmartDJ



Games

- Exciting new games
- In-game purchasing
- Invites and turn notifications

We are excited to get these updated apps out to all of the people who have been using Windows 8 so far, and to have them ready for the Windows 8 launch for people buying new PCs or upgrading their existing PCs.

-- Gabriel Aul

Updating Windows 8 for General Availability

Steven Sinofsky | [2012-10-09T09:30:00+00:00](#)

We are pleased to be releasing a set of improvements to Windows 8 in broad areas of performance, power management and battery efficiency, media playback, and compatibility. These improvements are available starting today via Windows Update. We wanted to briefly talk about our improvements to the engineering system and in particular the speed at which we were able to deliver these updates to you.

With every release of Windows we have had approximately 8-12 weeks from when we released the code to OEMs and manufacturing and when the product was available on new PCs and for retail customers. This time has historically been used to match newly developed PCs, which can include a variety of new or enhanced components, drivers, and companion software, with the final code for Windows. Because these hardware and software components are brand new, it could be the case that they uncover the need for changes and improvements to Windows in the areas of *fundamentals*.

We would often create dozens of changes for each OEM for these new PCs. Those changes would be deployed during manufacturing of those PCs and thus would be invisible to customers. While those changes could potentially apply to a broader range of PCs, we did not have in place the testing and certification to broadly distribute these updates. As a result, customers would have to wait until the first service pack to see these enhancements. We know many folks would spend time working to uncover these OEM enhancements in a desire to have the most up to date Windows.

During the final months of Windows 8 we challenged ourselves to create the tools and processes to be able to deliver these “post-RTM” updates sooner than a service pack. By developing better test automation and test coverage tools we are happy to say that Windows 8 will be totally up to date for all customers starting at General Availability. If you are an MSDN or enterprise customer, these updates will be available for your Windows 8 PCs via Windows Update as of today (October 9), following our standard cadence for Windows Updates on the second Tuesday of each month at about 10:00am.

As we have always done, any updates will have a *knowledge base* (KB) article and documentation. Documentation for these updates are documented [here](#), and the text is reproduced below. We will of course continue to issue and publish changes and enhancements from this point forward, just as we have done with Windows 7.

We think this new pace of delivering high quality updates to Windows will be a welcome enhancement for all of our customers.

--Steven

KB article title:

“**Windows 8 and Windows Server 2012 General Availability Cumulative Update**”

Description:

Windows 8 Client and Windows Server 2012 General Availability Cumulative Update is available. This cumulative update package provides a collection of performance and reliability improvements that are designed to improve the Windows 8 experience. We recommend that you apply this cumulative update as part of your regular maintenance routines.

Improvements:

- Increased power efficiency to extend battery life
- Performance improvements in Windows 8 applications and Start screen
- Improved audio and video playback in many scenarios
- Improved application and driver compatibility with Windows 8

Known issues:

- When you turn a Windows feature on or off, the computer may require a restart. For example, this action may be necessary when you turn Remote Access on or off.