

# The Engineering Windows 7 Blog or “e7”

This blog was a product of the Windows 7 development team from August 2008 through February 2010. It was originally hosted on [blogs.msdn.com](http://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](http://hardcoresoftware.learningbyshipping.com) and also contributed to the Internet Archive project.

Please enjoy,  
Steven Sinofsky  
August 2022



# *Table of Contents* in chronological order

08-08-14 Guidelines on Comments  
08-08-14 Welcome to Engineering Windows 7  
08-08-17 The Windows 7 Team  
08-08-20 Measuring the scale of a release  
08-08-27 Windows 7 — Approach to System Performance  
08-08-29 Boot Performance  
08-09-02 Product Planning for Windows...where does your feedback really go  
08-09-05 Organizing the Windows 7 Project  
08-09-06 Reflecting on a few recent threads...  
08-09-13 Follow-up on High DPI resolution  
08-09-16 More Follow up to discussion about High DPI  
08-09-18 The Ecosystem  
08-09-23 User Interface Starting, Launching, and Switching  
08-09-29 Follow-up Starting, Launching, and Switching  
08-10-01 User Interface Managing Windows windows  
08-10-04 Follow-up Managing Windows windows  
08-10-08 User Account Control  
08-10-13 Windows Desktop Search  
08-10-15 Engineering 7 A view from the bottom  
08-10-18 From Idea to Feature A view from Design  
08-10-23 Follow-up Windows Desktop Search  
08-11-01 Back from the PDC...next up, WinHEC  
08-11-11 Action Center  
08-11-19 Disk Space  
08-11-20 The Windows 7 Taskbar  
08-11-30 Accessibility in Windows 7  
08-12-15 Continuing our discussion on performance  
08-12-30 At Home with HomeGroup in Windows 7

09-01-06 Windows 7 Energy Efficiency

09-01-10 Primer on Device Support and Testing for Windows 7

09-01-15 User Account Control (UAC) – quick update

09-01-19 Engineering the Windows 7 “Windows Experience Index”

09-01-21 Follow-up Accessibility in Windows 7

09-01-25 Disk Defragmentation – Background and Engineering the Windows 7 Improvements

09-01-28 Showcasing Windows 7 Platform with Applets

09-01-30 Our Next Engineering Milestone1

09-02-05 UAC Feedback and Follow-Up

09-02-05 Update on UAC

09-02-09 Recognizing Improvements in Windows 7 Handwriting

09-02-13 Advances in typography and text rendering in Windows 7

09-02-18 Engineering the Windows 7 Boot Animation

09-02-25 Feedback and Engineering Windows 7

09-02-26 Some Changes Since Beta for the RC

09-03-03 Application Compatibility Testing — Overview

09-03-06 Beta to RC Changes – Turning Windows Features On or Off

09-03-09 Application Compatibility Testing — International

09-03-13 A few more changes from Beta to RC...

09-03-17 Designing Aero Snap

09-03-23 Federating Windows Search with Enterprise Data Sources

09-03-25 Touching Windows 7

09-04-07 Delivering a quality upgrade experience

09-04-23 Ink Input and Tablet PC

09-04-25 Engineering Windows 7 Graphics Performance

09-04-27 Improvements to AutoPlay

09-05-02 A Little Bit of Personality

09-05-05 Support and Q&A for Solid-State Drives

09-05-11 Our Next Engineering Milestone2

09-05-12 Media Streaming with Windows 7

09-05-26 Safeguarding Windows 7 – Parental Controls  
09-06-03 Creating, Saving, Sharing Themes in Windows 7  
09-06-17 Improving Audio Glitch Resilience in Windows 7  
09-06-23 Engineering Changes to ClearType in Windows 7  
09-07-07 Engineering Windows 7 for a Global Market  
09-07-22 Our Next Engineering Milestone RTM  
09-08-10 The Windows Feedback Program  
09-08-10 What we do with a bug report  
10-02-08 Windows 7 Battery Notification Messages

# Guidelines on Comments

Steven Sinofsky | [2008-08-14T11:59:00+00:00](#)

---

As the community participates in the E7 blog, we want to offer some guidelines on how we are going to handle comments in general. Our primary goal is for this to be a place for open discussion about Windows Engineering, so we don't want to have lots of overhead and process.

We love comments. We know everyone on the Windows team will be watching for comments and is looking forward to the dialog. We will work to make sure that Microsoft employees represent themselves as such, especially if they work on Windows.

Things we want to see in comments:

- Lots of good interesting responses on Windows and the posts on E7Blog
- Keep it on topic
- Keep it respectful
- Keep it fun

Things that will get comments edited/deleted:

- Offensive or abusive language or behavior
- 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 "handle" isn't offensive, abusive, or misrepresentative
- Blog-spam of any kind

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

Steven and Jon

# Welcome to Engineering Windows 7

Steven Sinofsky | [2008-08-14T14:20:00+00:00](#)

---

Welcome to our first post on a new blog from Microsoft—the Engineering Windows 7 blog, or E7 for short. E7 is hosted by the two senior engineering managers for the Windows 7 product, [Jon DeVaan](#) and [Steven Sinofsky](#). Jon and Steven, along with members of the engineering team will post, comment, and participate in this blog.

Beginning with this post together we are going to start looking forward towards the “Windows 7” project. We know there are tons of questions about the specifics of the project and strong desire to know what’s in store for the next major release of Windows. Believe us, we are just as excited to start talking about the release. Over the past 18 months since Windows Vista’s broad availability, the team has been hard at work creating the next Windows product.

The audience of enthusiasts, bloggers, and those that are the most passionate about Windows represent the folks we are dedicating this blog to. With this blog we’re opening up a two-way discussion about *how* we are making Windows 7. Windows has all the challenges of every large scale software project—picking features, designing them, developing them, and delivering them with high quality. Windows has an added challenge of doing so for an extraordinarily diverse set of customers. As a team and as individuals on the team we continue to be humbled by this responsibility.

We strongly believe that success for Windows 7 includes an open and honest, and two-way, discussion about how we balance all of these interests and deliver software on the scale of Windows. We promise and will deliver such a dialog with this blog.

Planning a product like Windows involves systematic learning from customers of all types. In terms of planning the release we’ve been working with a wide variety of customers and partners (PC makers, hardware developers, enterprise customers, developers, and more) since the start of the project. We also continue our broad consumer learning through telemetry (Customer Experience Improvement Program), usability studies, and more. One area this blog will soon explore is all the different ways we learn from customers and the marketplace that inform the release.

We have two significant events for developers and the overall ecosystem around Windows this fall. The Professional Developers Conference ([PDC](#)) on October 27 and the Windows Hardware Engineering Conference ([WinHEC](#)) the following week both represent the first venues where we will provide in-depth technical information about Windows 7. This blog will provide context over the next 2+ months with regular posts about the behind the scenes development of the release and continue through the release of the product.

In leading up to this blog we have seen a lot of discussion in blogs about what Microsoft might be trying to accomplish by maintaining a little bit more control over the communication around Windows 7 (some might say that this is a significant understatement). We, as a team, definitely learned some lessons about “disclosure” and how we can all too easily get ahead of ourselves in talking about features before our understanding of them is solid. Our intent with Windows 7 and the pre-release communication is to make sure that we have a reasonable degree of confidence in what we talk about when we do talk. Again, top of mind for us is the responsibility we feel to make sure we are not stressing priorities, churning resource allocations, or causing strategic confusion among the tens of thousands of partners and customers who care deeply and have much invested in the evolution of Windows.

Related to disclosure is the idea of how we make sure not to set expectations around the release that end up disappointing you—features that don’t make it, claims that don’t stick, or support we don’t provide. Starting from the first days of developing Windows 7, we have committed as a team to “promise and deliver”. That’s our goal—share with you what we’re going to get done, why we’re doing it, and deliver it with high quality and on time.

We’re excited about this blog. As active bloggers on Microsoft’s intranet we are both looking forward to turning our attention and blogging energies towards the community outside Microsoft. We know the ins and outs of blogging and expect to have fun, provide great information, and also make a few mistakes. We know we’ll misspeak or what we say will be heard differently than we intended. We’re not worried. All we ask is that we have a dialog based on mutual respect and the shared goal of making a great release of Windows 7.

Our intent is to post “regularly”. We’ll watch the comments and we will definitely participate both in comments and potentially in follow-up posts as required. We will make sure that members of the Windows 7 development team represent themselves as such as well. While we want to keep the dialog out in the open, please feel free to use email to [steven.sinofsky@microsoft.com](mailto:steven.sinofsky@microsoft.com) should you wish to. In particular, email is a good way to suggest topics we might have a chance to discuss on the blog.

With that, we conclude our welcome post and ask you to stay tuned and join us in this dialog about the engineering of Windows 7.

Steven and Jon

Please note the availability of this blog in several other languages via the links on the nav pane. These posts are also created by members of our development team and we welcome dialog on these sites as well. We will continue to expand the list in other languages based on feedback.

# The Windows 7 Team

Steven Sinofsky | [2008-08-17T19:00:00+00:00](#)

---

Thanks to everyone who provided comments and sent me mail. I definitely appreciate the discussion we have kicked off. There's also a ton of energy in our hallways as this blog started. It seems like a good thing to do to start off is sort of an introduction to the Windows development team. This post provides an overview of the team that is represented by this blog.

Before diving into the main topic, let's talk a bit more about what to expect from this blog. First a few words on the comments and emails I've received. I've received a ton—most of the weekend was spent reading emails and comments. There are definitely some themes. I would say by and large the reception has been very warm and we definitely appreciate that. The most frequent request was to discuss Windows performance and/or just “make Windows faster”. There's a lot to this topic so we expect to talk about this quite a bit over the next months. There are many specific requests—often representing all possible sides of an issue such as some folks saying “please get rid of (or don't do) <x>” and then other folks saying “whatever you do it is really important to keep (or do) <x>”. A big part of this blog for me personally is having the discussion about the multiple facets of any given issue. Even something that sounds as binary as performance proves to have many subtle elements. For example, some folks suggested that the best thing for boot performance is to not start anything until idle time and others suggested that the delay loading feels like it slows them down and still others have suggested that the best approach is to provide a startup manager that pushes everyone to choose what to start up. All of these have merit worth discussing and also demonstrate the subtlety and complexity of even the most straight forward request.

Second, much to the surprise of both Jon and I a number of folks questioned the “authenticity” of the post. A few even suggested that the posts are being “ghost written” or that this blog is some sort of ploy. I am typing this directly in Windows Live Writer and hitting publish. This blog is the real deal—typos, mistakes, and all. There's no intermediary or vetting of the posts. We have folks on the team who will be contributing, but we're not having any posts written by anyone other than who signs it. We will use one user name for all the posts since that keeps the blog security and ownership clear, but posts will be signed by the person that hit publish. (If I participate in the comments I will use my msdn name, [steven\\_sinofsky](#).)

And third, what frequency should folks expect and when do we get to the “features of Windows 7”. When we wrote that we would post “regularly” we meant that we don't have a schedule or calendar of posts and we don't want to commit to an artificial frequency which generally seems inconsistent with blogging. We do expect to follow a pattern similar to what you have become familiar with on the IEBlog. FWIW, on my internal blog no one has yet accused me of not contributing enough. ??

As we said in the introductory post we think it will be good to talk about the engineering of Windows 7 (the “how”) and the first step is establishing who the engineers are that do the engineering before we dive into the product itself (the “why” and “what”).

So let's meet the team..

It is pretty easy to think of the Windows team as one group or one entity, and then occasionally one specific person comes to represent the team—perhaps she gave a talk at a conference, wrote a book or article folks become familiar with, or maybe he has a blog. Within Microsoft, the Windows product is really a product of the whole company with people across all the development groups contributing in some form or another. The Windows engineering team “proper” is jointly managed by Jon and me. Jon manages the core operating system, which is, among many things, the kernel, device infrastructure, networking, and the engineering tools and system (all of which are both client and server). I am part of the Windows client experience team which develops, among many things, the shell and desktop, graphics, and media support. One other significant part of the Windows product is the Windows Media Center which is a key contribution managed along with all of Microsoft's TV support (IPTV, extenders, etc.).

There's a lot to building an org structure for a large team, but the most important part is planning the work of the team. This planning is integral to realizing our goal of improving the overall consistency and “togetherness” for Windows 7. So rather than think of one big org, or two teams, we say that the Windows 7 engineering team is made up of about 25 different *feature teams*.

A feature team represents those that own a specific part of Windows 7—the code, features, quality, and overall development. The feature teams represent the locus of work and coordination across the team. This also provides a much more manageable size—feature teams fit in meeting



spaces, can go to movies, and so on. On average a feature team is about 40 developers, but there are a variety of team sizes. There are two parts to a feature team: what the team works on and who makes up a team.

Windows 7's feature teams sound a lot like parts of Windows with which you are familiar. Because of the platform elements of Windows we have many teams that have remained fairly constant over several releases, whereas some teams are brand new or represent relatively new areas composed of some new code and the code that formed the basis of the team. Some teams do lots of work for Server (such as the VM work) and some might have big deliverables outside of Windows 7 (such as Internet Explorer).

In general a feature team encompasses ownership of combination of architectural components and scenarios across Windows. "Feature" is always a tricky word since some folks think of feature as one element in the user-interface and others think of the feature as a traditional architectural component (say TCP/IP). Our approach is to balance across scenarios and architecture such that we have the right level of end-to-end coverage and the right parts of the architecture. One thing we do try to avoid is separating the "plumbing" from the "user interface" so that teams do have end-to-end ownership of work (as an example of that, "Find and Organize" builds both the indexer and the user interface for search). Some of the main feature teams for Windows 7 include (alphabetically):

- Applets and Gadgets
- Assistance and Support Technologies
- Core User Experience
- Customer Engineering and Telemetry
- Deployment and Component Platform
- Desktop Graphics
- Devices and Media
- Devices and Storage
- Documents and Printing
- Engineering System and Tools
- File System
- Find and Organize
- Fundamentals
- Internet Explorer (including IE 8 down-level)
- International
- Kernel & VM
- Media Center
- Networking - Core
- Networking - Enterprise
- Networking - Wireless
- Security
- User Interface Platform

- Windows App Platform

I think most of these names are intuitive enough for the purposes of this post—as we post more the members of the team will identify which feature team they are on. This gives you an idea of the subsystems of Windows and how we break down a significant project into meaningful teams. Of course throughout the project we are coordinating and building features across teams. This is a matter of practice because you often want to engineer the code in one set of layers for efficiency and performance (say bottom up), but end-users might experience it across layers, and IT pros might want to manage a desktop from the top-down. I admit sometimes this is a little bit too much of an insider view as you can't see where some interesting things “live”. For example, the tablet and inking functionality is in our User Interface Platform team along with speech recognition, multi-touch and accessibility. The reason for this is the architectural need to share the infrastructure for all mechanisms of “input” even if any one person might not cross all those layers. This way when we design the APIs for managing input, developers will see the benefits of all the modes of user interaction through one set of APIs.

The other aspect of our feature teams is the exact composition. A feature team represents three core engineering disciplines of [software development engineers](#) (sde or dev), software development engineers in test ([sdet](#) or test, sorry but I haven't written a job description externally), and [program managers](#) (pm). Having all three of these engineering disciplines is a unique aspect of Microsoft that has even caught the attention of some [researchers](#). In my old blog I described dev and pm which I linked to above (I still owe a similar post on SDET!).

The shortest version of these roles is dev is responsible for the architecture and code, pm is responsible for the feature set and specification, and test is responsible for validation and the ultimate advocate for the customer experience. Everyone is responsible for quality and performance, each bringing their perspective to the work. For any given feature, each of dev, test, and pm work as a team of peers (both literally and conceptually). This is a key “balance of power” in terms of how we work and makes sure that we take a balanced approach to developing Windows 7. Organizationally, we are structured such that devs work for devs, sdets work for sdets, and pm works for pm. That is we are organized by these “engineering functions”. This allows for two optimizations—the focus on expertise in both domain and discipline and also the ability to make sure we are not building the product in silos, but focused on the product as a whole.

We talk about these three disciplines together because we create feature teams with  $n$  developers,  $n$  testers, and  $1/2n$  program managers. This ratio is pretty constant across the team. On average a feature team is about 40 developers across the Windows 7 project.

We also have core members of our engineering team that work across the entire product:

- **Content Development** – the writers and editors that create the online assistance, web site, SDK documents, and deployment documents.
- **Product Planning** – responsible for the customer research and learning that informs the selection of features. Product Planning also coordinates the work we do with partners across the ecosystem in terms of partnering through the design and development of the release.
- **Product Design** – develops the overall interaction model, graphical language, and design language for Windows 7
- **Research and Usability** – creates field and lab studies that show how existing products and proposed feature perform with customers

Some have said that the Windows team is just too big and that it has reached a size that causes engineering problems. At the same time, I might point out that just looking at the comments there is a pretty significant demand for a broad set of features and changes to Windows. It takes a set of people to build Windows and it is a big project. The way that I look at this is that our job is to have the Windows team be the right size—that sounds cliché but I mean by that is that the team is neither too large nor too small, but is effectively managed so that the work of the team reflects the size of the team and you see the project as having the benefits we articulate. I'm reminded of a scene from *Amadeus* where the Emperor suggests that the *Marriage of Figaro* contains “too many notes” to which Mozart proclaims “there are just as many notes, Majesty, as are required, neither more nor less.” Upon the Emperor suggesting that Mozart remove a few notes, Mozart simply asks “which few did you have in mind?” Of course the people on the team represent the way we get feature requests implemented and develop end to end scenarios, so the challenge is to have the right team and the right structure to maximize the ability to get those done—neither too many nor too few.

I promised myself no post would be longer than 4 pages and I am getting close. The comments are great and are helping us to shape future posts. I hope this post starts to develop some additional shared context.

--Steven

# Measuring the scale of a release

Steven Sinofsky | [2008-08-20T03:00:00+00:00](#)

---

Thanks for all the feedback that we have been getting. That much of it is positive is certainly appreciated. I've been answering mails as best I can and along with members of the team we've been having the discussion in the comments. Everyone has done a great job sharing their views on specifics, wishes, and requests. I love getting these mails and reading the comments. It is fantastic. I just want to make sure folks know I can't answer each one! What we are going to do is look to the emails and comments as a way of suggesting posts we should write. The team overall appreciate the warm reception from all those that have joined us--we know we have lots of energetic discussions ahead of us and we're genuinely happy to start.

With this post, I am hoping to continue the dialog on the way we think "inside the Win7 team" so to speak—in a sense this is about expanding the team a bit and bringing you into some more of the discussions we have about planning a release. This conversation about major or minor releases is very much like the one I have with my boss as we start planning ??

When we started planning the release, the first thing some might think we have to decide is if Windows 7 (client) would be a "major release" or not. I put that in quotes because it turns out this isn't really something you decide nor is it something with a single answer. The magnitude of a release is as much about your perspective on the features as it is about the features themselves. One could even ask if being declared a major release is a compliment or not. As engineers planning a product we decide up front the percentage of our development team will that work on the release and the extent of our schedule—with the result in hand customers each decide for themselves if the release is "major", though of course we like to have an opinion. On the [server blog](#) we talked about the schedule and we shared our opinion of the scale of the releases of Windows 7 client and server.

Our goal is about building an awesome release of Windows 7.

Across all customers, there is always a view that a major release is one that has features that are really the ones for **me**. A minor release is one that doesn't have anything for **me**. It should then be pretty easy to plan a major release—just make sure it exactly the right features for everyone (and given the focus on performance, it can't have any extra features, even if other people want them)! As engineers we all know such a design process is really impossible, especially because more often than not any two customers can be found to want exactly opposite features. In fact as I type this I received sequential emails one saying "[N]obody cares about touch screen nonsense" and the other saying "[Win7 needs] more advanced/robust 'touch' features". When you just get unstructured and unsolicited input you see these opposites quite a bit. I'm sure folks are noticing this on the blog comments as well.

Let's explore the spectrum of release magnitude across a couple of (but not all) different types of customers: end-users, developers, partners, IT professionals, and *influentials*.

**End-users** are generally the most straight-forward in terms of deciding how big a release is going to be. For an end-user a release is a big deal if they want to go out and buy an upgrade or buy a new PC. We could call that a major release. Seems simple enough and a major release is good for everyone. On the other hand, one could also imagine that a release is really cool and people want to buy it, but they also want to use their existing PC and the release requires more memory, updated drivers that might not be available, or maybe some specific hardware to be fully realized. Then it seems that a major release goes from a positive to a bit of an under-taking and thus loses some of its luster. Of course we all know that what folks really want is all the things they want that runs on the hardware they want—then that is a great product to get (whether it is major or not).

**Developers** look at a release through a different lens. Obviously for developers a release is a major one if there are new APIs and capabilities to take advantage of in their software—again straight-forward enough. It could also be the case that a previous release had a lot of new APIs and folks are just getting familiar with using them and so what they really want is to round out the APIs and maybe improve performance. So one might suspect that the first release is a major release and the second type is a minor release. But if you look at the history of software products, it is often these "minor" releases that themselves become the major releases – Windows 3.1, Office 4.2, or even Windows XP SP2. In each of these cases, the target for developers became the "minor" release but in the eyes of the market that was the "major" release. The reason developers want to use new APIs is to differentiate their products or focus their energies on domain expertise they bring to the table, not just call new APIs for the sake of calling them. In that sense, a release might be a major one if it just happens to free up enough time for an ISV that they bet on the new APIs because they can focus on some things that are a major deal to them.

**Partners** represent the broad set of folks who create PCs, hardware, and the infrastructure we think of as the ecosystem that Windows is part of. Partners tend to think about a major release in terms of the opportunity it creates and thus a major release might be one with a lot of change and thus it affords the opportunity to provide new hardware and infrastructure to customers. On the other hand, incompatibilities with the past might be viewed in a less than positive light if it means a partner needs to stop moving forward and revisit past work to bring it up to the required compatibility with a new release of Windows. If they choose, for any number of reasons, not to do that work then the release might be viewed as a minor one because of the lack of ecosystem support. So again we see that a big change can be viewed through the lens of a major or a minor release.

**IT professionals** are often characterized as conservative by nature and thus take a conservative view of change. Due to the business focused nature of the role, the evaluation of any software product is going to take place in the context of a return on investment. So for an IT professional a major release would be one that delivers significant business value. This business value could be defined as a major investment in deployment and management of the software for example. Yet for end-users or developers, these very same features might not even be interesting let alone worthy of being a major or minor release.

**Influentials** are all the folks who are in the business of providing advice, analysis, and viewpoints on the software we make. These folks often look at releases through the metric of “change”. Big changes equal major release. A big change can be a “re-architecture” as we saw in the transition from Windows 9x to Windows 2000—even though these products looked the same there was tons of change to talk about under the hood. So for reviewers and analysts it was definitely a major release. Big changes can also be big changes in the user-interface because that drives lots of discussion and it is easy to show all the change. Yet for each of these, this definition of major can also be viewed as a less than positive attribute. Re-architecture means potential incompatibilities. New user-interface can mean learning and moving from the familiar.

We’ve seen a lot of comments and I have gotten a lot of email talking about re-architecting Windows as a symbol of a major release. We’ve also gotten a lot of feedback about how a major release is one that breaks with supporting the past. If I could generalize, folks are usually implying that if we do things like that then a number of other major benefits will follow—re-architecting leads to better performance, breaking with the past leads to using less memory. It is always tricky to debate those points because we are comparing a known state to a state where we fix all the things we know to fix, but we don’t yet know what we might introduce, break, or otherwise not fix. So rather than define a major release relative to the implementation, I think it makes sense define the success of the release relative to the benefits of whatever implementation is chosen. We will definitely continue to pick up on this part of the discussion--there's a lot of dialog to have.

The key is always a balance. We can have big changes for all customers if we prepare all the necessary folks to work through the change. We can have small changes have a big impact if they are the right changes at the right time, and those will get recorded over time as a major release.

We’ve talked about the timing and the way we structure the team, so you have a sense for the “inputs” into the project. If we listened well and focused our efforts correctly, then each type of customers will find things that make the product worthwhile. And if we do our job at effectively communicating the product, then even the things that could be “problems” are seen in the broader context of an ecosystem where everyone collectively benefits when a few people benefit significantly.

From our perspective, we dedicated our full engineering team and a significant schedule to building the Windows 7 client OS. That makes it a major undertaking by any definition. We intend for Windows 7 to be an awesome release.

I hope this helped to see that perspective is everything when it comes to deciding how big a release is for each type of customer.

--Steven

# Windows 7 — Approach to System Performance

Steven Sinofsky | [2008-08-27T03:00:00+00:00](#)

---

Many folks have commented and written email about the topic of performance of Windows. The dialog has been wide ranging—folks consistently want performance to improve (of course). As with many topics we will discuss, performance, as absolute and measurable as it might seem, also has a lot of subtlety. There are many elements and many tradeoffs involved in achieving performance that meets everyone's expectations. We know that even meeting expectations, folks will want even more out of their Windows PCs (and that's expected). We've re-dedicated ourselves to work in this area in Windows 7 (and IE 8). This is a major initiative across each of our feature teams as well as the primary mission of one of our feature teams (Fundamentals). For this post, I just wanted to frame the discussion as we dig into the topic of performance in subsequent posts. Folks might find this post on [IE8 performance](#) relevant along with the [beta 2 release](#) of IE 8.

Performance is made up of many different elements. We could be talking about response time to a specific request. It might mean how much RAM is "typical" or what CPU customers need. We could be talking about the clock time to launch a program. It could mean boot or standby/resume. It could mean watching CPU activity or disk I/O activity (or lack disk activity). It could mean battery life. It might even mean something as mundane as typical disk footprint after installation. All of these are measures of performance. All of these are systematically tracked during the course of development. We track performance by running a known set of scenarios (there are thousands of these) and developers can run specific scenarios based on exercising more depth or breadth. The following represent some (this is just a partial list) of the metrics we are tracking and while developing Windows 7:

- **Memory usage** – How much memory a given scenario allocates during a run. As you know, there is a classic tradeoff in [time v. space](#) in computer science and we're not exempt. We see this tradeoff quite a bit in caches where you can use more memory (or disk space) in order to improve performance or to avoid re-computing something.
- **CPU utilization** – Clearly, modern microprocessors offer enormous processing power and with the advent of multiple cores we see the opportunity for more parallelism than ever before. Of course these resources are not free so we measure the CPU utilization across benchmark runs as well. In general, the goal should be to keep the CPU utilization low as that improves multi-user scenarios as well as reduces power consumption.
- **Disk I/O** – While hard drives have improved substantially in performance we still must do everything we can do minimize the amount that Windows itself does in terms of reading and writing to disk (including paging of course). This is an area receiving special attention for Windows 7 with the advent of solid state storage devices that have dramatically different "characteristics".
- **Boot, Shutdown, Standby/Resume** – All of these are the source of a great deal of focus for Windows 7. We recognize these can never be fast enough. For these topics the collaboration with the PC manufacturers and hardware makers plays a vital role in making sure that the times we see in a lab (or the performance you might see in a "clean install") are reflected when you buy a new PC.
- **Base system** – We do a great deal to measure and tune the base system. By this we mean the resource utilization of the base system before additional software is loaded. This system forms the "platform" that defines what all developers can count on and defines the system requirements for a reasonable experience. A common request here is to kick something out of the base system and then use it "on demand". This tradeoff is one we work on quite a bit, but we want to be careful to avoid the situation where the vast majority of customers face the "on demand" loading of something which might reduce perceived performance of common scenarios.
- **Disk footprint** – While not directly related to runtime performance, many folks see the footprint of the OS as indicative of the perceived performance. We have some specific goals around this metric and will dive into the details soon as well. We'll also take some time to explain `\Windows\WinSxS` as it is often the subject of much discussion on technet and msdn! Here rather than runtime tradeoffs we see convenience tradeoffs for things like on disk device drivers, assistance content, optional Windows components, as well as diagnostics and logging information.

We have criteria that we apply at the end of our milestones and before we go to beta and we won't ship without broadly meeting these criteria. Sometimes these criteria are micro-benchmarks (page faults, processor utilization, working set, gamer frame rates) and other times they are more scenario based and measure time to complete a task (clock time, mouse clicks). We do these measurements on a variety of hardware platforms (32-bit or 64-bit; 1, 2, 4GB of RAM; 5400 to 7200 RPM or solid-state disks; a variety of processors, etc.) Because of the inherent tradeoffs in some architectural approaches, we often introduce conditional code that depends on the type of hardware on which Windows is running.

On the one hand, performance should be straight forward—use less, do less, have less. As long as you have less of everything performance should

improve. At the extreme that is certainly the case. But as we have seen from the comments, one person's must-have is another person's must-not-have. We see this a lot with what some on have called "eye candy"—we get many requests to make the base user interface "more fun" with animations and graphics ("like those found on competing products") while at the same time some say "get rid of graphics and go back to Windows 2000". Windows is enormously flexible and provides many ways to tune the experience. We heard lots on this forum about providing specific versions of Windows customized for different audiences, while we also heard quite a bit about the need to reduce the number of versions of Windows. However, there are limits to what we can provide and at the same time provide a reliable "platform" that customers and developers can count on and is robust and manageable for a broad set of customers. But of course within a known context (within your home or within a business running a known set of software) it will always be possible to take advantage of the customization and management tools Windows has to offer to tune the experience. The ability to have choice and control what goes on in your PC is of paramount importance to us and you will see us continue to focus on these attributes with Windows 7.

By far the biggest challenge in delivering a great PC experience relative to performance is that customers keep using their PCs to do more and more things and rightfully expect to do these things on the PC they own by just adding more and more software. While it is definitely the case that Windows itself adds functionality, we work hard to pick features that we believe benefit the broadest set of customers. At the same time, a big part of Windows 7 will be to continue to support choice and control over what takes place in Windows with respect to the software that is provided, what the default handlers are for file types and protocols, and providing a platform that makes it easy for end-users to personalize their computing experience.

Finally, it is worth considering real world versus idealized settings. In order to develop Windows we run our benchmarks in a lab setting that allows us to track specifically the code we add and the impact that has. We also work closely with the PC Manufacturers and assist them in benchmarking their systems as they leave the factory. And for true real-world performance, the [Microsoft Customer Experience Improvement Program](#) provides us (anonymous, private, opt-in) data on how machines are really doing. We will refer to this data quite a bit over the next months as it forms a basis for us to talk about how things are really working, rather than using anecdotes or less reliable forms of information.

In our next post we will look at startup and boot performance, and given the interest we will certainly have more to say about the topic of performance.

--Steven

# Boot Performance

Steven Sinofsky | [2008-08-29T03:00:00+00:00](#)

*Ed. Note: This is our first post from a senior member of the development team. Allow me to introduce Michael Fortin who is one of Microsoft's Distinguished Engineers and leads the Fundamentals feature team that is in our Core Operating System group. Michael leads performance and reliability efforts across the Windows platform. --Steven (PS: Be sure to visit [www.microsoft.com/ie](http://www.microsoft.com/ie) and try out the beta 2 release of Internet Explorer 8).*

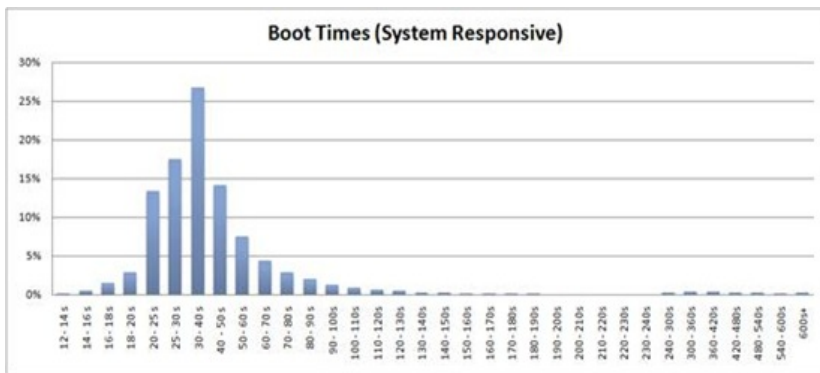
For Windows 7, we have a dedicated team focused on startup performance, but in reality the effort extends across the entire Windows division and beyond. Our many hardware and software partners are working closely with us and can rightly be considered an extension to the team.

Startup can be one of three experiences; boot, resume from sleep, or resume from hibernate. Although resume from sleep is the default, and often 2 to 5 seconds based on common hardware and standard software loads, this post is primarily about boot as that experience has been commented on frequently. For Windows 7, a top goal is to significantly increase the number of systems that experience very good boot times. In the lab, a very good system is one that boots in under 15 seconds.

For a PC to boot fast a number of tasks need to be performed efficiently and with a high degree of parallelism.

- Files must be read into memory.
- System services need to be initialized.
- Devices need to be identified and started.
- The user's credentials need to be authenticated for login.
- The desktop needs to be constructed and displayed.
- Startup applications need to be launched.

Because systems and configurations differ, boot times can vary significantly. This is verified by many lab results, but can also be seen in independent [analysis](#), such as that conducted by Ed Bott. Sample data from Ed's population of systems found that only 35% of boots took less than 30 seconds to give control to the user. Though Ed's data is from a small population, his data is nicely in line with what we're observing. Windows Vista SP1 data (below) also indicates that roughly 35% of systems boot in 30 seconds or less, 75% of systems boot in 50 seconds or less. The Vista SP1 data is real world telemetry data. It comes to us from the very large number of systems (millions) where users have chosen to send anonymous data to Microsoft via the [Customer Experience Improvement Program](#).





From our perspective, too few systems consistently boot fast enough and we have to do much better. Obviously the systems that are greater than 60 seconds have something we need to dramatically improve—whether these are devices, networking, or software issues. As you can see there are some systems experiencing very long boot times. One of the things we see in the PC space is this variability of performance—variability arises from the variety of choices, and also the variety of quality of components of any given PC experience. There are also some system maintenance tasks that can contribute to long boot times. If a user opts to install a large software update, the actual updating of the system may occur during the next boot. Our metrics will capture these and unfortunately they can take minutes to complete. Regardless of the cause, a big part of the work we need to do as members of the PC ecosystem is address long boot times.

In both Ed's sample and our telemetry data, boot time is meant to reflect when a machine is ready and responsive for the user. It includes logging in to the system and getting to a usable desktop. It is not a perfect metric, but one that does capture the vast majority of issues. On Windows 7 and Vista systems, the metric is captured automatically and stored in the system event log. Ed's article covers this in depth.

We realize there are other perceptions that users deem as reflecting boot time, such as when the disk stops, when their apps are fully responsive, or when the start menu and desktop can be used. Also, "Post Boot" time (when applications in the Startup group run and some delayed services execute), the period before Windows boot is initiated, and BIOS time can be significant. In our efforts, we've not lost sight of what users consider being representative of boot.

Before discussing some of our Windows 7 efforts, we'd like to point out there is considerable engagement with our partners underway. In scanning dozens of systems, we've found plenty of opportunity for improvement and have made changes. Illustrating that, please consider the following data taken from a real system. As the system arrived to us, the off-the-shelf configuration had a ~45 second boot time. Performing a clean install of Vista SP1 on the same system produced a consistent ~23 second boot time. Of course, being a clean install, there were many fewer processes, services and a slightly different set of drivers (mostly the versions were different). However, we were able to take the off-the-shelf configuration and optimize it to produce a consistent boot time of ~21 seconds, ~2 seconds faster than the clean install because some driver/BIOS changes could be made in the optimized configuration.

For this same system, it is worth noting the resume from sleep time is approximately 2 seconds, achieving a nearly instant on experience. We do encourage users to choose sleep as an alternative to boot.

As an example Windows 7 effort, we are working very hard on system services. We aim to dramatically reduce them in number, as well as reduce their CPU, disk and memory demands. Our perspective on this is simple; if a service is not absolutely required, it shouldn't be starting and a trigger should exist to handle rare conditions so that the service operates only then.

Of course, services exist to complete user experiences, even rare ones. Consider the case where a new keyboard, mouse or tablet HW is added to the system while it was off. If this new HW isn't detected and drivers installed to make the HW work during startup, then the user may not be able to enter their credentials and log into the machine. For a given user, this may be a very rare or never encountered event. For a population of 100s of millions of users, this can happen frequently enough to warrant having mechanisms to support it. In Windows 7, we will support this scenario and many others with fewer auto start services because more comprehensive service trigger mechanisms have been created.

As noted above, device and driver initialization can be a significant contributor as well. In Windows 7, we've focused very hard on increasing parallelism of driver initialization. This increased parallelism decreases the likelihood that a few slower devices/drivers will impact the overall boot time.

In terms of reading files from the disk, Windows 7 has improvements in the "prefetching" logic and mechanisms. Prefetching was introduced way back in Windows XP. Since today's disks have differing performance characteristics, the scheduling logic has undergone some changes to keep pace and stay efficient. As an example, we are evaluating the prefetcher on today's solid state storage devices, going so far as to question if is required at all. Ultimately, analysis and performance metrics captured on an individual system will dynamically determine the extent to which we utilize the prefetcher.

There are improved diagnostic experiences in Windows 7 as well. We aim to quickly identify specific issues on individual systems, and provide

help to assist in resolving the issues. We believe this is an appropriate way to inform users about some problems, such as having too many startup applications or the presence of lengthy domain-oriented logon scripts. As many users know, having too many startup applications is often the cause of long boot times. Few users, however, are familiar with implications of having problematic boot or logon scripts. In Windows XP, Vista and in Windows 7, the default behavior for Windows is to log the user into the desktop without waiting for potentially lengthy networking initialization or scripts to run. In corporate environments, however, it is possible for IT organizations to change the default and configure client systems to contact servers across the network. Unfortunately, when configuring clients to run scripts, domain administrators typically do so in a synchronous and blocking fashion. As a result, boot and logon can take minutes if networking timeouts or server authentication issues exist. Additionally, those scripts can run very expensive programs that consume CPU, disk and memory resources.

In addition to working on Windows 7 specific features and services, we are sharing tools, tests and data with our partners. The tools are available to enthusiasts as well. The tools we use internally to detect and correct boot issues are freely available today [here](#) as a part of the Windows Performance Toolkit. While not appropriate for most users, the tools are proving to be very helpful for some.

One of the topics we want to talk about in the future which we know has been written about a great deal and is the subject of many comments, is the role that additional software beyond the initial Windows installation plays in overall system performance. The sheer breadth and depth of software for Windows means that some software will not have the high quality one would hope, while the vast majority is quite good. Microsoft must continue to provide the tools for developers to write high performance software and the tools for end-users to identify the software on their system that might contribute to performance that isn't meeting expectations. Windows itself must also continue to improve the *defensive* tactics it uses to isolate and inform the end-user about software that *might* contribute to poor performance.

Another potential future topic pertains to configuration changes a user can make on their own system. Many recommended changes aren't helpful at all. For instance, we've found the vast majority of "registry tweak" recommendations to be bogus. Here's one of my favorites. If you perform a Live search for "Enable Superfetch on XP", you'll get a large set of results. I can assure you, on Windows XP there is no Superfetch functionality and no value in setting the registry key noted on these sites. As with that myth, there are many recommendations pertaining to CPU scheduling, memory management and other configuration changes that aren't helpful to system performance.

Startup is one topic on performance. As described in the previous post we want to continue the discussion around this topic. What are some of the elements you'd like to discuss more?

Michael Fortin

# Product Planning for Windows...where does your feedback really go?

Steven Sinofsky | [2008-09-02T20:08:00+00:00](#)

---

*Ed. Note: Allow me to introduce Mike Angiulo who leads the Windows PC Ecosystem and Planning team. Mike's team works closely with all of our hardware and software partners and leads the engineering team's product planning and research efforts for each new version of Windows.*  
--Steven

In Windows we have a wide variety of mechanisms to learn about our customers and marketplace which all play roles in helping us decide what we build. From the individual questions that our engineers will answer at [WinHEC](#) and [PDC](#) to the millions of records in our telemetry systems we have tools for answering almost every kind of question around what you want us to build in Windows and how well it's all working. Listening to all of these voices together and building a coherent plan for an entire operating system release is quite a challenge – it can feel like taking a pizza order for a billion of your closest friends!

It should come as no surprise that in order to have a learning organization we need to have an organization that is dedicated to learning. This is led by our Product Planning team, a group of a couple dozen engineers that is both organized and sits with the program managers, developers and testers in the feature teams. They work throughout the product cycle to ensure that our vision is compelling and based on a deep understanding of our customer environment and is balanced with the business realities and competitive pressures that are in constant flux. Over the last two years we've had a team of dozens of professional researchers fielding surveys, listening to focus groups, and analyzing telemetry and product usage data leading up to the vision and during the development of Windows 7 – and we're not done yet. From our independently run marketing research to reading your feedback on this blog we will continue to refine our product and the way we talk about it to customers and partners alike. That doesn't mean that every wish goes answered! One of the hardest jobs of planning is in turning all of this data into actionable plans for development. There are three tough tradeoffs that we have been making recently.

First there is what I think of as the 'taste test challenge.' Over thirty years ago this meme was introduced in a famous war between two colas. Remember New Coke? It was the result of overemphasizing the very initial response to a product versus longer term customer satisfaction. We face this kind of challenge all the time with Windows – how do we balance the need for the product to be approachable with the need for the product to perform throughout its lifecycle? Do you want something that just boots as fast as it can or something that helps you get started? Of course we can take this to either extreme and you can say we have – we went from c:\ to [Microsoft Bob](#) in only a matter of a decade. Finding the balance between a product that is fresh and clean out of the box and continues to perform over time is a continual balance. We have ethnographers who gather research that in some cases starts even before the point of purchase and continues for months with periodic visits to learn how initial impressions morph into usage patterns over the entire lifecycle of our products.

Second we're always looking out for missing the 'trees for the forest.' By this I mean finding the appropriate balance between aggregate and individual user data. A classic argument around PCs has always been that a limited subset of actions comprises a large percentage of the use case. The resulting argument is that a limited function device would be a simpler and more satisfying experience for a large percentage of customers! Of course this can be shown to be flawed in both the short term and the long term. Over the long term this 'common use case' has changed from typing & printing to consuming and burning CDs and gaming to browsing and will continue to evolve. Even in the short term we have studied the usage of thousands of machines (from users who opt-in of course) and know that while many of the common usage patterns are in fact common, that nearly every single machine we've ever studied had one or more unique applications in use that other machines didn't share! This long tail phenomena is very important because if we designed for the "general case" we'd end up satisfying nobody. This tradeoff between choice and complexity is one that benefits directly from a rigorous approach to studying usage of both the collective and individual and not losing sight of either.

Third is all about timing. Timing is everything. We have an ongoing process for learning in a very dynamic market – one that is directly influenced by what we build. The ultimate goal is to deliver the ultimate in software & hardware experiences to customers – the right products at the right time. We've seen what happens if we wait too long to release software support for a new category (we should have done a better job with an earlier Bluetooth pairing standard experience) and what also happens when we ship software that the rest of the ecosystem isn't ready for yet. This problem has the dimension of working to evangelize technologies that we know are coming, track competing standards, watch user scenarios evolve and try to coordinate our software support at the same time. To call it a moving target isn't saying enough! It does though explain why we're constantly taking feedback, even after any given version of

Windows is done.

These three explicit tradeoffs always make for lively conversation – just look at the comments on this blog to date! Of course being responsive to these *articulated needs* is a must in a market as dynamic and challenging as ours. At the same time we have to make the biggest tradeoff of them all – balancing what you’re asking for today with what we think you’ll be asking for tomorrow. That’s the challenge of defining *unarticulated needs*. All technology industries face this tradeoff whether you call it the need to innovate vs. fix or subscribe to the S curve notion of discontinuities. Why would two successful auto companies, both listening to the same market input, release the first commercial Hummer and first hybrid Prius in the same year? It wasn’t that 1998 was that confusing, it was that the short term market demands and the long term market needs weren’t obviously aligned. Both forces were visible but readily dismissed – the need for increased off road capacity to negotiate the crowded suburban mall parking lots and the impending environmental implosion being predicted on college campuses throughout the world. We face balancing acts like this all the time. How do we deliver backwards compatibility and future capability one release at a time? Will the trend towards 64 bit be driven by application scenarios or by 4GB machines selling at retail?

We have input on key tradeoffs. We have a position on future trends. That’s usually enough to get started on the next version of the product and we stay connected with customers and partners during throughout development to keep our planning consistent with our initial direction but isn’t enough to know we’re ready to ship. Really being done has always required some post engineering feedback phase whether it’s a Community Technical Preview, Technology Adoption Program or a traditional public Beta. The origin of Beta testing and even the current definition of the term aren’t clear. Some products now seem to be in Beta forever! We work to find the best possible timing for sharing the product and gathering final feedback. If we release it too early it’s usually not in any shape to evaluate, especially with respect to performance, security, compatibility and other critical fundamentals. If we release too late we can’t actually take any of the feedback you give us, and I can’t think of a worse recipe for customer satisfaction than to ask for feedback which gets systematically ignored. I was just looking at another software “feedback” site where a bunch of the comments just asked the company to “please read this site!” For Windows 7 we’re going to deliver a Beta that is good enough to experience and leaves us enough time to address areas where we need more refinement. This blog will be an important part of the process because it will provide enough explanation and content and guidance to help you understand the remaining degrees of freedom, some of the core assumptions that went into each area and will structure our dialogue so that we can listen and respond to as much feedback as you’re willing to give. Some of this will result in bugs that get fixed, some will result in bugs in drivers or applications that we help our partners fix. And of course sometimes we’ll just end up with healthy debate – but even in this case we will be talking, we will respond to constructive comments, bugs and ideas and we will both be starting that conversation with more context than ever. So please do keep your comments coming. Please participate in the [Customer Experience Improvement program](#). Give us feedback at [WinHEC](#) and [PDC](#) and in the newgroups and forums – we’re listening!

Thanks,

- Mike

# Organizing the Windows 7 Project

Steven Sinofsky | [2008-09-05T03:00:00+00:00](#)

---

Hi Jon DeVaan here.

Steven wrote about how we organize the engineering team on Windows which is a very important element of how work is done. Another important part is how we organize the engineering project itself.

I'd like to start with a couple of quick notes. First is that Steven reads and writes about ten times faster than I do, so don't be too surprised if you see about that distribution of words between the two of us here. (Be assured that between us I am the deep thinker :-). Or maybe I am just jealous.) Second is that we want to keep sharing the "how we build Windows 7" topics since that gives us a shared context for when we dive into feature discussion as we get closer to the PDC and WinHEC. We want to discuss how we are engineering Windows 7 including the lessons learned from Longhorn/Vista. All of these realities go into our decision making on Windows 7.

OK, on to the tawdry bits.

Steven linked last time to the book [Microsoft Secrets](#), which is an excellent analysis of what I like to call version two of the Microsoft Engineering System. (Version one involved index cards and "floppy net" and you really don't want to hear about it.) Version two served Microsoft very well for far longer than anyone anticipated, but learning from Windows XP, the truly different security environment that emerged at that time and from Longhorn/Vista, it became clear that it was time for another generational transformation in how we approach engineering our products.

The lessons from XP revolve around the changed security landscape in our industry. You can learn about how we put our learning into action by looking at the [Security Development Lifecycle](#), which is the set of engineering practices recommended by Microsoft to develop more secure software. We use these practices internally to engineer Windows.

The comments on this blog show that the quality of a complete system contains many different attributes, each of varying importance to different people, and that people have a wide range of opinions about Vista's overall quality. I spend a lot of time on core reliability of the OS and in studying the telemetry we collect from real users (only if they opt-in to the [Customer Experience Improvement Program](#)) I know that Vista SP1 is just as reliable as XP overall and more reliable in some important ways. The telemetry guided us on what to address in SP1. I was glad to see one way pointed out by people commenting about sleep and resume working better in Vista. I am also excited by the prospect of continuing our efforts (we are) using the telemetry to drive Vista to be the most reliable version of Windows ever. I add to the list of Vista's qualities successfully [cutting security vulnerabilities by just under half compared to XP](#). This blog is about Windows 7, but you should know that we are working on Windows 7 with a deep understanding of the performance of Windows Vista in the real world.

In the most important ways, people who have emailed and commented have highlighted opportunities for us to improve the Windows engineering system. Performance, reliability, compatibility, and failing to deliver on new technology promises are popular themes in the comments. One of the best ways we can address these is by better day-to-day management of the engineering of the Windows 7 code base—or the daily build quality. We have taken many concrete steps to improve how we manage the project so that we do much better on this dimension.

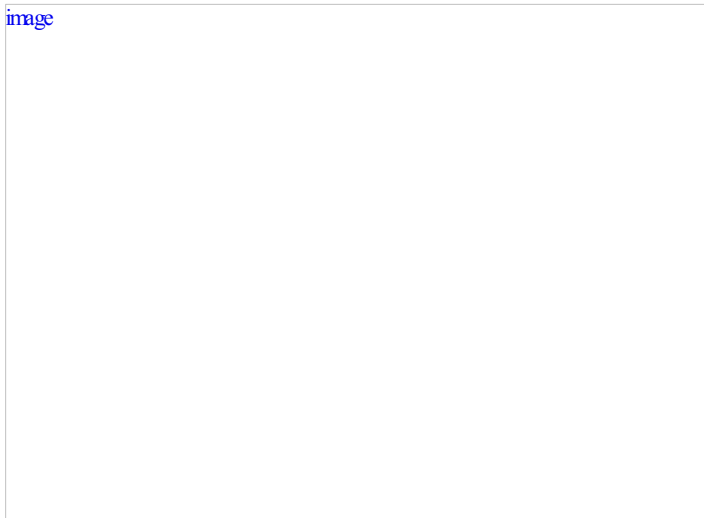
I hope you are reading this and going, "Well, duh!" but my experience with software projects of all sizes and in many organizations tells me this is not as obvious or easily attainable as we wish.

## Daily Build Quality

Daily quality matters a great deal in a software project because every day you make decisions based on your best understanding of how much work is left. When the average daily build has low quality, it is impossible to know how much work is left, and you make a lot of bad engineering decisions. As the number of contributing engineers increases (because we want to do more), the importance of daily quality rises rapidly because

the integration burden increases according to the probability of any single programmer's error. This problem is more than just not knowing what the number of bugs in the product is. If that were all the trouble caused then at least each developer would have their fate in their own hands. The much more insidious side-effect is when developers lack the confidence to integrate all of the daily changes into their personal work. When this happens there are many bugs, incompatibilities, and other issues that we can't know because the code changes have never been brought together on any machine.

I've prepared a graph to illustrate the phenomenon using a simple formula predicting the build breaks caused by a 1 in 100 error rate on the part of individual programmers over a spectrum of group sizes (blue line). A one percent error rate is good. If one used a typical rate it would be a little worse than that. I've included two other lines showing the build break probability if we cut the average individual error rate by half (red line) and by a tenth (green line). You can see that mechanisms that improve the daily quality of each engineer impacts the overall daily build quality by quite a large amount.



For a team the size of Windows, it is quite a feat for the daily builds to be reliable.

Our improvement in Windows 7 leveraged a big improvement in the Vista engineering system, an investment in a common test automation infrastructure across all the feature teams of Windows. (You will see here that there is an inevitable link between the engineering processes themselves and the organization of the team, a link many people don't recognize.) Using this infrastructure, we can verify the code changes supplied by every feature team before they are merged into the daily build. Inside of the feature team this infrastructure can be used to verify the code changes of all of the programmers every day. You can see in the chart how the average of 40 programmers per feature team balances the build break probability so that inside of a feature team the build breaks relatively infrequently.

For Windows 7 we have largely succeeded at keeping the build at a high level of quality every day. While we have occasional breaks as we integrate the work of all the developers, the automation allows us to find and repair any issues and issue a high quality build virtually every day. I have been using Windows 7 for my daily life since the start of the project with relatively few difficulties. (I know many folks are anxious to join me in using Windows 7 builds every day—hang in there!)

For fun I've included a couple pictures from our build lab where builds and verification tests for servers and clients are running 24x7:

clip\_image004



clip\_image006



## **Conclusion**

Whew! That seems like a wind sprint through a deep topic that I spend a lot of time on, but I hope you found it interesting. I hope you start to get the idea that we have been very holistic in thinking through new ways of working and improvements to how we engineer Windows through this example. The ultimate test of our thinking will be the quality of product itself. What is your point of view on this important software engineering issue?

# Reflecting on a few recent threads...

Steven Sinofsky | [2008-09-06T03:00:00+00:00](#)

---

When we kicked off this blog, the premise was a dialog—a two-way conversation about the engineering of Windows 7. We couldn't be happier with the way things have been going in this short time. 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 400 email messages (and answered quite a few) and all total we have had about 900 English language comments from about 500 different readers (with a few of you > 10 comments). Early numbers show we have about 10x that latter number in readers+page views.

A number of folks on the blog have asked for more details about how we build Windows—what's the feature selection process, the daily build process, globalization, and so on. And in keeping with our new tradition of seeing the other "side" of an issue, many folks have also said they feel like they have enough of that information and want to know the features. So in this post I want to offer a perspective on a couple of features that have been talked about a bunch, and also a perspective on talking about features and feature selection.

We love the response. We have seen that some topics have created a forum for folks to do a lot of asking for features, and we will do our best to respond in the context of what we set out to do, which is to have a discussion about how Windows 7 is engineered, including how we make choices about what goes in the product. I admit that it might be tempting (for me) to blog a big long list of features and then say "give us feedback". It is tempting because I have seen this in the past and it is a certainly an easy thing to do that might make people feel happier and more involved. However, there are some challenges with this technique that make these sorts of forums less than satisfying for all of us. First, it is "reactive" in that it asks you to just react to what you see. Absent a shared context we won't be remotely on the same page in terms of motivations, priorities, and so on. This is especially the case when a feature is early and we aren't really capable of "marketing" it effectively and telling the story of the feature. Second, a broad set of anecdotal feedback (that is free text) is not really actionable data and doesn't capture the dialog and discussion we are having. Making decisions this way is almost certain to not go well with the "half" of the folks who don't agree with the decision or prioritization. And third, there's a tendency to feel that feedback given yields action in that direction. These are some of the reasons why we have taken the approach of talking about how we are making Windows 7.

Some have suggested that we publish a list of features and then have a ranking/voting process. In fact some have gone as far as doing that for us on their own web sites. Thank you—these are interesting sites and we do look at them. But I think we can all agree that there is also a challenge that many folks are familiar with which is that a self-selected group provides one type of feedback which is likely to be different than a group that is selected intentionally as being representative. I was recalling an old episode of [Saturday Night Live](#), "[Larry the Lobster](#)", where for a toll call you could vote to save Larry from the stove or not. We all know that is a non-scientific poll, but we also don't even know if it is a non-scientific poll of views of animal rights or of food preferences. I think the value of voting on specific features goes beyond just entertainment, but we also have to spend the energy making sure we are thinking about the issues within the same context. We also want any sample of customers we do to be representative of either the broad base of customers or the specific target customer "segment".

Thus a big part of this blog is about creating a forum where we hear from each other about what is important and what our relative contexts are that we bring to the discussion. That's why we think about this as a dialog—it is not a question and answer, request and response, point and counter-point, or announcement and comment. Personally, I am genuinely benefiting from the dynamic nature of what we are going to blog about based on those participating in the blog. So this is much more like a social where we all come to meet and talk, than a business meeting where we each have specific goals or a training class where one party does all the talking.

In that spirit, it seems good to continue a conversation about a few points that have come up quite a bit and I think folks have been asking for a point of view on these. Each is worthy of a post on its own, but I also wanted to offer a point of view about some specific feature requests. Let's look at some topics that have come up as we have talked about performance or the overall Windows experience. Because this is "responding" to comments and input, there is a potential to delve into point/counter-point, I am hoping we can look back at the "context" discussions we have been having before we get too deep in debate.

## Profile-based Setup

In terms of feature ideas, a number of you have suggested that we offer a way at setup time to configure Windows for a specific scenario. Some



have suggested scenarios such as gaming, casual use, business productivity, web browsing, email, "lightweight usage", and so on. There is an implication in there that Windows could perform (speed, space, etc.) better if we tune it for a specific scenario along these lines, but in reality this assumption probably won't pan out in a consistent or general way. There are many ways to consider this feature—it could be one where we tweak the contents of the Start Menu (something admins do in corporations all the time), or the performance metrics for some low level components (disk block size, tcp/ip frame sizes, etc.) or the level of user interface polish (aka "eye candy" as some have called it), and so on. We've seen scenario or role-based setup as a very popular feature for Windows Server 2008. In the server environment, however, each of these roles represents a different piece of hardware (likely with different configurations) or perhaps a specific VM on a very beefy machine, and also represent very clearly understood "workloads" (file server, print server, web server).

The desktop PC (or laptop) is different because there is only a single PC and the roles are not as well defined. Only in the rarest cases is that PC **dedicated** to a single purpose. And as Mike in product planning blogged, the reality is that we see very few PCs that run **only** a specific piece of software and in nearly every study we have ever done, just about every PC runs at least one piece of software that other people do not run. So we should take away from this the difficulty in even labeling a PC as being role specific. Now there are role-specific times when using a PC, and for that the goal of an OS is to adapt well in the face of changing workloads. As just one example of this in Windows Vista, consider the work on making the indexer a low priority activity using the new [low-priority I/O APIs](#). I know some have mentioned that this is "something I always turn off" but the reality is that there is an upfront cost and then the ongoing cost of indexing is indeed very low. And this is something we have made significant improvements in for Desktop Search 4.0 (released as a download) and in Windows 7. The reality is that a general purpose OS should adjust to the workloads asked of it. We know things are not perfect, and we know many of you (particularly gamers) are looking for every single potential ounce of performance. But we also know that the complexity and fragility introduced by trying to "outsmart" core system services often overshadows the performance improvements we see across the broadest sampling of customers. There's a little bit of "mythbusters" we could probably embark on so -- how about sharing the systematic results you have achieved and we can address those in comments?

Another challenge would be in developing this very taxonomy. This is something I personally tried hard to do for Office 95 and Office 97. We thought we could have a setup "wizard" ask you how much you used Word, Excel, PowerPoint, and Access, or a taxonomy that asked you a profession (lawyer, accountant, teacher). From that we were going to pick not just which applications but which features of the applications we would install. We consistently ran into two problems. First, just arriving at descriptors or questions to "categorize" people failed consistently in usability tests—the classic problem when given a spectrum of choices people would peg all of them in the middle or would just "freeze up" feeling that none fit them (people don't generally like labels). Second, we always had the problem of either multiple users of the same PC or people who would change roles or usage patterns. It turns out our corporate customers learned this same thing for us and it became routine to "install everything" and thus began an era of installing the full suite of products and then training was used to narrow the usage scenarios.

The final challenge has been just how do you present this to customers and when. This sequence of steps, the out of box experience, or OOBE, is what you go through when you unbox a PC (the overwhelming majority of Windows customers get it this way) or run setup from a DVD (the retail "packaged product" customer). This leads to the next item which is looking to the OOBE as a place to do performance optimizations. Trying to solve performance at this step is definitely a challenge and leads to our "context" for the out of box experience.

## Out of Box Experience - "OOBE"

The OOBE is really the place that customers first experience Windows on a new PC. As many have read in reviews of competitive (to Windows PCs) products the experience goals most people have relate to "how fast can I get from packing knife to the web". For Windows 7 we are working closely with our OEM partners to make sure it is possible to deliver the most streamlined experience possible. Of course OEMs have a ton of flexibility and differentiation opportunities in what they offer as part of setting up a new PC, and what we want to do is make sure that the "core OS" portion of this is the absolute minimum required to get to the fun of using your PC.

By itself, this goal would run counter to introducing a "profiling" or "wizard" help gauge the intended (at time of purchase) uses/usage of a PC. That doesn't mean that an OEM could not offer such a profiled experience that could provide a differentiated OOBE experience, but it isn't one we would ask all customers to go through as part of the "core OS" installation.

I recognize many of you as PC enthusiasts have gone through the experience of setting up a Linux PC using one of the varieties of package managers—probably many times just to get one installation working right. As you've seen with these installs (especially as things have recently converged on one particular end-user focused distro), the number of ways you can produce a poorly running system exceeds the number of ways you can produce a fully functional (for your needs) setup. In practice, we know that many components end up depending on many others and ultimately this dependency graph is a challenge to manage and get right, even with a software dependency manager (like Windows Installer). As a

result, we generally see customers benefitting from a broad base of software on the machine so long as that does not have a high cost—developing that install is a part of developing the product, balancing footprint, architectural connections, system reliability, etc.

So our context for the out of box experience would be that we don't want to introduce complexity there, where customers are least interested in dealing with it as they want to get to the excitement of using their new PC. I think of it a bit like the car dealers who won't hand you the keys to your car until you sit and watch a DVD about the car and then get a guided tour of the car—if you're like me you're screaming "give me the keys and let me out of here". We think PC buyers are pretty much like that and our research confirms that around the world.

We also recognize that there are expert users who might want to adjust the running system for any variety of reasons (performance, footprint, surface area, etc.) We call this the "[Turning Windows Features On or Off](#)" which is the next item we've heard from you about.

## Windows Features

If we install the typical installation of Windows as one that is basically all the features in the particular SKU a customer purchased, then what about the customer that wants to tweak what is installed and remove things? Customers might want to remove some features because they just never use them and don't want to accidentally use them or carry with them the "code" that might run. Customers might be defining a role for the PC (cash register) and so making sure that specific features are never there. There are many reasons for this. For many releases Windows has had the ability to install or uninstall various features that are part of Windows. In Windows Vista this was made more robust as the features are removed from the running system but also remained available for reuse without the original DVD. We also made the list of features longer in Windows Vista.

For Windows 7, many have asked for us to make this list longer and have more features in it. This is something we are strongly considering for Windows 7 as we think it is consistent with the design goals of "choice and control" that you have seen us talk about here and quite a bit with Internet Explorer 8.0 beta 2.

Of course we have the same challenge that Linux distributions have which is you can quickly remove things could break other features by being removed, and then you have to have all the complexity of informing the customer of these "dependencies" and ultimately you end up feeling like everything is connected to everything else. On some OS installations this packaging works reasonably well because there is duplication of features (you pick from several file browsers, several web browsers, several office suites, several GUIs even). The core Windows OS, while not free from some duplication, does not have this type of configuration. Rather we ship a platform where customers can add many components as they desire.

For customers that wish to remove, replace, or just prevent access to Windows components we have several available tools:

- **Set Your Default Programs (or Set Program Access and Defaults).** In Vista these features allow you to set the default programs/handlers by file type or protocol. This was introduced in Windows XP SP1. In Vista the SYDP was expanded and we expect all Microsoft software to properly register and employ this mechanism. So if you want to have a default email program, default handler for GIF, or your choice of web browser this is the user interface to use. Windows itself respects these defaults for all the file types it manages.
- **Customizing the start menu or group policy.** For quite some time, corporate admins have been creating "role-based" PCs by customizing the start menu (or even going way back to program) to only show a specific set of programs. We see this a lot in internet cafes these days as well. The SPAD functionality takes this a step further and provides an end-user tool for removing access to installed email programs, web browsers, media players, instant messengers, and virtual machine runtimes.
- **Removing code.** Sometimes customers just want to remove code. With small footprint disks many folks have looked to remove more and more of Windows just to fit on SSDs. I've certainly seen some of the tiny Windows installations. The supported tool for removing code from Windows is to use the "[Turn Windows Features on and off](#)" (in Vista) user interface. There are over 80 features in this tool in premium Vista packages today.

Many folks want the list of Windows features that can be turned on / off to be longer and there have been many suggestions on the site for things to

make available this way. This is more complex because of the Windows platform—that is many developers rely on various parts of the Windows platform and just “assume” those parts are there. Whether it is a media player that uses the windows address book, a personal finance package that uses advanced print spooling, or even a brand new browser that relies on advanced networking features. These are real-world examples of common uses of system APIs that don’t seem readily apparent from the end-user view of the software.

Some examples are quite easy to see and you should expect us to do more along these lines, such as the TabletPC components. I have a PC that is a very small laptop and while it has full tablet functionality it isn’t the best size for doing good ink work for me (I prefer a 12.1” or greater and this PC is a 10” screen). The tablet code does have a footprint in memory and on the 1GB machine if I go and remove the tablet components the machine does perform better. This is something I can do today. Folks have asked about Photo Gallery, Movie Maker, Windows Mail, Windows Calendar...this is good feedback and good things for us to consider for Windows 7.

An important point is that a vast majority of things you remove this way consume little or no resources if you are not using them. So while you can reduce the surface area of the PC you probably don’t make it perform better. As one example, Windows Mail doesn’t slow you down at all if you don’t have any mail (or news) accounts configured. And to be certain you could hide access with SPAD or just change the default protocol handler to your favorite mail program. Another example is you can just change the association and never see photogallery launched for images if that is your preference. That means no memory is taken by these features.

This was a chance to continue our discussion around how we are learning from our discussion and some specifics that have come up quite a bit. I hope we are gaining a shared view of how we look at some of the topics folks have brought up.

So this turned into a record long post. Please don’t expect this too often ??

--Steven

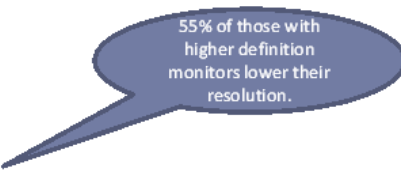
# Follow-up on High DPI resolution

Steven Sinofsky | [2008-09-13T03:00:00+00:00](#)

*One of the cool results of this dialog is how much interest there is in diving into the details and data behind some of the topics as expressed in the comment and emails. We're having fun talking in more depth about these questions and observations. This post is a follow-up to the comments about high DPI resolution, application compatibility, and the general problems with readability in many situations. Allow me to introduce a program manager lead on our Desktop Graphics team, Ryan Haveson, who will expand on our discussion of graphics and Windows 7. –Steven*

When we started windows 7 planning, we looked at customer data for display hardware, and we found something very interesting (and surprising). We found that roughly half of users were not configuring their PC to use the full native screen resolution. Here is a table representing data we obtained from the [Windows Feedback Program](#) which Christina talked about in an earlier post.

Monitor Default Resolution	% of Monitors Set to Default
1280X1024	56%
1400X1050	79%
1600X1200	32%
1680X1050	66%
1920X1050	39%
1920X1200	78%
<b>Set to Default</b>	<b>55%</b>



We don't have a way of knowing for sure why users adjust their screen resolution down, but many of the comments we've seen match our hypothesis that a lot of people do this to because they have difficulty reading default text on high resolutions displays. With that said, some users probably stumble into this configuration by accident; for example due to a mismatched display driver or an application that changed the resolution for some reason but did not change it back. Regardless of why the screen resolution is lower, the result is blurry text that can significantly increase eye fatigue when reading on a PC screen for a long period of time. For LCD displays, much of the blurriness is caused by the fact that they are made up of fixed pixels. In non-native resolution settings, this means that the system must render fractional pixels across fixed units, causing a blurred effect. Another reason for the relative blurriness is that when the display is not set to native resolution, we can't properly take advantage of our [ClearType text rendering technology](#), which most people (though not all) prefer. It is interesting to note that the loss of fidelity due to changing screen resolution is less pronounced on a CRT display than on an LCD display largely because CRTs don't have fixed pixels the way that LCDs do. However, because of the advantages in cost and size, and the popularity of the laptop PC, LCD displays are fast gaining market share in the installed base. Another problem with running in a non-native screen resolution is that many users inadvertently configure the display to a non-native aspect ratio as well. This results in an image that is both blurry and skewed! As you can imagine, this further exacerbates the issues with eye strain.

Looking beyond text, in these scenarios the resulting fidelity for media is significantly reduced as well. With the configuration that many users have, even if their hardware is capable, they are not able to see native "high def" 720p or 1080p TV content, which corresponds to 1280x720 and 1920x1080 screen resolutions respectively. The PC monitor has traditionally been the "high definition" display device, but without addressing this problem we would be at risk of trailing the TV industry in this distinction. While it is true that only about 10% of users have a truly 1080p capable PC screen today, as these displays continue to come down in price the installed base is likely to continue to grow. And you can bet that there will be another wave of even higher fidelity content in the future which users will want to take advantage of. As an example, when displays get to 400 DPI they will be almost indistinguishable from looking at printed text on paper. Even the current generation of eBook readers with a DPI of ~170 look very much like a piece of paper behind a piece of glass

From this we see that there is a real end user benefit to tap into here. It turns out that there is existing infrastructure in Windows called "High DPI" which can be used to address this. High DPI is not a new feature for Windows 7, but it was not until Vista that the OS user-interface made significant investments in support for high DPI (beyond the infrastructure present earlier). To try this out in Vista, rt. Click desktop -> personalize and select "Adjust Font Size (DPI)" from the left hand column. Our thinking for Windows 7 was that if we enable high DPI out of the box on capable displays, we will enable users to have a full-fidelity experience and also significantly reduce eye strain for on-screen reading. There is even infrastructure available to us to detect a display's native DPI so we can do a better job of configuring default settings out of the box. However, doing this will also open up the door to expose some issues with applications which may not be fully compatible with high DPI configurations.

One of the issues is that for GDI applications to be DPI aware, the developer must write code to scale the window frame, text size, graphical buttons, and layout to match the scaling factor specified by the DPI setting. Applications which do not do this may have some issues. Most of these issues are minor, such as mismatched font sizes, or minor layout artifacts, but some applications have major issues when run at high DPI settings.

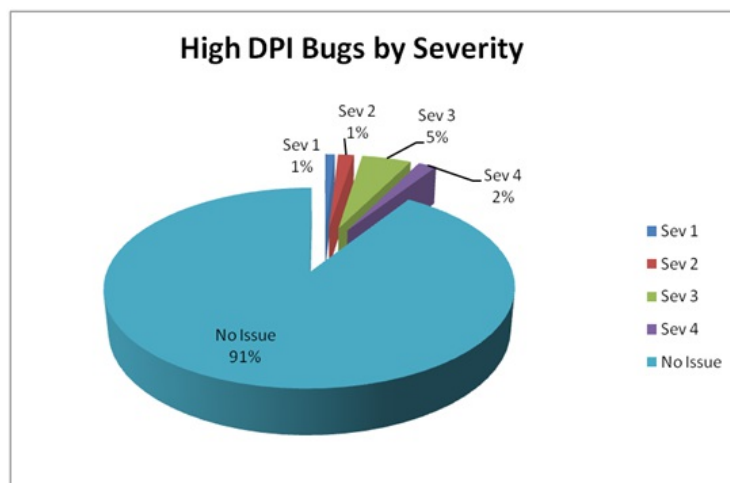
There are some mitigations that we can do in Windows, such as automatic scaling for applications which are not declared DPI aware (see [Greg Schechter's blog](#) on the subject), but even these mitigations have problems. In the case of automatic scaling, applications which are not DPI aware are automatically scaled by the window manager. The text size matches the user preference, but it also introduces a blurry effect for that application's window as a result. For people who can't read the small text without the scaling, this is a necessary feature to make the high DPI configuration useful. However, other customers may only be using applications that scale well at high DPI or may be less impacted by mismatched text sizes and may find the resulting blurry effect caused by automatic scaling to be a worse option. Without a way for the OS to detect whether an application is DPI aware or not, we have to pick a default option. It always comes back to the question of weighing the benefits and looking at the tradeoffs. In the long term, the solution is to make sure that applications know how to be resolution independent and are able to scale to fit the desired user preference, which requires support in both our tools and documentation. The challenge for a platform is to figure out how to get there over time and how to produce the best possible experience during the transition.

### Short term vs. long term customer satisfaction

Using the model of high definition TV, we can see that in the long term it is desirable to have a high fidelity experience. The only problem is that even though the high DPI infrastructure has been around for several windows releases (in fact there is an [MSDN article](#) dated 2001 on making applications DPI aware), we were not sure how many applications are actually tested in these configurations. So we were faced with an unquantified potential short term negative customer impact caused by enabling this feature more broadly. The first thing we did is to quantify the exposure. We did this by performing a test pass with over 1,000 applications in our *app compat* lab to see how they behave at high DPI settings. The results we found are shown below, which shows the distribution of issues for these 1000 applications.

*One quick thing, when we say "bug" we mean any time software behaves in a manner inconsistent with expectations—so it can be anything from cosmetic to a crash. We categorize the severity of these bugs on a scale from 1 to 4, where Sev 1 is a really bad issue (such as a crash and/or loss of data or functionality) and Sev 4 is an issue which is quite subtle and/or very difficult to reproduce.*

It turns out that most applications perform well at high DPI, and very few applications have major loss of functionality. Of course, it is not the ones that work well which we need to worry about. And if 1% of applications have major issues at high DPI, that could be a significant number. So we took a look at the bugs and put them into categories corresponding to the issue types found. Here is what we came up with:



What we found was that one of the most significant issues was with clipped UI. Looking into this deeper, it became apparent that most of these cases were in configurations where the effective screen resolution would be quite low (800x600 or lower). Based on this, we were able to design the configuration UI in such a way that we minimized the number of cases where users would configure such a low effective resolution. One by one we looked at the categories of issues and when possible, we came up with mitigations for each bucket. Of course, the best mitigation is prevention and so High DPI is a major focus for our developer engagement stories for PDC, WinHEC, and other venues coming up.

### **Aggregate vs. individual user data**

One thing for us to look at is how many users are taking advantage of high DPI today (Vista/XP). Based on the data we have, only a very small percentage of users are currently enabling the high DPI feature. This could easily be interpreted as a clear end user message that they don't care about this feature or have a need for this feature. An alternate explanation could be that the lack of adoption is largely because XP and Vista had only limited shell support for high DPI, and the version of IE which shipped on those platforms had significant issues with displaying mismatched font sizes and poorly scaled web pages. Also, we do know anecdotally that there are users who love this feature and have used it even before Vista. Once again, we have to make an interpretation of the data and it is not always crystal clear.

### **Timing: is this the right feature for the market in this point in time?**

Fortunately, we don't have a "chicken and egg" problem. The hardware is already out in the field and in the market, so it is just a matter of the OS taking advantage of it. From a software perspective, most of the top software applications are DPI aware (including browsers with improved zooming, such as [IE 8](#)), but there remain a number of applications which may not behave well at high DPI. Another key piece of data is that display resolution for LCD panels is reaching the maximum at standard DPI. For these displays, there is no reason to go beyond 1900x1200 without OS support for high DPI because the text would be too small for anyone to read. Furthermore, this resolution is already capable of playing the highest fidelity video (1080p) as well as 2 megapixel photos. The combination of existing hardware in the field, future opportunity to unlock better experiences, and the fact that the hardware is now blocked on the OS and the software speak to this being the right timing.

### **Conclusion**

Looking at customer data helps us identify ways to improve the Windows experience. In this case, we saw clearly that we had an opportunity to help users easily configure their display such that they would enjoy a high fidelity experience for media as well as crisp text rendered at an appropriate size. With that said, anytime we invest in a feature that can potentially impact the ecosystem of Windows applications we want to be careful about bringing forward your investments in software. We also want to make sure that we engage our community of ISVs early and deeply so they can take advantage of the platform work we have done to seamlessly deliver those benefits to their customers. In the meantime, the internal testing we did and the data that we gathered was critically important to helping us make informed decisions along the way. High DPI is a good example of the need for the whole ecosystem to participate in a solution and how we can use the customer data in the field, along with internal testing, to determine the issues people are seeing and to help us select the best course of action.

--Ryan

# More Follow up to discussion about High DPI

Steven Sinofsky | [2008-09-16T03:00:00+00:00](#)

---

*Excellent! What a fun discussion we've been having on High DPI. It has been so enriching that Ryan wrote up a summary of even more of the discussion. Thanks so much! --Steven*

There have been quite a few comments posted regarding high DPI, along with some lively discussion. Most of what has been said has been good anecdotal examples which are consistent with the data we have collected. For the areas where we didn't have data, the comments have helped to validate many of our assumptions for this group. It is also clear that there are some areas of this feature which are confusing, and in some cases there is a bit of "myth" around what is ideal, what is possible, and what is there. This follow up post is mostly to summarize what we have heard, and to provide some details around the areas where there has been a bit of confusion.

Here is a list of our top "assumptions" which have been echoed by the comments posted:

- Most people adjust the screen resolution either to get larger text, or because it was an accident
- There is a core of people who know about high DPI and who use it
- Some people prefer more screen real-estate while others people prefer larger UI
- Discoverability of the DPI configuration is a concern for some
- App compat is a common issue, even a "deal breaker" in some cases
- IE Scaling is one of the top issues listed (see [IE8](#) which fixes many of these!)
- Lots of complexities/subtleties and it is pretty hard to explain this feature to most people

There have also been a number of areas where there has been a bit of confusion:

- Is it true that if everything were vector-based, these problems would all go away?
- Shouldn't this just work without developers having to do anything?
- How is this different from per-application scaling like IE zooming?
- Should DPI be for calibration or for scaling?

Most of these topics have been covered to some degree in the comments, but since there has been so much interest, we decided to go into a bit more details around a few of the top issues/concerns.

## Vector Graphics vs. Raster Graphics

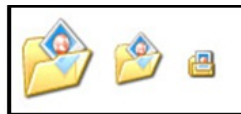
With PCs, there is always the hope of a "silver bullet" technology which solves all problems making life easy for users, developers, and designers

across the board. As an example, some of the comments to the original posting suggested that if we just made the OS fully vector based, these scaling problems would go away. It turns out that there are several issues with using vector graphics which are worth explaining.

The first issue is that oftentimes when an icon gets to be small in size, it is better to use an alternate representation so that the meaning is clearer. Notice the icons below. In this case, the designer has chosen a non-perspective view for the icon when it is rendered at it's smallest size.



This is because the designer felt that the information expressed by the icon was clearer with a straight-on view at the smallest size. Here is another example illustrating this point:



Of course, this means that the designer must create multiple versions of the source image design, so there is additional complexity. The point here is that there is a tradeoff that must be made and the tradeoff is not always clear.

Even when one vector source is used, it is common to have size-dependent tweaking to make sure that the result is true to what the designer had in mind. Imagine a vector graphic which has a 1-pixel line at 128x128 that gets scaled down by 1/2 to 64x64. The display has no way of rendering a perfect 1/2 pixel line! In many cases the answer is that the renderer will “round” to a nearby pixel line. However, doing this inherently changes the layout of the sub-elements of the image. And there is the question of, “which pixel line to round to?” If the designer does not hand tune the source material, it will be up to the rendering engine to make this decision, and that can result in undesirable effects. One could say that this should therefore define rules about what elements should be use in a vector, but that only further restricts what concepts can be represented.

It turns out that even the TrueType fonts which we use in Windows are hand-tuned with size-dependant information in order to make the result as high quality as possible. Most of the TrueType rendering pipeline is based on algorithmic scaling of a vector source, but there are size-dependent, hand-coded “hints” in TrueType which the designer specifies to direct the system how to handle edge cases, such as lines falling between pixel boundaries. Here is a link describing this in more detail: <http://blogs.msdn.com/fontblog/archive/2005/10/26/485416.aspx>

It is not even true that vector graphics are necessarily smaller in size (especially for small images). Most designers create graphics using an editor which builds up an image using many layers of drawings and effects. With bitmap based graphics, designers will “flatten” the layers before adding it to a piece of software. Most designers today pay little attention to the size of the layers because the flattening process essentially converts it to a fixed size based on the image resolution. With vector graphics, there is no such flattening technique so designers need to carefully consider the tools that they use and the effects that they add to make sure that their icon is not extremely large. I spent some time with one of our designers who had a vector graphic source for one of our bitmaps in Windows and the file was 622k! Of course that file size is fixed regardless of the resulting resolution, but that huge file flattens into this 23k PNG bitmap.



Of course, a hand-tuned vector based representation of this could be probably made smaller if the size constraints were part of the design time process. But that would be an additional constraint put on the designer, and one which they would need to learn how to do well.

## How do we help developers?

For applications that need to carefully control the layout and graphics, or scale the fidelity of the images based on the available resolution, having a way of specifying specific pixel locations for items is important to get the best result. This is as true on the Mac as it is on the PC (see <http://developer.apple.com/releasenotes/GraphicsImaging/RN-ResolutionIndependentUI/>). There is often a belief that if we just provided the right tools or the right framework then all these problems would go away. We all know that each set of tools and each framework have their own set of tradeoffs (whether that is Win 32, .net, or HTML). While there is no silver bullet, there are things we can do to make writing DPI aware applications easier for developers. As an example, there are two important upcoming ecosystem events in which we will be talking in detail about High DPI. We will also have materials which we will be making available to developers which will help educate them on how to convert existing applications to be DPI aware. The first event is [Microsoft Professional Developer Conference](#), where we will talk about High DPI as part of the talk “Writing your Application to Shine on Modern Graphics Hardware ([link](#))”. The second is the [Windows Hardware Engineering Conference](#), in which we will be discussing high DPI as part of the “High Fidelity Graphics and Media” track ([link](#)).

## Help with App Compat Issues

There have been several posts on app compat and high DPI (for example bluv’s comment). There have also been comments talking about the complexity and understandability of the High DPI configuration. In some cases the app compat issues can be mitigated by enabling or disabling the automatic scaling feature. This can be changed globally by going to the DPI UI, clicking the button labeled “Custom DPI” and changing the checkbox labeled, “Use Windows XP style DPI scaling”. When this checkbox is unchecked, applications which are not declared to be DPI aware are automatically scaled by the window manager. When it is checked, automatic scaling is disabled globally. It is interesting to note that for DPI settings < 144 DPI, this box is checked by default, and for DPI settings  $\geq$  144 it is unchecked by default. In some cases, changing the default settings can result in a better experience depending on the applications that you use and your DPI setting. It is also interesting to note that automatic scaling can be turned off on a per application basis using the Vista Program Compatibility properties. Here is a link for more info on how to do that: <http://windowshelp.microsoft.com/Windows/en-US/help/bf416877-c83f-4476-a3da-8ec98dcf5f101033.mspx>. (Look at the section for “Disable Display Scaling on high DPI settings”.)

## How is DPI scaling different from per-application scaling (like IE Zoom)?

A typical application UI is made up of a content window and a frame UI. The frame UI is where the menu items and toolbar buttons are. The content window is the “document view”. For example, in IE the content window is the actual webpage. It turns out the code required to support high DPI scaling for the content windows is the same code required to do the zooming feature. In fact, for the content window, IE8 simply uses the high DPI setting to configure the default zoom factor (see [DPI Scaling and Internet Explorer 8](#) for more details). However, high DPI has the additional side effect of scaling the size of the frame UI. Since most people use the scaling feature to make text larger to be more readable, it makes sense to scale the frame UI as well, since the text in the menu items and toolbar tooltips will also scale. In a sense if there is per-application scaling that is really about the content area, and applications will support that as developers see the customer need. DPI scaling makes the UI elements of all applications render similarly.

## Shouldn’t DPI really be used for calibrating the screen (so “an inch is an inch”)?

Some have suggested that we should just use high DPI as a way to calibrate the screen so that the physical size of an object is the same regardless of the display. This makes a ton of sense from a logical perspective. The idea would be to calibrate the display so “in inch is an inch”. We thought about doing this, but the problem is that it does not solve the end user need of wanting to have a way to configure the size of the text and the UI. If we then had a separate “global scale” variable, this would mean that application developers would need to pay attention to both metrics, which would add complexity to the developer story. Furthermore, if a user feels that the UI is too small, should it be up to the developer or the user to set the preference of how big the UI should be? In other words if the designer wants the button to be an inch, but the user wants the button to be 1.5 inches to make it easier to use, who should decide? The way the feature works today, it is up to the user to set their preference, but it is up to the application developer to make sure that the user preference is honored.

Once again, it is really great to see so much interest in high DPI. We certainly have some challenges ahead of us, but in many ways it seems like the ecosystem is ripe for this change. Hopefully this follow up post helped to consolidate some of feedback which we have heard on the previous post and explain some of the complexities of this feature in more detail.

--Ryan Haveson

# The "Ecosystem"

Steven Sinofsky | [2008-09-18T03:00:00+00:00](#)

---

In the emails and comments, there are many topics that are raised and more often than not we see the several facets or positions of the issue. One theme that comes through is a desire expressed by folks to choose what is best for them. I wanted to pick up on the theme of choice since that is such an incredibly important part of how we approach building Windows—choice in all of its forms. This choice is really because Windows is part of an *ecosystem*, where many people are involved in making many choices about what types of computers, configuration of operating system, and applications/services they create, offer, or use. Windows is about being a great component of the ecosystem and what we are endeavoring to do with Windows 7 is to make sure we do a great job on the ecosystem aspects of building Windows 7.

Ecosystem and choice go hand in hand. When we build Windows we think of a number of key representatives within the ecosystem beyond Windows:

- PC makers
- Hardware components
- Developers
- Enthusiasts

Each of these parties has a key role to play in delivering on the PC experience and also in providing an environment where many people can take a PC and provide a tailored and differentiated experience, and where companies can profit by providing unique and differentiated products and services (and choice to consumers). For Windows 7 our goals have been to be clearer in our plans and stronger in our execution such that each can make the most of these opportunities building on Windows.

**PC Makers (OEMs)** are a key integration point for many aspects of the ecosystem. They buy and integrate hardware components and pre-install software applications. They work with retailers on delivering PCs and so on. The choices they provide in form factors for PCs and industrial design are something we all value tremendously as individuals. We have recently seen an explosion in the arrival of lower cost laptops and laptops that are ultra thin. Each has unique combinations of features and benefits. The choice to consumers, while sometimes almost overwhelming, allows for an unrivaled richness. For Windows 7 we have been working with OEMs very closely since the earliest days of the project to develop a much more shared view of how to deliver a great experience to customers. Together we have been sharing views on ways to provide differentiated PC experiences, customer feedback on pre-loaded software, and partnering on the end-to-end measurement of the performance of new PCs on key metrics such as boot and shutdown.

**Hardware components** include everything from the CPU through the “core” peripherals of i/o to add-on components. The array of hardware devices supported by Windows through the great work of independent hardware vendors (IHVs) is unmatched. Since Windows 95 and the introduction of plug-and-play we have continued to work to improve the experience of obtaining a new device and having it work by just plugging it in—something that also makes it possible to experience OS enhancements independent of releases of Windows. This is an area where some express that we should just support fewer devices that are guaranteed to work. Yet the very presence of choice and ever-improving hardware depends on the ability of IHVs to provide what they consider differentiated experiences on Windows, often independent of a specific release of Windows. The device driver model is the core technology that Microsoft delivers in Windows to enable this work. For Windows 7 we have committed to further stabilization of the driver model and to pull forward the work done for Windows Vista so it seamlessly applies to Windows 7. Drivers are a place where IHVs express their differentiated experience so the breadth of choice and opportunity is super important. I think it is fair to say that most of us desire the experience where a “clean install” of Windows 7 will “just work” and seamlessly obtain drivers from Windows Update when needed. Today with most modern PCs this is something that does “just work” and it is a far cry from even a few years ago. As with OEMs we have also been working with our IHV partners for quite some time. At WinHEC we have a chance to show the advances in Windows 7 around devices and the hardware ecosystem.

**Developers** write the software for Windows. Just as with the hardware ecosystem, the software ecosystem supports a vast array of folks building for the Windows platform. Developers have always occupied a special place in the collective heart of Microsoft given our company roots in providing programming languages. Each release of Windows offers new APIs and system services for developers to use to build the software they

want to build. There are two key challenges we face in building Windows 7. First, we want to make sure that programs that run on Windows Vista continue to run on Windows 7. That's a commitment we have made from the start of the project. As we all know this is perhaps the most critical aspect of delivering a new operating system in terms of compatibility. Sometimes we don't do everything we can do and each release we look at how we can test and verify a broader set of software before we release. Beta tests help for sure but lack the systematic rigor we require. The telemetry we have improved in each release of Windows is a key aspect. But sometimes we aren't compatible and then this telemetry allows us to diagnose and address post-release the issue. If you've seen an application failure and were connected to the internet there's a good chance you got a message suggesting that an update is available. We know we need to close the loop more here. We also have to get better at the tools and practices Windows developers have available to them to avoid getting into these situations—at the other end of all this is one customer and bouncing between the ISV and Microsoft is not the best solution.

Our second challenge is in providing new APIs for developers that help them to deliver new functionality for their applications while at the same time provide enough value that there is a desire to spend schedule time using these APIs. Internally we often talk about "big" advances in the GUI overall (such as the clipboard or ability to easily print without developing an application specific driver model). Today functionality such as networking and graphics play vital roles in application development. We've talked about a new capability which is the delivery of touch capabilities in Windows 7. We've been very clear about our view that 64-bit is a place for developers to spend their energy as that is a transition well underway and a place where we are clearly focused.

**Enthusiasts** represent a key enabler of the ecosystem, and almost always the one that works for the joy of contributing. As a reader of this blog there's a good chance you represent this part of the ecosystem—even if we work in the industry we also are "fans" of the industry. There are many aspects to a Windows release that need to appeal the enthusiasts. For example, many of us are the first line of configuration and integration for our family, friends, and neighbors. I know I spent part of Saturday setting up a new wireless network for a school teacher/friend of mine and I'm sure many of you do the same. Enthusiasts are also the most hardcore about wanting choice and control of their PCs. It is enthusiasts sites/magazines that have started to review new PCs based on the pre-installed software load and how "clean" that load is. It is enthusiasts that push the limits on new hardware such as gaming graphics. It is enthusiasts who are embracing 64-bit Windows and pushing Microsoft to make sure the ecosystem is 64-bit ready for Windows 7 (we're pushing of course). I think of enthusiasts as the common thread running through the entire ecosystem, participating at each phase and with each segment. This blog is a chance to share with enthusiasts the ins and outs of all the choices we have to make to build Windows 7.

There are several other participants in the ecosystem that are equally important as integration points. The **system builders** and **VARs** provide PCs, software, and service for small and medium businesses around the world. Many of the readers of this blog, based on the email I have received, represent this part of the ecosystem. In many countries the **retailers** serve as this integration point for the individual consumer. For large enterprise customers the **IT professionals** require the most customization and management of a large number of PCs. Their needs are very demanding and unique across organizations.

Some have said that the an ecosystem is not the best approach that we could do a much better job for customers if we reduce the "surface area" of Windows and support fewer devices, fewer PCs, fewer applications, and less of Windows' past or legacy. Judging by the variety of views we've seen I think folks desire a lot of choice (just in terms of DPI and monitor size). Some might say that from an engineering view less surface area is an easier engineering problem (it is by definition), but in reality such a view would result in a radical and ever-shrinking reduction in the choices available for consumers. The reality is engineering is about putting constraints in place and those constraints can also be viewed as assets, which is how we view the breadth of devices, applications, and "history" of Windows. The ecosystem for PCs depends on opportunities for many people to try out many ideas and to explore ideas that might seem a bit crazy early on and then become mainstream down the road. With Windows 7 we are renewing our efforts at readying the ecosystem while also building upon the work done by everyone for Windows Vista.

The ecosystem is a pretty significant in both the depth and breadth of the parties involved. I thought for the purposes of our dialog on this blog it is worth highlighting this up front. There are always engineering impacts to balancing the needs each of the aspects of the ecosystem. Optimizing entirely along one dimension sometimes seems right in the short term, but over any period of time is a risky practice as the benefits of a stable platform that allows for differentiation is something that seems to benefit many.

With Windows 7 we committed up front to doing a better job as part of the PC ecosystem.

Does this post reflect your view of the ecosystem? How could we better describe all those involved in helping to make the PC experience amazing for everyone?

--Steven

# User Interface: Starting, Launching, and Switching

Steven Sinofsky | [2008-09-23T03:00:00+00:00](#)

---

*Where to Start? In this post, Chaitanya Sareen, a senior program manager on the Core User Experience team, sets the engineering context for the most frequently used user-interface elements in Windows – the Windows Taskbar. -- Steven*

It should come as no surprise that we receive lots of feedback about the taskbar and its functionality in general. It should also come as no surprise that we are constantly trying to raise the bar and improve the taskbar experience for our customers, while making sure we bring forward the familiarity and benefits (and compatibility) of the existing implementation and design. In this post, we would like to provide some insight into that unassuming bar most likely at the bottom of your Windows desktop. Let's take a closer look at its various parts, data we've collected and how this learning will inform the engineering of Windows 7.

## Taskbar Basics

Our taskbar made its debut way back in Windows 95 and its core functionality remains the same to this day. In short, it provides launching, switching and “whispering” functionality. Figure 1 shows the Vista taskbar and calls out its basic anatomy. Notable pieces are the taskband, Quick Launch, the Start Menu, Desktop Toolbars (aka Deskbands) and the Notification Area. Collectively, these components afford some of the most fundamental controls for customers to start, manage and monitor their tasks.

[Image of Windows taskbar pointing out names of various regions.](#)



**Fig. 1: Windows Taskbar Anatomy**

### Taskband: The faithful window switcher

The taskband is one of the most important parts of the taskbar. It hosts buttons which represent most of the windows open on the desktop. Think of the taskband as a remote control for your computer—you can switch windows just like switching channels on a TV. The idea of switching *windows* is the most fundamental aspect of the Windows taskbar. Other operating systems also have bars at the bottom of their screen, although theirs may have different goals. For example, Mac OS X has a Dock which is primarily a *program* launcher and a *program* switcher. Clicking on an icon on the Dock usually brings up all the windows of a running program. In 2003 Apple introduced a *window* switcher known as Expose which provides a different visual approach to our long-standing Alt-tab interface (Vista's Flip 3D is yet another visual approach). These dedicated window switchers all aim to provide customers with a broad view of their open windows, but they each require the customer to first invoke them. The taskband on the other hand, is designed to always be visible so that windows remain within quick access of the mouse. This makes the taskbar the most prominent window switcher of the Windows operating system.

Two noteworthy taskbar changes were introduced in the last eight years. Windows XP ushered in grouping which allows taskbar buttons to collapse into a single button to save space and organize windows by their process. Vista presented taskbar thumbnails. These visual representations give customers more information about the window they are looking for. While valuable, interfaces like the taskbar, Alt-tab and even Apple's own Expose reveal that thumbnails are not always large enough to guarantee recognition of a window. Their value further degrades when they have to shrink to accommodate many open windows, which is feedback we receive from those that often have lots of running programs x lots of open windows.

## The Start Menu: the Windows launch pad

The Start Menu has always been anchored off the taskbar as a starting point for the customer's key tasks such as launching or accessing system functionality. Microsoft of course used term "Start" and prominently labeled the Start Menu's button as such. You may even recall the huge marketing campaign for Windows 95 which featured the Rolling Stone's "Start Me Up". In all seriousness though, our research showed that many customers didn't always know where to go on their computer to start a task. When a customer was placed in front of a Windows 95 machine she now had a clearly labeled place to start. And yes, we've heard the joke that you click *start* to *shutdown* your machine. Speaking of shutdown, we did encounter some challenges with the power options in Vista's Start Menu. The goal was to bubble-up and advertise the sleep option so that customers enjoy a faster resume. However, we now know despite our good intentions, customers are opening that fly-out menu and selecting other options. We're looking into improving this experience.

The Start Menu has undergone many changes over the years. One notable change was the appearance of a MFU (most frequently used) section in Windows XP that suggests commonly (well frequently) used programs. The goal here was to save the customer time by not having to always go to All Programs. Since these items appear automatically based on usage, no manual customization was even required. All Programs itself has undergone several iterations. Customer feedback revealed that people encountered difficulty in traversing the original All Programs fly-out menu. It wasn't uncommon to have your mouse "fall off" the menu and then you'd have your restart the task all over again. This was particularly the case for laptop customers using a trackpad. It also didn't help that expanding this menu suddenly filled the entire desktop which looked visually noisy and it also required lots of mouse movement. And of course, for machines with large number of items and/or groups it was especially complex, and even more so on small screens. Vista introduced a single menu that requires less mouse acrobatics.

Search was another important addition to the Start Menu that makes launching even easier. This new feature in Vista provides fast access to programs and files without the need to use a mouse at all. Typing in a phrase quickly surfaces programs, files and even e-mails. We've received many positive comments from enthusiasts who feel this is a key performance win in terms of "time to launch". It may be interesting to note that Start Menu's search is optimized to first return program results as this was viewed as the most common scenario among our customers (using some of the Desktop Search technology). Search even permits customers to use parameters to further scope their queries. For instance, one can use "to:john" or "from:jane" to find a specific mail directly from the Start Menu. Our advanced customers also enjoy the benefit of using the Start Menu's search as a replacement of the Run Dialog. Just as they would type the name of an executable along with some switches in the dialog, they can now just type this directly into the search field. We could (and will) dedicate an entire blog post to search alone, but hopefully you get a sense of how search certainly provides a powerful launch alternative to mouse navigation.

## Quick Launch: Launching at your fingertips

Quick Launch provides a way for customers to launch commonly used programs, files, folders and websites directly off the taskbar. It was introduced to Windows 95 by Internet Explorer 4.0 with the Windows Desktop Update. Customizing Quick Launch is as simple as dragging shortcuts into to this area. It saves you a trip to the Start Menu, the desktop or a folder when you want to launch something. An interesting feature of Quick Launch that you may not be aware of is that it has always supported large icons (unlock the taskbar, right-click on Quick Launch and click on large icons under "View") as seen in figure 2. Of course growing the icons begins to intrude on the real-estate of the taskband which is one of the reasons we have not enabled this configuration by default. As an aside, Windows XP had Quick Launch turned off by default in an attempt to reduce the number of different launching surfaces throughout Windows. Based on your feedback, we quickly rectified this faux pas and Quick Launch was turned on by default again. Don't mess with quick access to things people use every day! We heard you loud and clear.

[clip\\_image004](#)



**Fig. 2: Large Icons in Quick Launch.** *Large icons on the taskbar have been supported since Windows 95 with IE 4*

## Desktop Toolbars (aka Deskbands): Gadgets for your taskbar

Desktop Toolbars offer extensible and specialized functionality at the top-level of the taskbar. This functionality also came to the taskbar via Internet Explorer 4.0 back in the '90s. You can access toolbars by right-clicking on your taskbar and expanding "Toolbars". Personally, I like to think of Desktop Toolbars as an early type of gadgets for the Windows platform. Over the years developers have written various toolbars including controls for background music (e.g. Windows Media Player's mini-mode shown in figure 1), search fields, richer views of laptop batteries, weather forecasts and many more.

One of the original scenarios of Desktop Toolbars was to allow customers to launch items directly off the taskbar. In fact, Quick Launch itself is a special type of toolbar that surfaces shortcuts in the Quick Launch folder. Did you know you can even create your own toolbar for any folder on your computer so that you have quick access to its contents (from the Toolbar menu, select "New Toolbar" and just choose the folder you'd like to access)? Apple's latest OS introduced similar functionality to the Dock called Stacks. While I think their implementation of this feature is generally more visually appealing, it is interesting to note they recently released a new list representation that matches our original functionality. Seems like we both agree a simple list is usually the most efficient way to parse and navigate lots of items.

After extolling all the greatness of Desktop Toolbars, we must also admit they introduce several challenges. For starters, they aren't the easiest thing to discover. They also take up valuable space on an already busy taskbar. Most importantly though, they don't always solve the customer goal. Sure you can have a folder's contents accessible off your taskbar, but what if the files you want quick access to aren't located in a single place? These are design challenges we intend to tackle.

## Notification Area: The whisperer

The Notification Area is pretty much what you expect—an area for notifications. It was an original part of the taskbar and it was designed to whisper information to the customer. Here you can easily monitor the system, be alerted to the state of a program or even check the time. Icons were the predominant way to convey information until later versions of Windows introduced notification balloons that provide descriptive alerts with text. Also added was a collapsible UI that hid inactive icons so the taskbar would appear cleaner.

With more developers leveraging its functionality, the Notification Area has grown in popularity over the years. Some may observe that it has changed from a subtle whisperer to something louder. Based upon the feedback we've collected from customers, we recognize the Notification Area could benefit from being less noisy and something more controllable by the end-user.

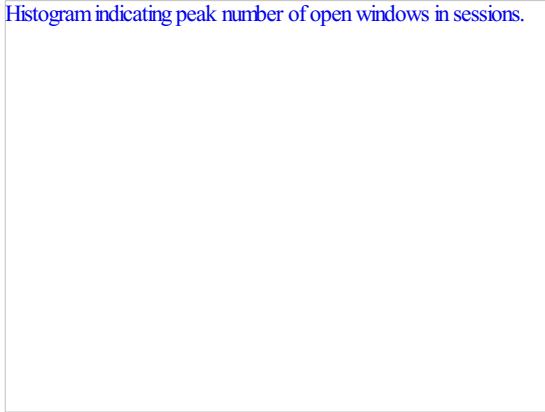
## Show Me the Data

Earlier posts to this blog discussed how customers can voluntarily and anonymously send us data on how they use our features. We use these findings to help guide our designs. Please note that data do not design features for us, but they certainly help us prioritize our investments as well as validate our approach. All too often we're all guilty of saying something like "we know *everyone* does <x>" or "all users do <y>". Given the reliability and statistical accuracy of this data, we can speak with more real-world accuracy about how things are in used in practice. Let's look at some interesting information we have collected about how our customers use the taskbar.

Figure 3 provides some of the most important data about the taskbar—window count. On average, we know that a vast majority of our customers encounter up to 6-9 simultaneous windows during a session (a session is defined as a log in / log out or 24 hours—whichever occurs first). It goes without saying that the taskbar should work for the entire distribution of this graph, but identifying the "sweet spot" helps focus our efforts on the area that matters most to the most amount of customers. So, we know that if we nail the 6-9 case and we work well for the 0-5 as well as the 10-14 scenarios, we've addressed almost 90% of typical sessions.

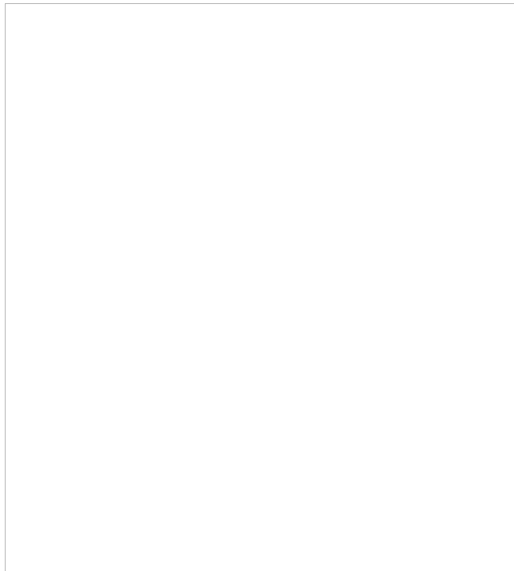


Histogram indicating peak number of open windows in sessions.



**Fig. 3: What's the maximum number of windows opened at a time?**

Figures 4 and 5 help us understand how customers customize their taskbars. We could probably spend an entire post focused solely on how we determine the options we expose. Perhaps another time we'll tackle the paradox of choice and how options stress our engineering process yet also make the product more fun for a set of customers. Until then, let's see what conclusions we can draw from these findings. The most obvious takeaway is that most customers do not change the default settings, which are a simple right-click Properties away. For example, it may be interesting to note how often end-users relocate the taskbar to other regions of the screen—less than 2% of sessions have a taskbar that's not at the bottom of the screen. We also know that some small percentage of machines accidentally relocate the taskbar and more often than not end-users have difficulty undoing such a state—though our data does not differentiate this situation. This data does not necessarily mean we would remove relocation functionality, but rather we could prioritize investments in a default horizontal taskbar over other configurations.



**Fig. 4: How do people customize their taskbar?** *The red number indicates percentage of sessions in which the corresponding checkbox is enabled.*

LOCATION	SESSION PERCENT
Bottom (default)	98.4%
Top	1.02%
Left	0.36%
Right	0.21%

**Fig. 5: Where do people put their taskbar?**

Figure 6 provides some insight into the Windows Media Player Desktop Toolbar. The Windows UX Guidelines prescribe that to create a toolbar on the customer's taskbar, you must call a Windows Shell API that asks the customer for permission. Looking at the Windows Media Player usage we found that only 10% sessions show that the customer consented. Even more surprising is that only 3% of sessions see the toolbar at all (you still need to minimize Media Player to see the controls). In other words, 97% of sessions aren't even enjoying this functionality at all! Since we do believe the scenario has value, we know to look into alternative designs. We'd like to surface this functionality to a larger set of customers while making sure the customer remains in control of her experience.



STATE	SESSION PERCENT
Toolbar enabled	10%
Toolbar enabled <i>and</i> visible	3%

**Fig. 6: How many people use the Windows Media toolbar?** *Enabled means user consented to the toolbar, visible means the toolbar actually appeared on the taskbar.*

## Evolving the Taskbar

Before the team even sat down to brainstorm ideas about improving the taskbar, we all took time to first respect the UI. The taskbar is almost 15 years old, everyone uses it, people are used to it and many consider it good enough. We also recognized that if we were to improve it, we could not afford to introduce usability failures where none existed. This automatically sets a very high bar. We proceeded carefully by first looking into areas for improvement.

Here's a small sample of some things we've learned from our data, heard from our customers and what we've observed ourselves. One of favorite ways of gaining verbatim comments in a lab setting where we can validate the instrumented data but also gain in-depth context via interviews and questionnaires. In engineering Windows 7 we have hundreds of hours of studies like these. Please remember this is just a glimpse of some feedback—this is not an exhaustive list nor is it implied that we will, or should, act upon all of these concepts.

- Please let me rearrange taskbar buttons! Pretty please?
- I sometimes accidentally click on the wrong taskbar button and get the wrong window.
- It would be great if the taskbar spanned multiple monitors so there's more room to show windows I want to switch to.
- There isn't always enough text on the taskbar to identify the window I'm looking for.
- There's too much text on the taskbar. (Yes, this is the exact opposite of the previous item—we've seen this quite a bit in the blog comments as well.)
- It may take several clicks to get to some programs or files that I use regularly.
- Icons of pinned files sometimes look too much alike—I wish I could tell them apart better.
- The bottom right side of my screen is too noisy sometimes. There are lots of little icons and balloons competing for my attention.
- How do I add/remove "X" from the taskbar?
- I would like Windows to tuck away its features cleverly and simplify its interface.

In the abstract, we can summarize this feedback with a few principles:

- Customers can switch windows with increased confidence and ease.
- Commonly used items and tasks should be at the customer's fingertips.
- Customers should always feel in control.
- The taskbar should have a cleaner look and feel.

We hope this post provides a little more insight into the taskbar as well as our process of collecting and reacting to customer feedback. Stay tuned for more details in the future.

- Chaitanya

# Follow-up: Starting, Launching, and Switching

Steven Sinofsky | [2008-09-29T03:00:00+00:00](#)

---

*Lots of discussion on the taskbar and associated user interface. Chaitanya said he thought it would be a good idea to summarize some of the feedback and thoughts. –Steven*

We'd like to follow up on some themes raised in comments and email. This post looks at some observations on consistent feedback expressed (though not universal) and also provides some more engineering / design context for some of the challenges expressed.

First it is worth just reinforcing a few points that came up that were consistently expressed:

- Many of you agree that the Notification Area needs to be more manageable and customizable.
- We received several comments about rearranging taskbar buttons. This speaks to the need for a predictable place where taskbar buttons appear as well as your desire for more control over the taskbar.
- There were comments that talked about Quick Launch being valuable, but that it could stand to be an even better launching surface (e.g. larger by default or more room).
- Thumbnails are valuable to many of you, but their size doesn't always help you find the window you are looking for. There is interest in a better identification method of windows that consistently provided the right amount of information.
- Better scaling of supported windows was discussed. This includes optimizing the taskbar for more windows and spanning multiple displays.

## Data

Several of you asked about the conclusions we are drawing from the data we collect and how we will proceed.

*@Computermensch writes "The problem with this "analysis" (show me the data) is that you're only managing current activities surrounding the taskbar. So with respect "to evolving the taskbar" you're only developing it within its current operational framework while developing or evolution of really should refer to developing the taskbars concept."*

*@Bluvg posts "What if the UI itself was a reason that people didn't run more than 6-9 windows? In other words, what if the UI has a window number upper bound of effectiveness? Prioritizing around that 6-9 scenario would be taking away the wrong conclusion from the data, if that were the case. The UI itself would be dictating the data, rather than being driven by user demand."*

As we've said in all our posts around the data we collect and how we use it, data do not translate directly into our features, but informs the decisions. Information we collect from instrumentation as well as from customer interviews merely provides us with real-world accuracy of how a product is currently used. The goal is not necessarily to *just* design for the status quo. However, we must recognize that if a new design emerges that does not satisfy the goals and behavior of our customers today, we risk resistance. This is not to say one should never innovate and change the game—just that to do so must be respectful of the ultimate goal of the customer. Offering a new solution to a problem is great; just make sure you're solving the right problem and that there is a path from where people are today to where you think the better solution resides. With that said, rest assured that our design process recognizes the need for the taskbar to scale more efficiently for larger sets of windows. This would allow those who possibly feel "trapped" in the 6-9 window case to more comfortably venture to additional windows, if they really require it. Also, the improvements we make to the 90% case should still hold benefits to the current outliers.

## Notification Area

With so much feedback, it is always valuable to recognize when customer comments converge. The original post called out the problems with the Notification Area and these issues were further emphasized with your thoughts.

@Jalf writes *“Having 20 icons and a balloon notification every 30th second taking up space at the taskbar where it's \*always\* taking up space is just not cool. By all means, the information should be there if I need it, but can't we just assume that if I don't actively look for the information, it's probably because I don't want it.*

Jalf's comment is particularly interesting because it speaks to both the pros and cons of notifications. They certainly can be valuable, but they can also very easily overwhelm the customer as many of you note. A careful balance therefore must be reached such that the customer is kept informed of information that is relevant while she continues to remain in control. Since relevant is relative, the need for control is fundamental. Rest assured we are aware of the issues and we are taking them very seriously.

## Multi-mon Support

It comes as no surprise that many of you wrote to discuss multi-monitor support for the taskbar. This is a popular request from our enthusiasts (and our own developers) and was called out as an area of investigation in the original post.

@Justaur is very direct with this comment: *“The lack of multi-monitor support is just about a crime. We've seen pictures of Bill Gate's office and his use of 3 monitors. Most developers have 2 monitors these days. Why was multi-monitor support for the taskbar missing? Once again, this is an example of the compartmentalization of the Windows team and the lack of a user orientation in defining and implementing features. The fact that this is even a "possible" and not an "of course we're going to..." shows that you folks STILL don't get it.”*

At least in this particular case we tend to think we “get it”, but we also tend to think that the design of a multi-mon taskbar is not as simple as it may seem. As with many features, there is more than one way to implement this one. For example, some might suggest a unique taskbar that exists on each display and others suggest a taskbar that spans multiple displays. Let's look at both of these approaches. While doing so also keep in mind the complexities of having monitors of different sizes, orientations, and alignments.

If one was to implement a taskbar for each display where each bar only contained windows for its respective portion of the desktop, some issues arise. Some customers will cite advantages of less mouse travel since there is always a bar at the bottom on their screen. However, such a design would now put the onus on the customer to track where windows are. Imagine looking for a browser window and instead of going to a single place, you now had to look across multiple taskbars to find the item you want. Worse yet, when you move a window from one display to another, you would have to know to look in a new place to find it. This might seem at odds with the request to rearrange taskbar buttons because customers want muscle memory of their buttons. It would be like having two remotes with dynamically different functionality for your TV. This is one of the reasons that almost every virtual desktop implementation keeps a consistent taskbar despite the desktop you are working on.

Another popular approach is a taskbar that spans multiple desktops. There are a few third-party tools that attempt to emulate this functionality for the Windows taskbar. The most obvious advantage of this approach (as well as the dual taskbar) is that there is more room offered for launching, switching and whispering. It is fairly obvious that those customers with multiple displays have more room to have more windows open simultaneously and hence, require even more room on their taskbar. Some of our advanced customers address this issue by increasing the height of the taskbar to reveal multiple rows. Others ask for a spanning taskbar. The key thing to recognize is that the problem is not necessarily that the taskbar doesn't span, but that more room is required to show more information about windows. So, it stands to reason that we should come up with the best solution to this problem, independent of how many displays the customer has.

We thought it would be good to just offer a brief discussion on the specifics of solving this design problem as it is one we have spent considerable time on. One of the approaches in general we are working to do more of, is to change things when we know it will be a substantial improvement

and not also introduce complexities that outweigh the benefits we are trying to achieve.

Once again, many thanks for your comments. We look forward to talking soon.

- Chaitanya

# User Interface: Managing Windows windows

Steven Sinofsky | [2008-10-01T03:00:00+00:00](#)

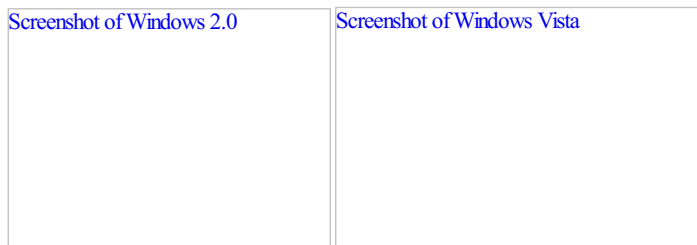
---

*We've booted the machine, displayed stuff on the screen, launched programs, so next up we're going to look at a pretty complex topic that sort of gets to the core role of the graphical user interface—managing windows. Dave Matthews is program manager on the core user experience team who will provide some of the data and insights that are going into engineering Windows 7. --Steven*

The namesake of the Windows product line is the simple “window” – the UI concept that keeps related pieces information and controls organized on screen. We'll use this post to share some of the background thinking and “pm philosophy” behind planning an update to this well established UI feature.

The basic idea of using windows to organize UI isn't new – it dates back (so I hear) to the first experiments with graphical user interfaces at Stanford over 40 years ago. It's still used after all this time because it's a useful way to present content, and people like having control over how their screen space is used. The “moveable windows” feature isn't absolutely needed in an operating system – most cell phones and media center type devices just show one page of UI at a time – but it's useful when multi-tasking or working with more than one app at a time. Windows 2.0 was the first Windows release that allowed moveable overlapping windows (in Window 1.0 they were only able to be tiled, not overlapping. This “tiled v. overlapping” debate had famous proponents on each side—on one side was Bill Gates and on the other side was [Charles Simonyi](#)). In addition, Windows also has the unique notion of “the multiple document interface” or MDI, which allows one frame window to itself organized multiple windows within it. This is somewhat of a precursor to the tabbed interfaces prevalent in web browsers.

As a side note, one of the earlier debates that accompanied the “tiled v. overlapping” *conversations* in the early Windows project was over having one menu bar at the top of the screen or a copy of the menu bar for each window (or document or application). Early on this was a big debate because there was such limited screen resolution (VGA, 640x480) that the redundancy of the menu bar was a real-estate problem. In today's large scale monitors this redundancy is more of an asset as getting to the UI elements with a mouse or just visually identifying elements requires much less movement. Go figure!

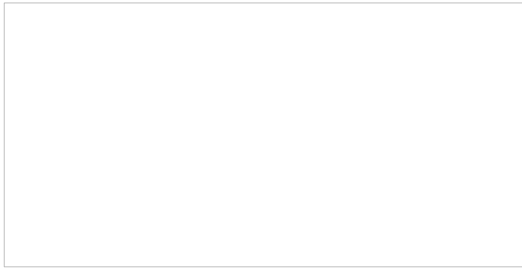


*From Windows 2.0 to Vista.*

An area I've been focusing on is in the “window management” part of the system – specifically the features involved in moving and arranging windows on screen (these are different than the window switching controls like the taskbar and alt-tab, but closely related). In general, people expect windows to be moveable, resizable, maximizable, minimizable, closeable; and expect them to be freely arranged and overlapping, with the currently used window sitting on top. These transformations and the supporting tools (caption buttons, resize bars, etc) make up the basic capabilities that let people arrange and organize their workspace to their liking.

In order to improve on a feature area like this we look closely at the current system - what have we got, and what works? This means looking at the way it's being used in the marketplace by ISVs, and the way it's used and understood by customers.

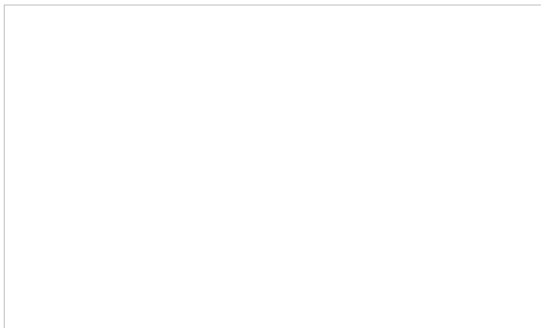




*Caption buttons give a simple way to minimize, maximize, and close. Resizable windows can be adjusted from any of their 4 edges.*

## **Data on Real-World Usage**

As pointed out in the previous Taskbar post, on average people will have up to 6–9 windows open during a session. But from looking at customer data, we find that most time is spent with only one or two windows actually visible on screen at any given time. It's common to switch around between the various open windows, but for the most part only a few are visible at once.

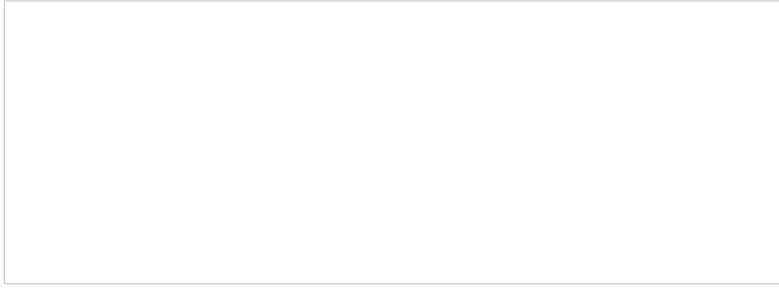


*Windows Feedback Panel data*

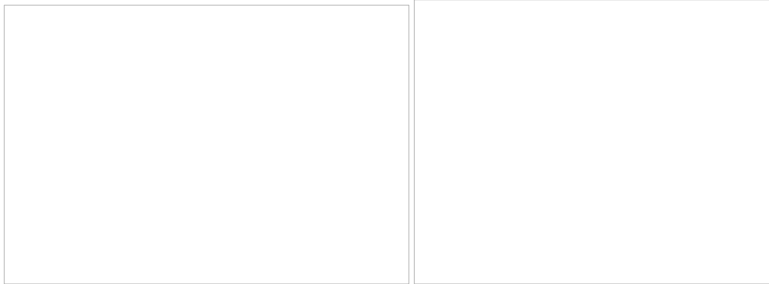
As part of our planning, we looked at how people spend their time and energy in moving and sizing their windows. This lets us understand what's working well in the current system, and what could be improved.

For example, we know that maximize is a widely used feature because it optimizes the work space for one window, while still being easy to switch to others. Users respond to that concept and understand it. Since most of the time users just focus on one window, this ends up being very commonly used. We know that for many applications people ask for every single pixel (for example spreadsheets where a few pixels gain a whole extra row of column) and thus the beyond maximize features for “full screen” become common, even for everyday productivity.

An issue we've heard (as recently as the comments on the taskbar post!) with maximize in Vista is that the customized glass color isn't very visible, because the windows and taskbar become dark when a window is maximized. (In Vista you can customize the glass window color – and in 29% of sessions a custom color has been set). The darker look was used to help make it clear that the window is in the special maximized state. This was important because if you don't notice that a window is maximized and then try to move it, nothing will happen - and that can be frustrating or confusing. For Windows 7 we're looking at a different approach so that the customized color can be shown even when a window is maximized.



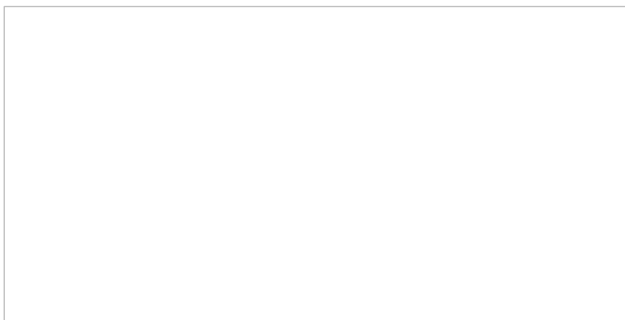
Interestingly, people don't always maximize their windows even when they're only using one window at a time. We believe one important reason is that it's often more comfortable to read a text document when the window is not too wide. The idea of maximizing is less useful on a wide monitor when it makes the sentences in an email run 20+ inches across the screen; 4 or 5 inches tends to be a more pleasant way to read text. This is important because large desktop monitors are becoming more common, and wide-aspect monitors are gaining popularity even on laptops. Since Windows doesn't have a maximize mode designed for reading like this, people end up manually resizing their windows to make them as tall as possible, but only somewhat wide. This is one of the areas where a common task like reading a document involves excessive fiddling with window sizes, because the system wasn't optimized for that scenario on current hardware.



*Resolution data suggests wide aspect-ratio monitors will become the norm.*

Being able to see two windows side by side is also a fairly common need. There are a variety of reasons why someone may need to do this – comparing documents, referring from one document into another, copying from one document or folder into another, etc. It takes a number of mouse movements to set up two windows side by side – positioning and adjusting the two windows until they are sized to roughly half the screen. We often see this with two applications, such as comparing a document in a word processor with the same document in a portable reader format.

Users with multiple monitors get a general increase in task efficiency because that setup is optimized for the case of using more than one window at once. For example, it's easy to maximize a window on each of the monitors in order to efficiently use the screen space. In a Microsoft Research [study](#) on multi-tasking, it was found that participants who had multiple monitors were able to switch windows more often by directly clicking on a window rather than using the taskbar, implying that the window they want to switch to was already visible. And interestingly, the total number of switches between windows was lower. In terms of task efficiency, the best click is an avoided click.



*MSR research [report](#)*

Single monitor machines are more common than multi-mon machines, but the window managing features aren't optimized for viewing multiple windows at once on one monitor. The taskbar does have context menu options for cascade, stack, or side-by-side, but we don't believe they're well understood or widely used, so most people end up manually resizing and moving their windows whenever they want to view two windows side by side.

An interesting multiple window scenario occurs when one of the windows is actually the desktop. The desktop is still commonly used as a storage folder for important or recent files, and we believe people fairly often need to drag and drop between the desktop and an explorer window, email, or document. The "Show Desktop" feature gives quick access to the desktop, but also hides the window you're trying to use. This means you either have to find and switch back to the original window, or avoid the Show Desktop feature and minimize everything manually. It's very interesting to see scenarios like this where the people end up spending a lot of time or effort managing windows in order to complete a simple task. This kind of experience comes across in our telemetry when we see complex sequences repeated. It takes further work to see if these are common errors or if people are trying to accomplish a multi-step task.

## Evolving the design

To find successful designs for the window management system, we explore a number of directions to see which will best help people be productive. From extremes of multi-tasking to focusing on a single item, we look for solutions that scale but that are still optimized for the most common usage. We look at existing approaches such as virtual desktops which can help when using a large number of different windows (especially when they are clustered into related sets), or docking palettes that help efficiently arrange space (as seen in advanced applications such as Visual Studio). And we look at novel solutions tailored to the scenarios we're trying to enable.

We also have to think about the variety of applications that the system needs to support. SDI apps (single document interface) rely heavily on the operating system to provide window management features, while MDI apps (multiple document interface) provide some of the window management controls for themselves (tabbed UI is an increasingly popular approach to MDI applications). And some applications provide their own window sizing and caption controls in order to get a custom appearance or behavior. Each of these approaches is valuable, and the different application styles need to be taken into account in making any changes to the system.

For Windows 7 our goal is to reduce the number of clicks and precise movements needed to perform common activities. Based on data and feedback we've gotten from customers, a number of scenarios have been called out as important considerations for the design. As with all the designs we're talking about—it is important to bring forward the common usage scenarios, make clear decisions on the most widely used usage patterns, address new and "unarticulated needs", and to also be sure to maintain our philosophy of "in control". Some of the scenarios that are rising to the top include:

- Can efficiently view two windows at once, with a minimal amount of set up.
- Simple to view a document at full height and a comfortable reading width.
- Quick and easy to view a window on the desktop.
- The most common actions should require the least effort - quicker to maximize or restore windows with minimal mouse precision required.
- Keyboard shortcuts to replace mouse motions whenever possible for advanced users.
- Useful, predictable, and efficient window options for a range of displays: from small laptops to 30" or larger screens; with single or multiple monitors.
- Easy to use different input methods: mouse, keyboard, trackpad, pen, or touch screens.
- Customized window glass color visible even when maximized.
- Overall - customers feel in control, and that the system makes it faster and easier to get things done.

This last point is important because the feeling of responsiveness and control is a key test for whether the design matches the way people really work. We put designs and mockups in the usability lab to watch how people respond, and once we see people smiling and succeeding easily at their task we know we are on the right track. The ultimate success in a design such as this is when it feels so natural that it becomes a muscle memory. This is when people can get the feeling that they've mastered a familiar tool, and that the computer is behaving as it should.

This is some of the background on how we think about window management and doing evolutionary design in a very basic piece of UI. We can't wait to hear feedback and reactions, especially once folks start getting their hands on Windows 7 builds.

- Dave

# Follow-up: Managing Windows windows

Steven Sinofsky | [2008-10-04T03:00:00+00:00](#)

---

*There's a lot of great discussion from the window arranging post. This really shows how important these details are to people. Being able to arrange how apps are shown on screen is key for productivity because it impacts almost every task. It's also very personal – people want to be in control of their work environment and have it set up the way that feels right.*

*One thing that should be clear is that it would not be possible for us to provide solutions to all the different ways people would like to work and all of the different tools and affordances people have suggested—I think everyone can see how overloaded we would be with options and UI absorbing all the suggestions! At first this might seem to be a bit of a bummer, but one thing we loved was hearing about all the tools and utilities you use (and you write!) to make a Windows PC **your PC**. Our goal is not to provide the solution to every conceivable way of potentially managing your desktop, but rather to provide an amazing way to manage your desktop along with customizations and personalizations plus a platform where people can develop tools that further enhance the desktop in unique and innovative ways. And as we have talked about, even that is a huge challenge as we cannot provide infinite customization and hooks—that really isn't technically possible. But with this approach Windows provides a high degree (but not infinite) flexibility, developers provide additional tools, computer makers can differentiate their PCs, and you can tune the UI to be highly personalized and productive for the way you want to work using a combination of those elements and your own preferences.*

*One other thing worth noting is that a lot of the comments referred to oft discussed elements in Windows, such as stealing the focus of windows, the registry, or managing the z-order of windows—a great source of history and witticisms about Windows APIs is from [Raymond Chen's blog](#). Raymond is a long-time developer on the Windows team and author of [Old New Thing, The: Practical Development Throughout the Evolution of Windows](#). This is also a good source to read where the boundaries are between what Windows does and what developers of applications can choose to be responsible for doing (and what they are capable of customizing).*

*With that intro, Dave wanted to follow up with some additional insights the team has taken away from the discussion. --Steven*

We saw several pieces of feedback popping up consistently throughout the comments. Paraphrasing the feedback (more details below), it sounds like there's strong sentiment on these points:

- The size of windows matters, but wasting time resizing windows is annoying.
- Just let me decide where the windows go – I know best where my windows belong.
- Dragging files around is cumbersome because the target window (or desktop) is often buried.
- Desire for better ways to peek at the running windows in order to find what we're trying to switch to.
- Want a predictable way to make the window fit the content (not necessarily maximized).
- Want to keep my personalized glass color, even when a window is maximized.

For each of these needs, there's a lot of great discussion around possible solutions – both features from other products, and totally novel approaches. It's clear from these comments that there's a desire for improvement, and that you've been thinking about this area long enough to have come up with some fairly detailed recommendations! Below are excerpts from some of the conversations ongoing in the comments.

## Put the windows where I want them

It's super interesting to see people discussing the existing features, and where they work or don't work.

For example, @d\_e is a fan of the existing tiling options in the taskbar:

Arranging windows in a split-window fashion is actually quite easy: While pressing CTRL select multiple windows in the taskbar. Then right-click them and select one of the tiling options...

But that approach doesn't quite meet the goal for @Xepol:

As for the window reorder buttons on the taskbar -> I've known they were there since Win95, but I never use them. They never do what I want. If they even get close to the right layout, its the wrong window order. Since I have to drag stuff around anyways, its just easier to get exactly what I want the first time.

@Aengel suggests taking the basic idea of tiled windows to the next level in order to make them really useful:

A very useful feature would be the ability to split the desktop into separate portions, especially on larger screens. For example, I might want to maximize my Messenger window to a small part on the right hand side of the desktop and still have the ability to maximize other windows into the remaining space. Non-maximized windows would be able to float across both (all) parts of the desktop.

It sounds like there's agreement that optimizing the screen space for more than one window would be super useful, if it would only let you stay in control of where windows ended up, and was easy and quick to use every day. The current tiling features in the taskbar give hints at how this could be valuable, but aren't quite fast and easy enough to be habit forming.

## Open at the right size

We saw a lot of comments on the "default size" of windows, and questions about how that's decided. Applications get to choose what size they open at, and generally use whichever size they were at the last time they were closed (or they can choose not to honor those settings). One of the cases that can trip people up is when IE opens a small window (websites will do this sometimes), because once you close it that will be the new "last size".

@magicaklick suggested a solution:

I wish I have one more caption button, FIXED SIZE. Actually it is a checkbox. When I check the box, it will save the window state for this application. After that, I can resize/move around. When I close window, it will not save the later changes.

@steven\_sinofsky offered this advanced user tip that you can use to start being more click-efficient right away:

@magicalclick I dislike when that one happens! Rather than add another button or space to click, I do the same thing in one click with a "power user" trick which is when you see the small window open don't close it until you first open up another copy of the application with the "normal" window size. Then close the small one and then the normal one.

Of course this is a pain and close to impossible for anyone to find, but likely a better solution than adding a fourth UI affordance on the title bar.

-steven

## Finding the right window

The word being used is "Expose":

@Joey\_j: Windows needs an Expose-like feature. I want to see all of my windows at once.

@Dan.F: one word - expose. copy it.

@GRiNSER : Expose has its own set of drawbacks: Like having 30 windows on a macbook pro 1400x1050 screen is really not that helpful. Though its way more helpful than Crap Flip 3D. Expose would be even more useful with keyboard window search...

Regardless of the name, there's a desire to visually find the window you're looking for. Something more random-access than the timeline approach of Alt-Tab or Flip-3d, and something that lets you pick the window visually from a set of thumbnails. This is very useful for switching when there are a lot of windows open – but some current approaches don't scale well and it is likely scaling will become an even more difficult problem as people run even more programs.

## Dragging files

There were several comments (and several different suggestions) on making it easier to drag between windows:

@Manicmarc: I would love to see something like Mac OS's Springloaded folders. Drag something over a folder and hover, it pops up, drag over to the next folder, drop it.

@Juan Antonio: It would be useful that when I'm dragging an object I could to open a list or thumbnail of the windows ( maybe a right- click )to select what window use to drop the object.

On this topic, I loved @Kosher's comment on the difference between being able to do something, and it feeling right.

The UI could be enhanced quite a bit to make it much easier to do things. It's not just about how easy it is but it's also about how smoothly the user transitions between common UI workflows and tasks. This is a bit like explaining the difference between a Ferrari and a Toyota to someone that has never driven a Ferrari though, so I don't know if it will ever happen.

In designing Windows 7, we've really been taking the spirit of this comment to heart. I can't wait to hear what car Windows 7 is compared to once it's available for a test drive.

- Dave



# User Account Control

Steven Sinofsky | [2008-10-08T15:00:00+00:00](#)

---

*We promised that this blog would provide a view of Engineering Windows 7 and that means that we would cover the full range of topics—from performance to user interface, technical and non-technical topics, and of course easy topics and controversial topics. This post is about User Account Control. Our author is Ben Fathi, vice president for core OS development. UAC is a feature that crosses many aspects of the Windows architecture—security, accounts, user interface, design, and so on—we had several other members of the team contribute to the post.*

*We continue to value the discussion that the posts seem to inspire—we are betting (not literally of course) that this post will bring out comments from even the most reserved of our readers. Let's keep the comments constructive and on-topic for this one.*

*FWIW, the blogs.msdn.com server employs some throttles on comments that aim to reduce spam. We don't control this and have all the "unmoderated" options checked. I can't publish the spam protection rules since that sort of defeats the purpose (and I don't know them). However, I apologize if your comment doesn't make it through. --Steven*

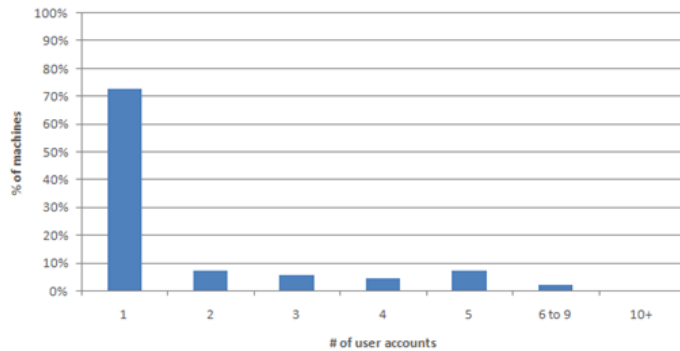
User Account Control (UAC) is, arguably, one of the most controversial features in Windows Vista. Why did Microsoft add all those popups to Windows? Does it actually improve security? Doesn't everyone just click "continue"? Has anyone in Redmond heard the feedback on users and reviewers? Has anyone seen a tv commercial about this feature?

In the course of working on Windows 7 we have taken a hard look at UAC – examining customer feedback, volumes of data, the software ecosystem, and Windows itself. Let's start by looking at why UAC came to be and our approach in Vista.

## The Why of UAC

Technical details aside, UAC is really about informing you before any "system-level" change is made to your computer, thus enabling you to be in control of your system. An "unwanted change" can be malicious, such as a virus turning off the firewall or a rootkit stealthily taking over the machine. However an "unwanted change" can also be actions from people who have limited privileges, such as a child trying to bypass Parental Controls on the family computer or an employee installing prohibited software on a work computer. Windows NT has always supported multiple user account types – one of which is the "standard user," which does not have the administrative privileges necessary to make changes like these. Enterprises can (and commonly do) supply most employees with a standard user account while providing a few IT pros administrative privileges. A standard user can't make system level changes, even accidentally, by going to a malicious website or installing the wrong program. Controlling the changes most people can make to the computer reduces help desk calls and the overall Total Cost of Ownership (TCO) to the company. At home, a parent can create a standard user account for the children and use Parental Controls to protect them.

However, outside the enterprise and the Parental Controls case, most machines (75%) have a single account with full admin privileges. This is partly due to the first user account defaulting to administrator, since an administrator on the machine is required, and partly due to the fact that people want and expect to be in control of their computer. Since most users have an Administrator account, this has historically created an environment where most applications, as well as some Windows components, always assumed they could make system-level changes to the system. Software written this way would not work for standard users, such as the enterprise user and parental control cases mentioned above. Additionally, giving every application full access to the computer left the door open for damaging changes to the system, either intentionally (by malware) or unintentionally (by poorly written software.)



**Figure 1.** Percentage of machines (server excluded) with one or more user accounts from January 2008 to June 2008.

User Account Control was implemented in Vista to address two key issues: one, incompatibility of software across user types and two, the lack of user knowledge of system-level changes. We expanded the account types by adding the Protected Admin (PA), which became the default type for the first account on the system. When a PA user logs into the system, she is given two security tokens – one identical to the Standard User token that is sufficient for most basic privileges and a second with full Administrator privileges. Standard users receive only the basic token, but can bring in an Administrator token from another account if needed.

When the system detects that the user wants to perform an operation which requires administrative privileges, the display is switched to “secure desktop” mode, and the user is presented with a prompt asking for approval. The reason the display is transitioned to “secure desktop” is to avoid malicious software attacks that attempt to get you to click yes to the UAC prompt by mimicking the UAC interface (spoofing the UI.) They are not able to do this when the desktop is in its “secure” state. Protected Admin users are thus informed of any system changes, and only need to click yes to approve the action. A standard user sees a similar dialog, but one that enables her to enter Administrative credentials (via password, smart card PIN, fingerprint, etc) from another account to bring in the Administrator privileges needed to complete the action. In the case of a home system utilizing Parental Controls, the parent would enter his or her login name and password to install the software, thus enabling the parent to be in control of software added to the system or changes made to the system. In the enterprise case, the IT administrator can control the prompts through group policy such that the standard user just gets a message informing her that she cannot change system state.

## What we have learned so far

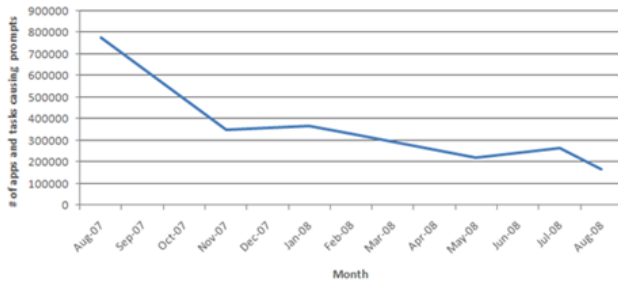
We are always trying to improve Windows, especially in the areas that affect our customers the most. This section will look at the data around the ecosystem, Windows, and end-users—recognizing that the data itself does not tell the story of annoyance or frustration that many reading this post might feel.

UAC has had a significant impact on the software ecosystem, Vista users, and Windows itself. As mentioned in previous posts, there are ways for our customers to voluntarily and anonymously send us data on how they use our features (Customer Experience Improvement Program, Windows Feedback Panel, user surveys, user in field testing, blog posts, and in house usability testing). The data and feedback we collect help inform and prioritize the decisions we make about our feature designs. From this data, we’ve learned a lot about UAC’s impact.

## Impact on the software ecosystem

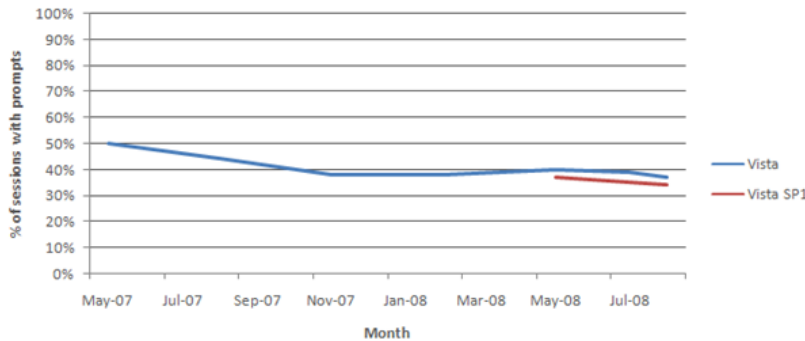
UAC has resulted in a radical reduction in the number of applications that unnecessarily require admin privileges, which is something we think improves the overall quality of software and reduces the risks inherent in software on a machine which requires full administrative access to the system.

In the first several months after Vista was available for use, people were experiencing a UAC prompt in 50% of their “sessions” - a session is everything that happens from logon to logoff or within 24 hours. Furthermore, there were 775,312 unique applications (note: this shows the volume of unique software that Windows supports!) producing prompts (note that installers and the application itself are not counted as the same program.) This seems large, and it is since much of the software ecosystem unnecessarily required admin privileges to run. As the ecosystem has updated their software, far fewer applications are requiring admin privileges. Customer Experience Improvement Program data from August 2008 indicates the number of applications and tasks generating a prompt has declined from 775,312 to 168,149.



**Figure 2.** Number of unique applications and tasks creating UAC prompts.

This reduction means more programs work well for Standard Users without prompting every time they run or accidentally changing an administrative or system setting. In addition, we also expect that as people use their machines longer they are installing new software or configuring Windows settings less frequently, which results in fewer prompts, or conversely when a machine is new that is when there is unusually high activity with respect to administrative needs. Customer Experience Improvement Program data indicates that the number of sessions with one or more UAC prompts has declined from 50% to 33% of sessions with Vista SP1.



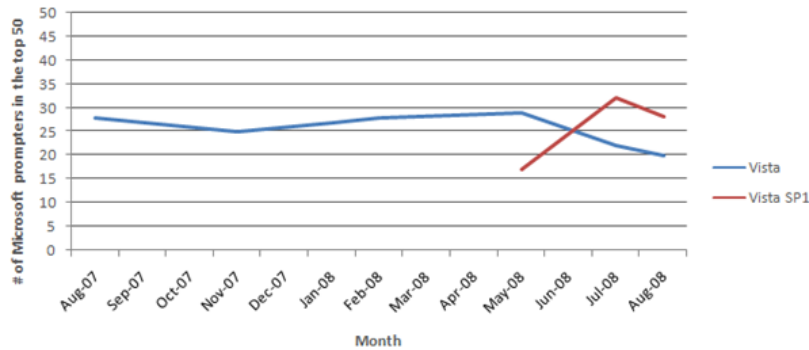
**Figure 3.** Percentage of sessions with prompts over time.

## Impact on Windows

An immediate result of UAC was the increase in engineering quality of Windows. There are now far fewer Windows components with full access to the system. Additionally, all the components that still need to access the full system must ask the user for permission to do so. We know from our data that Windows itself accounts for about 40% of all UAC prompts. This is even more dramatic when you look at the most frequent prompts: Windows components accounted for 17 of the top 50 UAC prompts in Vista and 29 of the top 50 in Vista SP1. Some targeted improvements in Vista SP1 reduced Windows prompts from frequently used components such as the copy engine, but clearly we have more we

can (and will) do. The ecosystem also worked hard to reduce their prompts, thus the number of Windows components on the top 50 list increased. Windows has more of an opportunity to make deeper architectural changes in Windows 7, so you can expect fewer prompts from Windows components. Reducing prompts in the software ecosystem and in Windows is a win-win proposition. It enables people to feel confident they have a greater choice of software that does not make potentially destabilizing changes to the system, and it enables people to more readily identify critical prompts, thus providing a more confident sense of control.

One important area of feedback we've heard a lot about is the number of prompts encountered during a download from Internet Explorer. This is a specific example of a more common situation - where an application's security dialogs overlap with User Account Control. Since XP Service Pack 2, IE has used a security dialog to warn users before running programs from the internet. In Vista, this often results in a double prompt - IE's security dialog, followed immediately by a UAC dialog. This is an area that should be properly addressed.



**Figure 4.** Number of Microsoft prompts in the top 50 over time.

## Impact on Customers

One extra click to do normal things like open the device manager, install software, or turn off your firewall is sometimes confusing and frustrating for our users. Here is a representative sample of the feedback we've received from the Windows Feedback Panel:

- "I do not like to be continuously asked if I want to do what I just told the computer to do."
- "I feel like I am asked by Vista to approve every little thing that I do on my PC and I find it very aggravating"
- "The constant asking for input to make any changes is annoying. But it is good that it makes kids ask me for password for stuff they are trying to change."
- "Please work on simplifying the User Account control.....highly perplexing and bothersome at times"

We understand adding an extra click can be annoying, especially for users who are highly knowledgeable about what is happening with their system (or for people just trying to get work done). However, for most users, the potential benefit is that UAC forces malware or poorly written software to show itself and get your approval before it can potentially harm the system.

Does this make the system more secure? If every user of Windows were an expert that understands the cause/effect of all operations, the UAC prompt would make perfect sense and nothing malicious would slip through. The reality is that some people don't read the prompts, and thus gain

no benefit from them (and are just annoyed). In Vista, some power users have chosen to disable UAC – a setting that is admittedly hard to find. We don't recommend you do this, but we understand you find value in the ability to turn UAC off. For the rest of you who try to figure out what is going on by reading the UAC prompt, there is the potential for a definite security benefit if you take the time to analyze each prompt and decide if it's something you want to happen. However, we haven't made things easy on you - the dialogs in Vista aren't easy to decipher and are often not memorable. In one lab study we conducted, only 13% of participants could provide specific details about why they were seeing a UAC dialog in Vista. Some didn't remember they had seen a dialog at all when asked about it. Additionally, we are seeing consumer administrators approving 89% of prompts in Vista and 91% in SP1. We are obviously concerned users are responding out of habit due to the large number of prompts rather than focusing on the critical prompts and making confident decisions. Many would say this is entirely predictable.

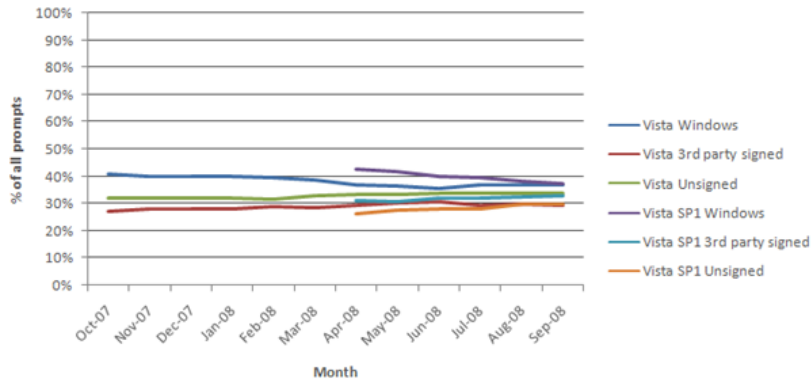


Figure 5. Percentage of prompts over time per prompt type.

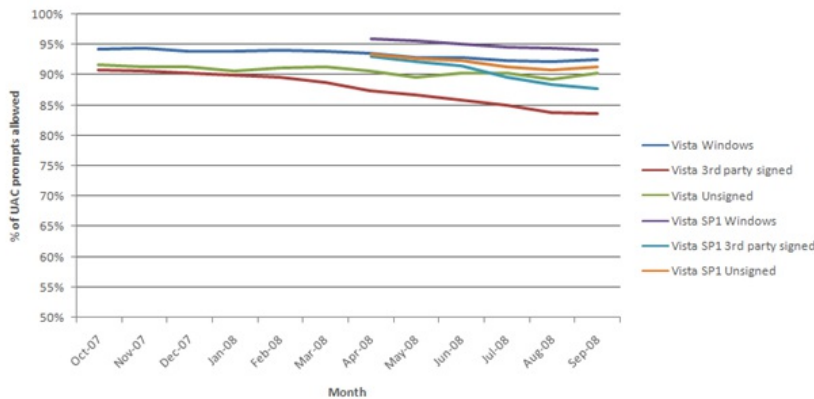


Figure 6. Percentage of UAC prompts allowed over time.

## Looking ahead...

Now that we have the data and feedback, we can look ahead at how UAC will evolve—we continue to feel the goal we have for UAC is a good one and so it is our job to find a solution that does not abandon this goal. UAC was created with the intention of putting you in control of your system, reducing cost of ownership over time, and improving the software ecosystem. What we've learned is that we only got part of the way there in Vista and some folks think we accomplished the opposite.

Based on what we've learned from our data and feedback we need to address several key issues in Windows 7:

- Reduce unnecessary or duplicated prompts in Windows and the ecosystem, such that critical prompts can be more easily identified.
- Enable our customers to be more confident that they are in control of their systems.
- Make prompts informative such that people can make more confident choices.
- Provide better and more obvious control over the mechanism.

The benefits UAC has provided to the ecosystem and Windows are clear; we need to continue that work. By successfully enabling *standard users* UAC has achieved its goal of giving IT administrators and parents greater control to lock down their systems for certain users. As shown in our data above, we've seen the number of external applications and Windows components that unnecessarily require Admin privileges dramatically drop. This also has the direct benefit of reducing the total amount of prompts users see, a common complaint we hear frequently. Moving forward we will look at the scenarios we think are most important for our users so we can ensure none of these scenarios include prompts that can be avoided. Additionally, we will look at "top prompters" and continue to engage with third-party software vendors and internal Microsoft teams to further reduce unnecessary prompts.

More importantly, as we evolve UAC for Windows 7 we will address the customer feedback and satisfaction issues with the prompts themselves. We've heard loud and clear that you are frustrated. You find the prompts too frequent, annoying, and confusing. We still want to provide you control over what changes can happen to your system, but we want to provide you a better overall experience. We believe this can be achieved by focusing on two key principles. 1) Broaden the control you have over the UAC notifications. We will continue to give you control over the changes made to your system, but in Windows 7, we will also provide options such that when you use the system as an administrator you can determine the range of notifications that you receive. 2) Provide additional and more relevant information in the user interface. We will improve the dialog UI so that you can better understand and make more informed choices. We've already run new design concepts based on this principle through our in-house usability testing and we've seen very positive results. 83% of participants could provide specific details about why they were seeing the dialog. Participants preferred the new concepts because they are "simple", "highlight verified publishers," "provide the file origin," and "ask a meaningful question."

In summary, yes, we've heard the responses to the UAC feature – both positive and negative. We plan to continue to build on the benefits UAC provides as an agent for standard user, making systems more secure. In doing so, we will also address the overwhelming feedback that the user experience must improve.

Ben Fathi

# Windows Desktop Search

Steven Sinofsky | [2008-10-13T03:00:00+00:00](#)

---

*One of the points of feedback has been about disabling services and optionally installing components—we've talked about our goals in this area in previous posts. A key driver around wanting this type of control (but not the only driver) is a perception around performance and resource consumption of various platform components. A goal of Windows is to provide a reliable and consistent platform for developers—one where they can count on system services as being available, as well as a set of OS features that all customers have the potential to benefit from. At the same time we must do so in a way that is efficient in system resource usage—efficient enough so the benefit outweighs the cost. We recognize that some percentage of customers believe solving this equation can only be done manually—much like some believe that the best car performance can only come from manual transmission. For this post we're going to look into the desktop search functionality from the perspective of the work we're doing as both a broadly available platform component and to provide the rich end-user functionality, and also look at the engineering tradeoffs involved and techniques we use to build a great solution for everyone. Chris McConnell, a principal SDE on the Find and Organize team, contributed this post. -Steven*

Are you one of those folks who believes that search indexing is the cause of your drive light flashing like mad? Do you believe this is the reason you're getting skooled when playing first person shooters with friends? If so, this blog post is for you! The Find and Organize team owns the 'Windows Search' service, which we simply refer to as the 'indexer'. A refrain that we hear from some Vista power-users is they want to disable the indexer because they believe it is eating up precious system resources on their PC, offering little in return. Per our telemetry data, at most about 1.5% of Vista users disable the indexing service, and we believe that this perception is one motivator for doing so.

The goal of this blog post is to clarify the role of the indexer and highlight some of the work that has been done to make sure the indexer uses system resources responsibly. Let's start by talking about the function of the indexing service – what is it for? why should you leave it running?

## Why Index?

Today's PCs are filled with many rich types of files, such as documents, photos, music, videos, and so on. The number of files people have on their PC is growing at a rapid pace, making it harder and harder for them to find what they're looking for, no matter how organized their files may (or may not) be. Increasingly, these files contain a good deal of structure, with metadata properties which describe their contents. A typical music file contains properties which describe the artist, album name, year of release, genre, duration of the song, and others which can be very useful when searching for music.

Although search indexing technologies date back to the early days of Windows, With Windows Vista Microsoft introduced a consumer operating system that brought this functionality to mainstream users more prominently. Prior to Vista, searching was pretty rudimentary – often a brute force crawl through the files on your machine, looking only at simple file properties such as file name, date modified, and size, or an application specific index of application specific data. Within Windows, a more comprehensive search option allowed you to also examine the contents of the files, but this wasn't widely used. It was fairly basic functionality – it treated all files just the same, without the tapping in to the rich metadata properties available in the files.

In Windows Vista, the indexing service is on by default and includes expanded support in terms of the number of file formats and properties which are indexed. The indexer watches specific folders on your PC and catalogues their contents to facilitate fast searching of those files. When Windows indexes your music files, it also knows how to extract the music-specific properties which you're most likely to search for. This enables support for more powerful searches and richer views over your files which wasn't possible before. But this indexing doesn't come free, and this is where engineering gets interesting. There's a non-zero cost (in terms of system resources) that has to be paid to enable this functionality, and there are trade-offs involved in when and how you pay that price. There is nothing unique to indexing—all features have this cost-benefit tradeoff.

## Trade-Offs

Many search solutions follow(ed) the traditional "grep" model which means every search will read **all** of the files you wanted to search. In this

case, you paid with your time as you waited for the search to execute. The more files you searched, the longer you waited each time you searched. If you wanted to perform the same search again, you would “pay” again. And the value you were getting in return wasn’t very good since the search functionality wasn’t particularly powerful. With Windows Vista, the indexer tries to read all of your files *before* you search so that when you search, it’s generally quicker and more responsive. This requires the indexer to scan all of your files just once initially, and not each and every time you perform a search. If the file were to change, the indexer would receive a notification (a “push” event) so that it could read that file again. When the indexer reads a file, it extracts the pertinent information about the file to enable more powerful searches and views. The challenge is to do this quickly enough so that the index is always up to date and ready for you to search, but also doing so in such a way that it doesn’t impact the performance of your system in a negative way. This is always a balancing act requiring trade-offs, and there are a number of things the indexer does to maintain its standing as a good Windows citizen while working to make sure that the index is always up-to-date.

## A Model Citizen

A lot of work has gone into making the indexer be a model Windows citizen. We’ve written an extensive [whitepaper](#) on the issue, but it’s worth covering some of the highlights here. First and foremost, the indexer only monitors certain folders, which limits the amount of work it needs to do to just those files that you’re most likely to search. The indexer also “backs off” when you are actively using your PC. It indexes files more slowly, or stops entirely depending on the level of activity on the PC. When the indexer is reading files it uses low priority I/O and CPU and immediately releases the file if another application needs access.

It’s critical that we get all of these issues right for the indexer, because it’s not only important for the features that our team builds (like Windows Search), but it’s important to the Windows platform as a whole. There are a host of applications which require the ability to search file contents on the PC. Imagine if each one of those applications built their own version of the indexer! Even if all of these applications did a great job, there will be a lot of unnecessary and redundant activity happening on your PC. Every time you saved one of your documents there will be a flurry of activity as these different indexers rushed to read the new version. To combat that, the indexer is designed to do this work for any application which might choose to use it and provide an open platform and API with flexibility and extensibility for developers. The API designed to be flexible enough to meet needs across the Windows ecosystem. Out of the box, the indexer has knowledge of about 200 common file types, cataloging nearly 400 different properties by default. And there is support for applications to add new file types and properties at any time. Applications can also add support for indexing of data types that aren’t file-based at all, like your e-mail. Just a few of the applications that are leveraging the indexer today are Microsoft Office [Outlook](#) and OneNote, Lotus Notes, Windows Live Photo Gallery, [Internet Explorer 8](#), and [Google Desktop Search](#). As with all extensible systems, developers often find creative uses for components for the system services. One example of this is the way the Tablet PC components leverage the index contents to improve handwriting accuracy.

## Constantly Improving

We’re constantly working to improve the indexer’s performance and reliability. Version 3 shipped in Windows Vista. Major improvements in this version included:

- The indexer runs as a system service vs. as a per user process. This minimizes impact on multi-user scenarios e.g. only one catalog per system results in reduction in catalog size and prevents re-indexing of the same content over and over. Additional benefit is gained from the robust nature of services.
- The indexer employs [low priority I/O](#) to minimize impact of indexing on responsiveness of PC. Before Windows Vista, all I/O was treated equally.

We’ve already released Windows Search version 4 as an enhancement to either Windows XP or Vista which goes even further in terms of performance and stability improvements, such as:

- Significant improvements across the board for queries which involve sorting, filtering or grouping. Example improvements on Vista include:
  1. Getting all results while sorting or grouping has been improved. Typical query improvements are up to 38% faster.



2. CPU time has been reduced by 80%
  3. Memory usage has been reduced by 20%
- Load on Exchange servers is reduced over 95% when Outlook is running in online mode. With previous versions of Windows Search, large numbers of Outlook clients running in online mode could easily overwhelm the Exchange server.
  - Reliability improvements including:
    1. We made a number of fixes to address user-reported situations that previously caused indexing to stop working.
    2. We improved the indexer's ability to both prevent and recover from index corruptions. Now, when catalog corruption is detected it is always rebuilt automatically – previously this only happened in certain cases.
    3. We added new logging and events to help track down and fix reliability issues.

And we've done even more to improve performance and reliability for the indexer in Windows 7 which you'll soon see at the PDC. If you still believe that the indexer is giving you trouble, we've got a few things for you to try:

- Download and install [Windows Search 4](#) (on Vista or XP).
- Download and install the [Indexer Gadget](#) from the Windows Live Gadget Gallery (Vista only). This gadget was written by one of our team members, and gives you a quick way to view the number of items indexed. It also allows you to pause indexing, or to make it run full-speed (without backing off).
- If you're one of those people who like to get under the hood of the car and poke around the engine, you can use the Windows Task manager and/or Resource Monitor to monitor the following processes: SearchIndexer, SearchFilterHost, SearchProtocolHost.

If you feel as though your system is slow, and you suspect the indexer is the culprit, watch the gadget as you work with your PC. Is the number of indexed items changing significantly when you're experiencing problems? If you pause the indexer, does your system recover? We're always looking to make our search experience better, so if you are still running into issues, we want to hear about them. Send your feedback to [idx-help@microsoft.com](mailto:idx-help@microsoft.com)

Chris McConnell

Find and Organize

# Engineering 7: A view from the bottom

Steven Sinofsky | [2008-10-15T03:00:00+00:00](#)

---

## Aka: A developers view of the Windows 7 Engineering process

*This post is by Larry Osterman. Larry is one of the most “experienced” developers on the Windows team and has been at Microsoft since the mid 1980’s. There are only three other folks who have worked at Microsoft longer on the entire Windows team! Personally, I remember knowing about Larry when I started at Microsoft back in 1989—I remember he worked on “multimedia” (back when we used to host the Microsoft CD-ROM Conference) and he was one of those people that stood up and received a “5 Year” award from Bill Gates at the first company meeting I went to—that seemed amazing back then! For Windows 7, Larry is a developer on the Devices and Media team which is where we work on audio, video, bluetooth, and all sorts of cool features for connecting up devices to Windows.*

*Larry wrote this post without any prodding and given his experience on so many Windows releases these thoughts seemed really worthwhile in terms of sharing with folks. This post goes into “how” we work as a team, which for anyone part of a software team might prove pretty interesting. While this is compared and contrasted with Vista, everyone knows that there is no perfect way to do things and this is just a little well-informed perspective.*

*So thank you Larry! --Steven*

Thanks to Steven and Jon for letting me borrow their soapbox :-).

I wanted to discuss my experiences working on *building* Windows 7 (as opposed to the other technical stuff that you’ve read on this blog so far), and to contrast that with my experiences building Windows Vista. Please note that these are MY experiences. Others will have had different experiences; hopefully they will also share their stories here.

The experience of building Windows 7 is dramatically different from the experience of building Vista. The rough outlines of the product development process haven’t changed, but organizationally, the Windows 7 process is dramatically better.

For Windows Vista, I was a part of the WAVE (Windows Audio Video Excellence) group. The group was led by a general manager who was ultimately responsible for the deliverables. There was a test lead, a development lead and a program management lead who reported to the general manager. The process of building a feature roughly worked like this: the lead program managers decided (based on criteria which aren’t relevant to the post) which features would be built for Windows and which program managers would be responsible for which feature. The development leads decided which developers on the team would be responsible for the feature. The program manager for the feature wrote a functional specification (which described the feature and how it should work) in conjunction with development. Note that the testers weren’t necessarily involved in this part of the process. The developer(s) responsible for the feature wrote the design specification (which described how the feature was going to be implemented). The testers associated with the feature then wrote a test plan which described how to test the feature. The program manager or the developer also wrote the threat model for the feature.

The developer then went off to code the feature, the PM spent their time making sure that the feature was on track, and when the developer was done, the tester started writing test cases.

Once the feature was coded and checked into the source tree, it moved its way up to the “winmain” branch. Aside: The Windows source code has been arranged into “branches” – the root is “winmain”, which is the code base that would ultimately become Windows Vista. Each developer works in what are called “feature branches”, which merge changes into “aggregation branches”, the aggregation branches move into winmain.

After the feature was coded, the testers tested, the developers fixed bugs and the program managers managed the program :-). As the product moved further along, it got harder and harder to get bug fixes checked into winmain (every bug fix carries with it a chance that the fix will introduce a regression, so the risk associated with each bug fix needs to be measured and the tolerance for risk decreases incrementally). The team responsible for managing this process met in the “ship room” where they made decisions every single day about which changes went into the product and which ones were left out. There could be a huge amount of acrimony associated with that – often times there were debates that lasted for hours as the various teams responsible for quality discussed the merits associated with a particular fix.

All-in-all, this wasn’t too different from the way that features have been developed at Microsoft for decades (and is basically consistent with what I was taught back in my software engineering class back in college).

For Windows 7, management decided to alter the engineering structure of the Windows organization, especially in the WEX [Windows Experience] division where I work. Instead of being fairly hierarchical, Steven has 3 direct reports, each representing a particular discipline: Development, Test and Program Management. Under each of the discipline leads, there are 6 development/test/program management managers, one for each of the major groups in WEX. Those 2nd level managers in turn have a half a dozen or so leads, each one with between 5 and 15 direct reports. This reporting structure has been somewhat controversial, but so far IMHO it’s been remarkably successful.

The other major change is the introduction of the concept of a “triad”. A “triad” is a collection of representatives from each of the disciplines – Dev, Test and PM. Essentially all work is now organized by triads. If there’s ever a need for a group to concentrate on a particular area, a triad is spun off to manage that process. That means that all three disciplines provide input into the process. Every level of management is represented by a triad – there’s a triad at the top of each of the major groups in WEX, each of the second level leads forms a triad, etc. So in my group (Devices and Media) there’s a triad at the top (known as DKCW for the initials of the various managers). Within the sound team (where I work), there’s another triad (known as SNN for the initials of the various leads). There are also triads for security, performance, appcompat, etc.

Similar to Windows Vista, the leads of all three disciplines get together and decide a set of features that go in each release. They then created “feature crews” to implement each of the features. Typically a feature crew consists of one or two developers, a program manager and one or two testers.

This is where one of the big differences between Vista and Windows 7 occurs: In Windows 7, the feature crew is responsible for the entire feature. The crew together works on the design, the program manager(s) then writes down the functional specification, the developer(s) write the design specification and the tester(s) write the test specification. The feature crew collaborates together on the threat model and other random documents. Unlike Windows Vista where senior management continually gave “input” to the feature crew, for Windows 7, management has pretty much kept their hands off of the development process. When the feature crew decided that it was ready to start coding (and had signed off on the 3 main documents), the feature crew met with the second level triad (in my case with DKCW) to sanity check the feature – this part of the process is critical because the second level triad gets an opportunity to provide detailed feedback to the feature crew about the viability of their plans.

And then the crew finally gets to start coding. Sort-of. There are still additional reviews that need to be done before the crew can be considered “ready”. For instance, the feature’s threat model needs to be reviewed by one of the members of the security triad. There are other parts of the document that need to be reviewed by other triads as well.

A feature is not permitted to be checked into the winmain branch until it is complete. And I do mean complete: the feature has to be capable of being shipped before it hits winmain – the UI has to be finished, the feature has to be fully functional, etc. In addition, when a feature team takes a dependency on another Windows 7 feature, the feature teams for the two features MUST sign a service level agreement to ensure that each team knows about the inter-dependencies. This SLA is especially critical because it ensures that teams know about their dependants – that way when they change the design or have to cut parts of the feature, the dependent teams aren’t surprised (they may be disappointed but they’re not surprised). It also helps to ensure tighter integration between the components – because one team knows the other team, they can ensure that both teams are more closely in alignment.

Back in the Vista day, it was not uncommon for feature development to be spread over multiple milestones – stuff was checked into the tree that really didn’t work completely. During Win7, the feature crews were forced to produce coherent features that were functionally complete – we were told to operate under the assumption that each milestone was the last milestone in the product and not schedule work to be done later on. That meant that teams had to focus on ensuring that their features could actually be implemented within the milestone as opposed to pushing them out.

For the nuts and bolts, The Windows 7 development process is scheduled over several 3-month long milestones. Each milestone allowed for 6 weeks of development and 6 weeks of integration – essentially time to fine-tune the feature and ensure that most of the interoperability problems were shaken out.

Ok, that's enough background (it's bad when over half a post on Windows 7 is actually about Windows Vista, but a baseline needed to be established). As I said at the beginning, this post is intended to describe my experiences as a developer on Windows 7. During Windows 7, I worked on three separate feature crews. The first crew delivered two features, the second crew delivered about 8 different features all relatively minor and the third crew delivered three major features and a couple of minor features. I also worked as the development part of the WEX Devices and Media security team (which is where my [series of post on Threat Modeling](#) came from – I wrote them while I was working with the members of D&M on threat modeling). And I worked as the development part of an end-to-end scenario triad that was charged with ensuring that scenarios that the Sound team defined at the start of the Windows 7 planning process were actually delivered in a coherent and discoverable way.

In addition, because the test team was brought into the planning process very early on, the test team provided valuable input and we were able to ensure that we built features that were not only code complete but also test complete by the end of the milestone (something that didn't always happen in Vista). And it ensured that the features we built were actually testable (it sounds stupid I know, but you'd be surprised at how hard it can be to test some features). As a concrete example, we realized during the planning process that some aspect of one of the features we were working on in M2 couldn't be completed during the milestone. So before the milestone was completed, we ripped the feature out (to be more accurate, we changed the system so that the new code was no longer being built as a part of the product). During the next milestone, after the test team had finished writing their tests, we re-enabled the feature. But we remained true to the design philosophy – at the end of the milestone everything that was checked into the "main" branch was complete – it was code AND test complete, so that even if we had to ship Windows 7 without M3 there was no test work that was not complete. This is a massive change from Vista – in Vista, since the code was complete we'd have simply checked in the code and let the test team deal with the fallout. By integrating the test teams into the planning process at the beginning we were able to ensure that we never put the test organization into that bind. This in turn helped to ensure that the development process never spiraled out of control. Please note that features *can and do* stretch across multiple milestones. In fact one of the features on the Sound team is scheduled to be delivered across three milestones – the feature crews involved in that feature carefully scheduled the work to ensure that they would have something worth delivering whenever Windows 7 development was complete.

Each of the feature crews I've worked on so far has had dramatically different focuses – some of the features I worked on were focused on core audio infrastructure, some were focused almost entirely on UX (user experience) changes, and some features involved much higher level components. Because each of the milestones was separate, I was able to work on a series of dramatically different pieces of the system, something I've really never had a chance to do before.

In Windows 7, senior management has been extremely supportive of the various development teams that have had to make the hard decisions to scale back features that were not going to be able to make the quality bar associated with a Windows release – and there absolutely are major features that have gone all the way through planning only to discover that there was too much work associated with the feature to complete it in the time available. In Vista it would have been much harder to convince senior management to abandon features. In Win7 senior management has stood behind the feature teams when they've had to make the tough decisions. One of the messages that management has consistently driven home to the teams is "cutting is shipping", and they're right. If a feature isn't coming together, it's usually far better to decide NOT to deliver a particular feature then to have that feature jeopardize the ability to ship the whole system. In a typical Windows release there are thousands of features and it would be a real shame if one or two of those features ended up delaying the entire system because they really weren't ready.

The process of building 7 has also been dramatically more transparent – even sitting at the bottom of the stack, I feel that I've got a good idea about how decisions are being made. And that increased transparency in turn means that as an individual contributor I'm able to make better decisions about scheduling. This transparency is actually a direct fallout of management's decision to let the various feature teams make their own decisions – by letting the feature teams deeper inside the planning process, the teams naturally make better decisions.

Of course that transparency works both ways. Not only were teams allowed to see more about what was happening in the planning process, but because management introduced standardized reporting mechanisms across the product, the leads at every level of the hierarchy were able to track progress against plan at a level that we've never had before. From an individual developer's standpoint, the overhead wasn't too onerous – basically once a week, you were asked to update your progress against plan on each of your work items. That status was then rolled up into a series of spreadsheets and web pages that allowed each manager to track all the teams' progress against plan. This allowed management to easily and quickly identify which teams were having issues and take appropriate action to ensure that the schedules were met (either by simplifying designs, assigning more developers, or whatever).

In general, it's been a total blast building 7. We've built some truly awesome features into the operating system and we've managed to keep the system remarkably stable during that entire process.

--Larry Osterman

# From Idea to Feature: A view from Design

Steven Sinofsky | [2008-10-18T03:00:00+00:00](#)

---

*Larry is very appreciative of the reception and comments his post received. Thank you! It is worth noting that we've surpassed over 2000 comments and I've received an equal amount of email. I am trying to reply as often as I can!*

*We're 10 days from the PDC and so we might take a short break from the blog while we practice our demos of Windows 7...we'll keep an eye on comments for sure and maybe a post or two on the way. ??*

Let's move "up" in the dev process and look at how we come up with what is in a release and how we think about taking a feature from an idea to feature.

*As we've posted on various engineering challenges we've often distilled the discussion down to a few decisions, often between two options (make a feature optional or not, add a window management feature one of two ways, etc.) Yet this doesn't quite get to the challenge of where does the product definition begin and how do we take an idea and turn it into a feature. Most choices in engineering Windows are not between two choices, but the myriad of considerations, variables, and possibilities we have before we even get to just a couple of options. This post looks a bit at the path from an idea to a feature.*

*A common thread we've seen in the feedback is to make "everything customizable and everything optional" (not a direct quote of course). Of course, by virtue of providing a platform we aim to offer the utmost in extensibility and customization by writing to the APIs we provide. There is an engineering reality that customization and extensibility have their cost—performance, complexity, and forward compatibility come to mind. One way to consider this is that if a feature has two "modes" (often enable the new feature or enable the old feature) in one release, then in a follow-up release if the feature is changed it potentially has four modes (old+old, old+new, new+old, new+new), and then down the road 8 modes, and so on. The complexity of providing a stable and consistent platform comes with the cost that we aren't always able to "hook" everything and do have to make practical choices about how a feature should work, in an effort to plan for the future. Designing a feature is also about making choices, tough choices. At the same time we also want to provide a great experience at the core operating system functions of launching programs, managing windows, working with files, and using a variety of peripherals—to name just a few things Windows does. This experience should be one that meets the needs of the broadest set of people across different skill levels and different uses of PCs, and also providing mechanisms to personalize with user interface and to customize with code. Every release we plan is a blending of fixing things that just don't work like we all had hoped and developing new solutions to both old and new problems, a blending of features and extensibility, and a blending of better support for existing hardware and support for new hardware.*

*This post is jointly written by Samuel Moreau the manager of the user experience design team for the Windows Experience, Brad Weed, Director of User Experience Design and Research for Windows and Windows Live, and Julie Larson-Green, the VP of Program Management for the Windows Experience. With the number of comments that describe a specific feature idea, we thought it would be good to give you an overview of how we approach the overall design process and how ideas such as the ones you mention flow into our process. Also for those of you attending the PDC, Sam will be leading a session on the [design principles of Windows 7](#). –Steven*

## Designing Windows – from idea to feature

In general, we follow a reasonably well-understood approach to product design, but that doesn't make it easy or "automatic". Often this is referred to as a "design funnel" where ideas go from concept to prototype to implementation and then refinement. By reading the various design ideas in the comments of Chaitanya's post on "[Starting, Launching and Switching](#)", you can see how difficult it can be to arrive at a refined feature design. In those comments you can find equally valid, yet somewhat opposite points of view. Additionally, you can also find comments that I would paraphrase as saying "do it all". It is the design process that allows us to work through the problem to get from idea to feature in the context of an overall product that is Windows.

From a product design perspective, the challenge of building Windows is the breadth of unique usage of just a single product. In a sense, one of

the magic elements of software is that it is “soft” and so you can provide all the functionality to all customers with little incremental cost and little difference in “raw materials” (many comments have consistently suggested we have everything available along with options to choose components in use and we have talked about minimizing the cost when components and features are not used even if they are available). And at the same time, there is a broad benefit to developers when they can know *a priori* that a given PC has a common set of functions and can take advantage of specific APIs that are known to be there and known to behave a specific way--the platform. This benefit of course accrues to individuals too as you can walk up to any PC and not only have a familiar user experience, but if you want to do your own work, use a specific device, or run a certain program on the PC you can also do that. This breadth of functionality is a key part of the value of a Windows PC. Yet it also poses a pretty good design challenge. Learning, understanding, and acting on the broad set of inputs into designing Windows is an incredibly fun and challenging part of building Windows.

As Larry pointed out the design and feature selection happens takes place in his part of the organization (not way up here!). There’s another discussion we will have in a future post about arriving at the themes of the overall release and how we develop the overall approach to a release so that the features fit together and form a coherent whole and we address customer needs in an *end-to-end* fashion.

We have a group of product designers that are responsible for the overall interaction design of Windows, the sequence and visualization of Windows. Program Managers across the team work with product designers as they write the specifications. Along with designers we have UX Researchers who own the testing and validation of the designs as we’ve talked about before. The key thing is that we apply a full range of skills to develop a feature while making sure that ownership is clear and end-to-end design is clear. The one thing we are not is a product where there is “one person” in charge of everything. Some might find that to be a source of potential problems and others might say that a product that serves so many people with such a breadth of features could not be represented by a single point of view (whether that is development, testing, design, marketing, etc.). We work to make sure engineers are in charge of engineering, that the product has a clear definition that we are all working towards implementing and that product definition represents the goals across all the disciplines it takes to deliver Windows to customers around the world. And most importantly, with Windows 7 we are making renewed effort at “end to end” design.

Let’s look at the major phases of product design in Engineering Windows. What we’ll talk about is of course generalized and doesn’t apply to each specific incident. One thing we always say internally is that we’re a learning organization—so no process is perfect or done and we are always looking to make it better as we move through each and every iteration of building Windows.

Throughout this post when we say “we” what this really means is the individuals of each discipline (dev, test, pm, design) working together—there’s no big feature or design committee.

## **Pick the question or get an idea**

We get an idea from somewhere of something to do—it could be big (build UX to support a new input method such as touch), wild (change the entire UI paradigm to use 3D), or an improvement / refinement of an existing feature (multi-monitor support), as some examples. There is no shortage of creative ideas, because frankly, that is the easy part. Ideas flow in from all corners of the ecosystem, ourselves included. We’ve talked a lot about comments and feedback from this blog and that is certainly one form of input. Product reviews, enterprise customers, customer support lines, PC makers, hardware and software developers, blogs, newsgroups, MVPs, and many others have similar input streams into the team.

The key is that working on Windows is really a constant stream of inputs. We start with a framework for the release that says what goals and scenarios we wish to make easier, better, faster. And with that program management builds up candidate ideas—that is ideas that will make their way through features. The job of getting a feature “baked” enough falls to program management and they do this work by working across the product and working with design, development, and testing (as Larry described).

With regard to where ideas come from, what we like to say is that the job of program management is not to have all the great ideas but to make sure all the great ideas are ultimately picked. The best program managers make sure the best ideas get done, no matter where they come from.

## **Gather information and data**

Given any idea, the first step is to understand what data we have “surrounding” the idea. Sometimes the idea itself comes to us in a data-centric manner (customer support incidents) or other times it is anecdotal (a blog).

The first place we look is to see what data do we have based on real world usage that would support the development of a hypothesis, refute or support the conventional wisdom, or just shed some light on the problem. The point is that the feature starts its journey by adding more perspectives to the input.

Essentially, we need an objective view that illuminates the hypothesis. We gather this data from multiple sources including end users, customers, partners, and in various forms such as instrumentation, research, usability studies, competitive products, direct customer feedback, and product support.

As many (including us) have pointed out, telemetry data has limitations. First, it can never tell you what a person might have been trying to do—it only tells you what they did. Through usability, research, and observation, we are able to get more at the intent. For example, the way we talked about high dpi and how the telemetry showed one thing but the intent was different (and the impact of those choices was unintended). The best way to see this is to remember that a person using a PC is not interesting in “learning to use a PC” but is trying to get their own work done (or their own playtime). And when faced with a “problem” the only solutions available are the buttons and menu commands right there—the full solution set is the existing software. Our job is to get to the root of the problem and then either expand the solution set or just make the problem go away altogether.

What about *unarticulated* needs? The data plus intent shows the “known world” and “known solution space”, but one role we have is to be forward thinking and consider needs or desires that are not clearly articulated by those who do not have the full time job to consider all the potential solution spaces. The solution space could potentially be much broader than readily apparent from the existing and running product—it might involve a rearchitecture, new hardware, or an invention of a new user interface.

A great example of this was mentioned in one of the comments on the taskbar post. The comment (paraphrasing) indicated that the order of icons on the taskbar matters so sometimes he/she would simply close all the open programs and then restart them just so the programs were in the preferred order on the taskbar. Here the data would look like an odd sequence of launch/exit/launch/exit/launch/launch. And only through other means would we learn why someone would be doing that, and for the most part if you just walked up without any context and said “how can we make Windows easier” it isn’t likely this would bubble up to the top of the list of “requests”. Thus we see a number of neat things in this one example—we see how the data would not show the intent, the “request” would not be at the top of any list, and the solution might take any number of forms, and yet if solved correctly could be a pretty useful feature. Above all, in *hindsight* this is one of those “problems” that seems extraordinarily obvious to solve (and I am sure many of you are saying—“you should have just asked me!”) So we also learn the lesson that no matter what data and information we gather or what design we’re talking about, someone always noticed or suggested it :-).

## Hypothesize

The next step is where we propose a clear hypothesis – “people would benefit from rearranging icons on the taskbar because positional memory across different sessions will reduce the time to switch applications and provide a stronger sense of control and mastery of Windows”.

What is our hypothesis (in a scientific sort of way) as to what opportunity exists or what problem we would solve, and what the solution would look like, or why does the problem exist? Part of designing the feature is to think through the problem—why does it exist—and then propose the benefit that would come from solving the problem. It is important that we have a view of the benefit in the context of the proposed solution. It is always easy to motivate a change because it feels better or because something is broken so a new thing has to be better, but it is very important that we have a strong motivation for *why* something will benefit customers.

Another key part about the hypothesis is to understand the conventional wisdom around this area, especially as it relates to the target customer segment (end-user, enthusiast, PC maker, etc.) The conventional wisdom covers both the understanding of how/why a feature is a specific way today and also if there is a community view of how something should be solved. There are many historic examples where the conventional wisdom



was very strong and that was something that had to be considered in the design or something that had to be considered knowing the design was not going to take this into account—a famous example is the role of keyboard shortcuts in menus that the “DOS” world felt would be required (because not every PC had a mouse) but on the Mac were “unnecessary” because there was always a mouse. Conventional wisdom in the DOS world was that a mouse was optional.

## Experiment

For any hypothesis, there are numerous design alternatives. It is at this stage where we cast a broad net to explore various options. We sketch, write scenarios, story board, do wireframes and generate prototypes in varying fidelity. Along the way we are working to identify not just the “best answer” but to tease out the heart and soul of the problem and use the divergent perspectives to feed into the next step of validation.

This is a really fun part of the design process. If you walk our hallways you might see all sorts of alternatives in posters on the walls, or you might catch a program manager or designer with a variety of functional prototypes (PowerPoint is a great UI design tool for scenarios and click-thrus that balance time to create with fidelity, and Visio is pretty cool for this as well) or our designers often mock up very realistic prototypes we can thoroughly test in the lab.

## Interpret and Validate

With a pile of options in front of us we then take the next step of interpreting our own opinions, usability test data and external (to the team) feedback. This is the area where we end up in conversations that, as an example, could go something like this... “Option ‘A’ is better at elevating the discovery of a new feature, but option ‘B’ has a stronger sense of integration into the overall user experience”.

As we all know, at the micro level you can often find a perfect solution to a specific problem. But when you consider the macro level you start to see the pros and cons of any given solution. It is why we have to be very careful not to fall into the trap of a “tests”. The trap here is that it is not often possible to test a feature within the full context of usage, but only within the context of a specific set of scenarios. You can’t test how a feature relates to all potential scenarios or usages while also getting rich feedback on intent. This is why designing tests and interpreting the results is such a key part of the overall UX effort led by our researchers.

A mathematic way of looking at this is the “local min” versus a “global min”. A local min is one you find if you happen to start optimizing at the wrong spot on the curve. A good example of this in software is when faced with a usability challenge you develop a new control or new UI widget. It seems perfectly rational and often will test very well, especially if the task asked of the subject is to wiggle the widget appropriately. However, what we’re after is a global optimization where one can see that the potential costs (in code, quality, and usability) of another widget by erase any potential benefits gained by introducing a new widget. Much has been written about the role of decision theory as it relates to choosing between options, but our challenge with design is the preponderance of qualitative elements.

## Choosing

Ultimately we must pick a design and that choice is informed by the full spectrum of data, qualitative and quantitative.

Given a choice for a design and an understanding of how to implement it and the cost, there is still one more choice—should we do the feature at all. It sounds strange that we would go through all this work and then still maybe not build a specific feature. But like a movie director that shoots a scene that ends up on the cutting room floor, sometimes the design didn’t pan out as we had hoped, sometimes we were not able to develop an implementation plan within reason, or sometimes there were other ideas that just seemed better. And this is all before we get to the implementation, which as Larry pointed out has challenges as well.

We have two tools we use to assist us in prioritizing features and designs. First is the product plan—the plan says at a high level what we “require” the product to achieve in terms of scenarios, business goals, schedule, and so on. Most of the time features don’t make it all the way through prototyping and testing because they just aren’t going to be consistent with the overall goals of the release. These goals are important otherwise a product doesn’t “hang together” and runs the risk of feeling like a “bunch of features”. These high level goals inform us quite a bit in terms of what code we touch and what scenarios we consider for a release.

And second we have the “principles of design” for the release. These principles represent the language or vocabulary we use. These represent the core values—we often think of the design principles as anthropomorphizing the product—“if Windows were a person then it would be...”. This is the topic of Sam’s talk at the PDC.

As mentioned in the introduction, it isn’t possible to do everything twice. We do have to decide. This could be a whole series of posts—customization, compatibility modes, and so on. We definitely hear folks on these topics and always do tons of work to enable both “tweaking” and “staying put” and at the same time we need to balance these goals with the goals of providing a robust and performant platform and also moving the OS forward. Some of us were involved in Office 2007 and there is a fun [case study done by Harvard Business School](#) [note fee associated with retrieving the full text] about the decision to (or not to) provide a “compatibility mode” for Office 2007. This was a choice that was difficult at the time and a few folks have even mentioned it in some comments.

## **Implement and Integrate**

Finally, we build and iterate to refine the chosen solution. Inevitably there are new discoveries in the implementation phase and we adjust accordingly. As we integrate the solution into its place in Windows, that discovery continues. The beta period is a good example of how we continue to expand and learn from usage and feedback. In a Windows beta we are particularly interested in compatibility and real-world performance as those are two aspects of the design that are difficult to validate without the breadth of usage we can get if we do a great beta.

It is important to keep in mind that we follow intensely all the feedback we receive from all forms—reviews, blogs, and of course all the telemetry about how the product is used (realizing that the beta is a select group of people).

One of the things we hope to do with the blog, as you might have seen on the IE 8 Blog, is to discuss the evolution of the product in real-time. We’re getting close to this transition and are looking forward to talking more about the design choices we made!

-- Sam, Brad, and Julie

# Follow-up: Windows Desktop Search

Steven Sinofsky | [2008-10-23T03:00:00+00:00](#)

---

*The discussion and email about desktop search offered an opportunity for us to have a deeper architectural discussion about engineering Windows 7. There were a number of comments suggesting alternate implementation methods so we thought we'd discuss another approach and the various pros and cons associated with it. It offers a good example of the engineering balance we are striving for with Windows 7. Chris McConnell wrote this follow-up. --Steven (See you at the PDC in a week!)*

Thanks for all the great feedback on our first blog post on [Windows Desktop Search](#). I've summarized a number of points that have been made and added some comments about the architectural choices we have made and why.

## Integration with the File System

As some posters have pointed out, one possible implementation is to integrate indexing with the file system so that updating a file immediately updates the indices. Windows Desktop Search takes a different approach. There are two aspects of file system integration: knowing when a file changes and actually updating the indices before a file is considered "closed" and available. On an NTFS file system, the indexer is notified whenever a file changes. The indexer never scans the NTFS file system except during the initial index. It is on the second point—updating the indices immediately when a file is closed that we made a different choice. Updating immediately has the benefit that a file is not available until it is indexed, but it also comes with a number of potential disadvantages. We chose to decouple indexing from file system operations because it allows for more flexibility while still being almost real-time. Here are some of the benefits we see in the approach we took:

1. **Fewer resources are used.** Inverted indices are global. An inverted index maps from a word found in a property to a list of every document that contains that word. Indexing a single file requires updating an index for every single unique word found in the file. A single document might then update a very large number of individual indices. Making these changes and committing them with the same robustness found on individual files would be very expensive. The design of the indexer allows scheduling and aggregating these changes so that much less work is done overall—that means less CPU and less disk I/O. The system can be more robust because indexing doesn't only happen when a file is closed—and it can even be retried if necessary.
2. **File system operations are prioritized over indexing.** Getting files robustly updated and available is necessary for applications to use them. We don't want to delay that availability by forcing the cost of indexing into file close operations. Searching over files is important, but is less important than actually working with files. We wouldn't want applications to decide individually if the indexer should be turned on or off just because they were seeking the best performance with respect to the file system.
3. **There are lots of file types.** Microsoft supplies extractors (IFilter/IPropertyHandler) for many common file types as part of Windows. There are many other file types as well so it is important to allow non-Microsoft developers to write their own extractors. In Vista (and Windows 7), these extractors run in a locked down process that ensures that they are secure and do not affect the performance of the whole system. If indexing had to happen before a file was available, then an extractor could impact (intentionally or not) all file system operations.
4. **Some files are more valuable to index than others.** If indexing happened when a file is closed, then there is no control over the order files are indexed. Decoupling allows prioritizing indexing some files over others. For example, searching for music is much more likely than searching for binary files. If both music files and binary files have changed, then the indexer ensures it indexes the music files first. Some files are not worth indexing at all for most people. Several comments suggested that we should index the whole drive. We can do that—and for those who would find it valuable it easy to add folders to be indexed. (You can also remove them, but that is much less common so that is controlled through the control panel "Indexing Options.") For most people indexing system files is just a cost—they would never search for them and would be confused if they showed up as the result of a search.
5. **Not everything is a file in single file system.** Windows is all about supporting diversity. There are many different file systems like FAT32 and CDFS and we would like to be able to search over those as well. If we integrated with only NTFS, then we would have to still have a loosely coupled system for other file systems. Many applications also have databases optimized for their own needs. For example, Outlook has a database of email. If only files were indexed, then the email in the database could not be indexed unless Outlook either compromised their experience by using files only, or complicated their implementation by duplicating everything in both the file system and the database.

## Advanced Queries

A number of people expressed frustration with the lack of an advanced query UI. Microsoft has many advanced query user-interfaces in many products, but these are generally focused on well-defined query languages (SQL) or on specific domains (like the Advanced Find in Outlook). With Vista we wanted to address the query problem in a manner more familiar to people today—a single edit control. Our implementation supports a rich query language within that edit control. This is the same approach people are familiar with for web searching for both standard and advanced queries.

We had two observations that led to this approach:

1. The most important part of a search are the search terms. Usually a single term is enough (and as we know from web searching, the majority of searches are one or two words). And for refinement the file system tools of thumbnails, sorting, and/or type ahead can be used to narrow the search.
2. It is reasonable to consider a design for an advanced query UI covering property based search, but it will generally be unwieldy for all but the bravest people. As we mentioned, Windows Search covers over 300 properties by default so if you show every property then the UI is unusable. If we only show the most commonly used properties then how do you handle all of the other properties? Would properties be grouped by the common application or by attributes such as times, names, file attributes, etc.? Some of you might value the Outlook Advanced Find... interface, but there you see some of the challenges and that is within a specific domain where the grouping or related properties probably can be understood.

In designing Vista we incorporated the feedback that it is desirable to do precise queries. The approach taken in Vista was to support a rich [query language](#) which allows all properties and a fairly natural syntax. For example typing “from:gerald sent:today” will find all email from “Gerald” sent today! The big issue is that people do not know or the query language. In Windows 7, we have focused on helping people see how to use the query language in context. For now, you can see the following for some information on Vista’s [query syntax](#). Much of this syntax and experience is similar to web search that we all use today.

A number of comments were about substring matches in filenames, which we do not currently support. This is part of the overall discussion about advanced queries. In order to efficiently execute queries, the indexer builds indices that are based on individual words. In Vista we introduced “searching as you type” to our search UI. Under the hood this is implemented as prefix matches on the indexed words. So when you type, ‘foo’, we look for all terms that start with those letters including ‘food’ and ‘football’. Even more interesting if you type ‘foo net’ we will match on items that have the words ‘food’ and ‘network’ in them. (If what you really want is to match the phrase “foo net” then typing those words inside quotes will do that—another example of advanced query syntax) We have focused primarily on searching for terms found in any property, but there is no question that filenames are special. In recognition of that we support suffix queries on filenames. If you type ‘\*food’ then we will return files that end in ‘food’ like “GoodFood”. We do this by reversing the filename and then indexing it as a word. For example, the reverse filename of “GoodFood” would be “DooFdooG” which we index as a word. The suffix query ‘\*food’ is transformed into a prefix query “dooF\*” over the reverse filename index—clever, no? So we support prefix matches for all properties and suffix matches for filenames, but we do not support substring matches.

## Performance and Citizenship

A number of comments focused on improving performance and citizenship—and we definitely agree on this input. We are always striving to make Windows do more with fewer resources. For those who have turned off indexing all together we hope that our continued improvements will make you reconsider. Even if you organize all of your files and don’t find search useful for files, perhaps you will find start menu search, email search or Internet Explorer 8 address bar search useful. We have worked hard at improving performance and citizenship across Windows. Some of this progress is visible in WS4 and soon in Windows 7. We have improved along all of our dimensions including indexing cost, battery life, citizenship, query speed and scrolling speed. We have some tremendous tools that help us track down performance problems. If you want to help, please contact [idx-help@microsoft.com](mailto:idx-help@microsoft.com) and we will tell you how to collect performance traces we can analyze so that we can continue to make improvements.

Chris McConnell

Find and Organize

# Back from the PDC...next up, WinHEC

Steven Sinofsky | [2008-11-01T03:00:00+00:00](#)

---

This has been an amazingly special week for the Windows 7 team. We're all incredibly appreciative of the reception of Windows 7 this week at the PDC. Thank you!

All of us on the team have been closely watching the news reports and blogs of those who have been "kicking the tires" of the Windows 7 pre-beta. The reception has been fantastic and we're humbled by the excitement and enthusiasm for the release. We know we have a ton of work ahead of us to get to beta and then the path to RTM, and the reception has definitely given us an extra special motivation (though we were already pretty motivated).

Next week is our conference dedicated to the hardware partners in the ecosystem we have talked about. Called [WinHEC](#) (Windows Hardware Engineering Conference), we'll have another series of sessions and keynotes. Jon DeVaan will be taking the lead as we dive into the details of "fundamentals" and the work we are doing with some of the many partners involved in Windows 7. WinHEC also has a strong focus on Windows Server 2008 R2 (the server built off the Windows 7 kernel). These sessions will all be available online as well.

So with all the shows we're taking a short break from the blog as the folks that do the presenting are also the writers (myself included).

Below is a list of all the sessions on Windows 7 from the PDC. Please take some time to have a look as the information is very detailed for sure. How about using the comments on this post to ask questions of the sessions that you'd like to see more details on down the road? That would be really helpful for us to target our posts.

Many of you have written asking about the beta and how to sign up or download it. The best source for information on that will be the site <http://www.microsoft.com/windows/windows-7> which our product marketing team owns and will keep up to date as the beta information is available. Also note that the Windows Vista blog which is where you will see announcements / news has been updated to reflect the inclusion of Windows 7. This blog is now known as the [Windows Blog](#).

One of the very fun moments for me at the PDC was an "[Open Space](#)" session on the floor of the "Big Room" which was an open-microphone discussion. Channel9 captured this and might be a fun watch. See <https://channel9.msdn.com/posts/Charles/Steven-Sinofsky-at-the-PDC2008-Open-Space/>

For those of you interested in the Windows 7 APIs and what's new for developers there is an overview document that you might find valuable. See [Windows 7 Developer Guide](#) on MSDN.

Thank you very much for all the emails you have sent. I always share them with the team and really appreciate it.

**Presentation**

KYN02 Day Two #1 - Ray Ozzie, Steven Sinofsky, Scott Guthrie and David Treadwell (Windows 7 starts +17:00 minutes)

PC01 Windows 7: Web Services in Native Code

PC02 Windows 7: Extending Battery Life with Energy Efficient Applications

PC03 Windows 7: Developing Multi-touch Applications

PC04 Windows 7: Writing Your Application to Shine on Modern Graphics Hardware

**URL**

<https://channel9.msdn.com/pdc2008/KYN02/>

<https://channel9.msdn.com/pdc2008/PC01/>

<https://channel9.msdn.com/pdc2008/PC02/>

<https://channel9.msdn.com/pdc2008/PC03/>

<https://channel9.msdn.com/pdc2008/PC04/>

PC13 Windows 7: Building Great Audio Communications Applications	<a href="https://channel9.msdn.com/pdc2008/PC13/">https://channel9.msdn.com/pdc2008/PC13/</a>
PC14 Windows 7 Scenic Ribbon: The next generation user experience for presenting commands in Win32 applications.	<a href="https://channel9.msdn.com/pdc2008/PC14/">https://channel9.msdn.com/pdc2008/PC14/</a>
PC15 Windows 7: Benefiting from Documents and Printing Convergence	<a href="https://channel9.msdn.com/pdc2008/PC15/">https://channel9.msdn.com/pdc2008/PC15/</a>
PC16 Windows 7: Empower users to find, visualize and organize their data with Libraries and the Explorer	<a href="https://channel9.msdn.com/pdc2008/PC16/">https://channel9.msdn.com/pdc2008/PC16/</a>
PC18 Windows 7: Introducing Direct2D and DirectWrite	<a href="https://channel9.msdn.com/pdc2008/PC18/">https://channel9.msdn.com/pdc2008/PC18/</a>
PC19 Windows 7: Designing Efficient Background Processes	<a href="https://channel9.msdn.com/pdc2008/PC19/">https://channel9.msdn.com/pdc2008/PC19/</a>
PC22 Windows 7: Design Principles for Windows 7	<a href="https://channel9.msdn.com/pdc2008/PC22/">https://channel9.msdn.com/pdc2008/PC22/</a>
PC23 Windows 7: Integrate with the Windows 7 Desktop	<a href="https://channel9.msdn.com/pdc2008/PC23/">https://channel9.msdn.com/pdc2008/PC23/</a>
PC24 Windows 7: Welcome to the Windows 7 Desktop	<a href="https://channel9.msdn.com/pdc2008/PC24/">https://channel9.msdn.com/pdc2008/PC24/</a>
PC25 Windows 7: The Sensor and Location Platform: Building Context-Aware Applications	<a href="https://channel9.msdn.com/pdc2008/PC25/">https://channel9.msdn.com/pdc2008/PC25/</a>
PC42 Windows 7: Deploying Your Application with Windows Installer (MSI) and ClickOnce	<a href="https://channel9.msdn.com/pdc2008/PC42/">https://channel9.msdn.com/pdc2008/PC42/</a>
PC43 Deep Dive: What's New with user32 and comctl32 in Win32	<a href="https://channel9.msdn.com/pdc2008/PC43/">https://channel9.msdn.com/pdc2008/PC43/</a>
PC44 Windows 7: Programming Sync Providers That Work Great with Windows	<a href="https://channel9.msdn.com/pdc2008/PC44/">https://channel9.msdn.com/pdc2008/PC44/</a>
PC50 Windows 7: Using Instrumentation and Diagnostics to Develop High Quality Software	<a href="https://channel9.msdn.com/pdc2008/PC50/">https://channel9.msdn.com/pdc2008/PC50/</a>
PC51 Windows 7: Best Practices for Developing for Windows Standard User	<a href="https://channel9.msdn.com/pdc2008/PC51/">https://channel9.msdn.com/pdc2008/PC51/</a>
PC52 Windows 7: Writing World-Ready Applications	<a href="https://channel9.msdn.com/pdc2008/PC52/">https://channel9.msdn.com/pdc2008/PC52/</a>
ES20 Developing Applications for More Than 64 Logical Processors in Windows Server 2008 R2	<a href="https://channel9.msdn.com/pdc2008/ES20/">https://channel9.msdn.com/pdc2008/ES20/</a>

See you on this blog soon enough!

--Steven



# Action Center

Steven Sinofsky | [2008-11-11T03:00:00+00:00](#)

---

*We're back! We've had a pretty incredible couple of weeks at the PDC and WinHEC. Based on what we talked about you can imagine we are all rather busy as we transition from milestone 3 to beta. We trust many of you are enjoying 6801 (or perhaps we should say 6801+). Over the next few weeks we're going to start posting on the engineering and design of the specifics of different aspects of Windows 7 that we've talked about. Some posts will be very detailed and others will be a bit more high level and cover more territory. In all cases, we'll be watching the comments carefully and also looking for opportunities on follow up posts. Thank you!*

*One of the big themes of Windows 7 from a design perspective (as you might have seen in [Sam's PDC session](#) and certainly a topic we have talked about here) is making sure that you are "in control" of what is happening on your PC. This post, by senior program manager Sean Gilmour, is about "notifications" or the balloon popups that come from the system tray. In Vista we offered some controls over this area and in Windows 7 we have worked hard to make this an area that defaults to more well-behaved functionality and is also much more tunable to your needs. By improving how Windows itself uses the APIs and "guidelines" we want to encourage other ISVs to do the same. This topic is a great example of how the whole ecosystem comes into the picture as well and so we hope developers reading this will see the passion around the topic and the desire for software on Windows to take the steps necessary to honor the your intent. --Steven*

The notification area has been talked about a couple times in previous posts ([User Interface: Starting, Launching, and Switching](#) and [Follow-up: Starting, Launching, and Switching](#)). This post is going to go into a bit more detail regarding notification balloons as well as one of the ways we're working to quiet the system in Window 7.

## Where We're At Today

Windows can be a busy place – with many things vying for your attention, even while you're trying to do work. One we hear a lot about from you is the system notification balloons – those little pop-ups that appear above icons in the notification area (typically right side of the taskbar near the clock). In this post I'll be talking to notifications sent utilizing `Shell_NotifyIcon` function provided in Windows, not custom notifications, often called "toast", like the notifications presented by many applications (some like Outlook even from Microsoft). We see these in instant messenger programs, printer notifications, auto updaters, wifi and Bluetooth utilities, and more – these often use custom methods to present these "balloons" from the system tray, not necessary the Windows API. People have made their feelings loud and clear – Windows is too noisy and the noise distracts from the work at hand. Here are some quotes from the Windows Feedback Panel that illustrate that point.

*"Too many notification messages, esp. re: security (eg. Firewall), activation"*

*"Notifications telling me my system is secure, when I know it is secure, are annoying"*

*"I'm tired of error messages and pop ups."*

And some posts from the blog discussions

@Jalf writes "Having 20 icons and a balloon notification every 30th second taking up space at the taskbar where it's \*always\* taking up space is just not cool."

@Lyesmith writes "The single biggest annoyance in the taskbar is notification balloons."

So how noisy is the system? First a quick definition - a 'session' is the period of time between log-on and log-off or 24 hours whichever is shorter. As you can see from the following chart, 60% of sessions experience at least one notification. That doesn't sound all that bad, but if you dig a bit deeper you realize that 37% of sessions see two or more notifications and 25% of sessions see three or more notifications. That's a lot of distractions interrupting your work.

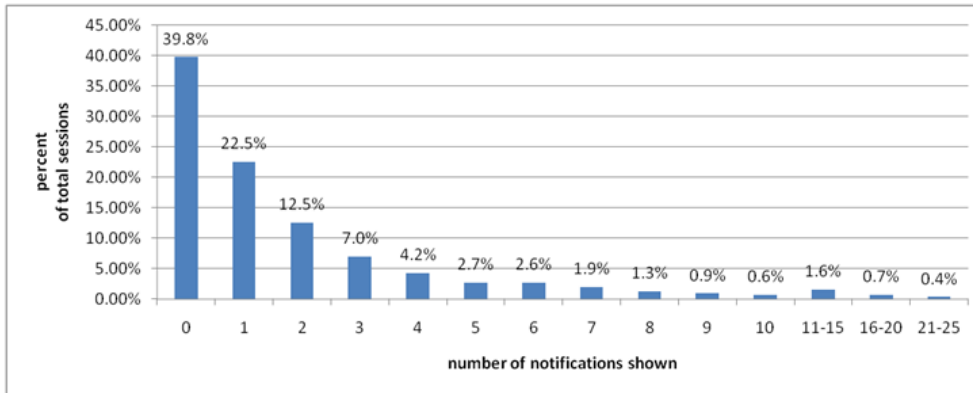


Figure 1: Number of notification sent per session as a percentage of total sessions - August through September, 2008

So we know how much noise notifications create but how effective are notifications? Well, as the following chart, notification click-through rate shows **the more notifications the less effective they become**.

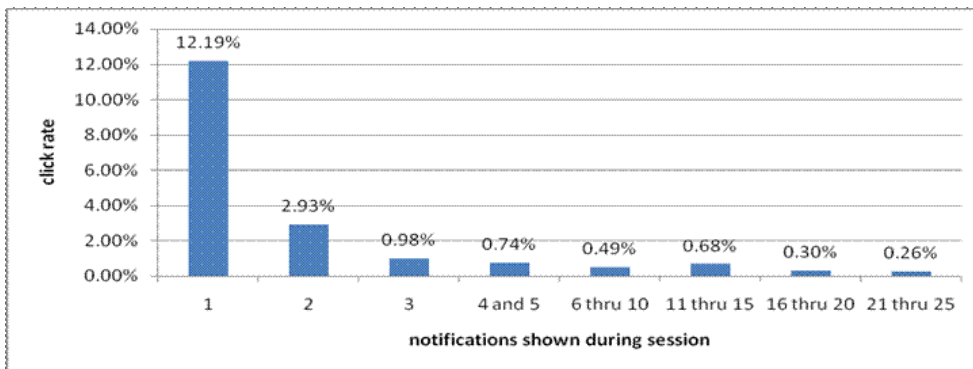
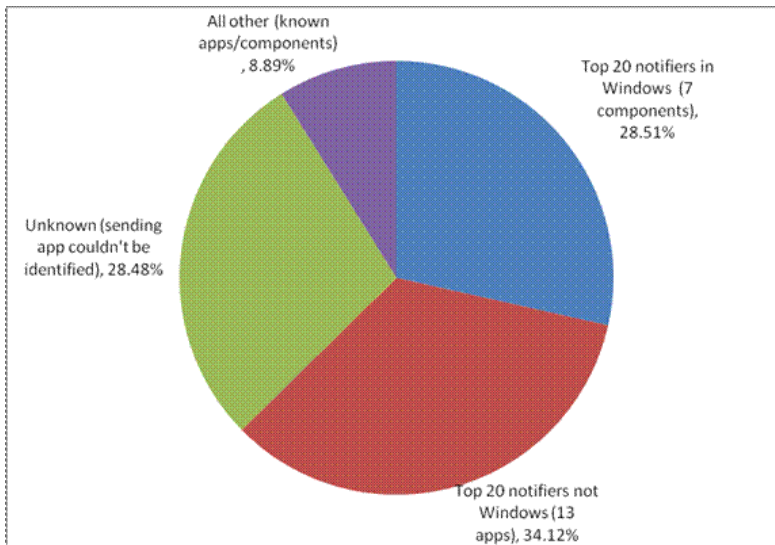


Figure 2: Notification click-through rate - August through September, 2008

So, as shown in the above chart, used sparingly and in the right context, notification balloons can be rather useful. Unfortunately, that isn't what is happening today. Instead the notification area often feels like a constant scrolling billboard of messages some important, many not. So what's the answer? It's a big area to tackle - there are system notifications, third party notification, and custom notifications. For Windows 7 we chose to focus on making sure Windows and its in-box components notify you responsibly and don't contribute to the noise in the system. Ideally the ISV

community will follow suit and as you've seen in some sessions, we're doing this work in Windows Live for example. One of the reasons we focused internally was data showing that Windows components are responsible for at least 28% of the notifications presented. Additionally, we were able to identify seven Windows components that are mostly responsible for that noise. In all, 20 applications account for 62% of the notifications presented. The following chart shows the break-out.



**Figure 3:** Which software accounts for notifications - August through September, 2008

## Windows 7

Our effort to quiet the system and make sure you are in control took the following approach:

- Working across Windows 7 to reduce unnecessary notifications
- Put you in control of the notifications you see
- Creating **Action Center** with the following goals
  - Reduce the number of notification balloons sent to you and make the ones that are sent more meaningful
  - Provide a contextual way to address the issues with a single click
  - Reduce the user-interface clutter in the system to streamline solving system issues

While there are many other efforts going around notifications and the notification area I'm going to focus on Action Center. In a nutshell, Action Center is a central location for dealing with messages about your system and the starting point for diagnosing and solving issues with your system. You can think of Action Center as a message queue displaying the items that need your attention that you can manage on your schedule. It serves as an aggregate for ten components in Windows Vista that contributed a large number of somewhat questionably effective notification balloons, but

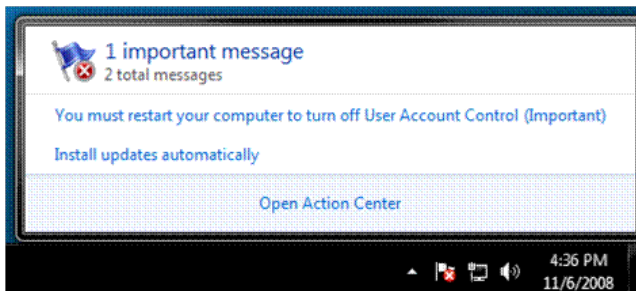
notifications that could not just be eliminated. At the heart of the Action Center effort is the idea that your time is extremely valuable it should never be wasted. To that end we took three steps.

First we looked hard at the messages we were sending and worked to reduce balloons and clarify messages. We took the following steps:

- Putting messages into one of two categories – normal or important. Normal messages simply appear in the Action Center control panel. Important messages send a notification balloon as well as appearing in the Action Center.
- Setting a high bar for important messages. A message is only deemed important if the security of the system or the integrity of your data is at risk.
- Reducing the frequency of notifications so that you're not seeing them pop-up "all the time"
- Looking at all the messages and asking the hard questions – "is this something you really need to know about?"

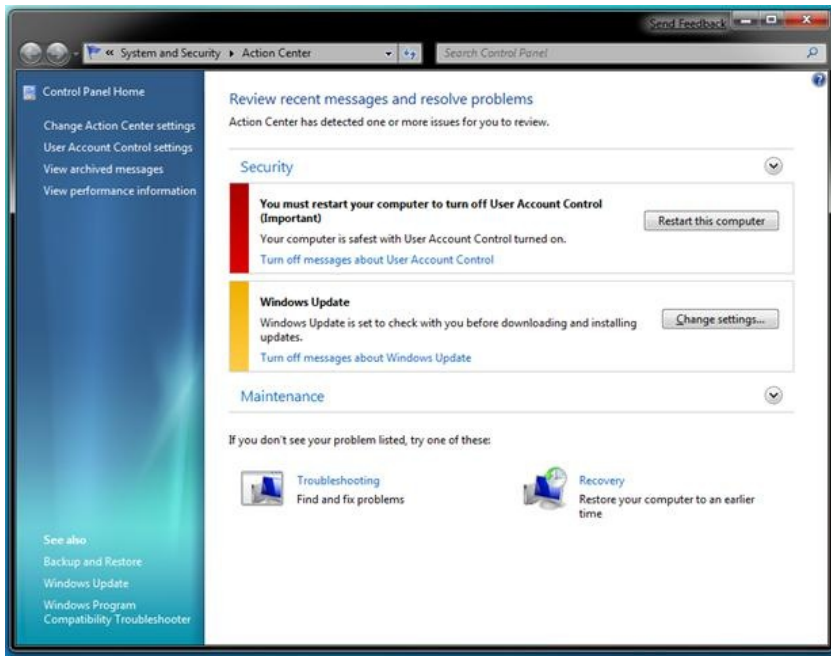
The last filter led to our second step. We decided that all messages need to have an action associated with them - a solution, if you will, to whatever problem we were presenting to you. This meant cutting any FYI, Action Success, and Confirmation messages. It also meant that the way we presented these messages would be action based. For example, we replaced, "Antivirus is out of date", with "Update Antivirus Signatures." We believe that we should let people know specifically how to resolve an issue instead of making them guess or read lots of text. This is the heart of the other goal of Action Center – to help people solve system issues quickly and conveniently.

Finally, we designed the user experience (UX) of the Action Center in two parts. The first and most immediately visible is system icon in the notification area, which is a "lighthouse" in 6801. In the spirit of our efforts, this icon replaces five notification area icons from Vista, further reducing the clutter and noise in the system. The lighthouse icon provides a high level view of the number of messages in Action Center and their importance. It also has a fly-out menu on single left click which lists the four most recent notifications and supports you acting on messages contextually. We give the people the ability to click on a notification in that fly-out menu and immediately go to the UI to solve the issue. Again, the focus is solving issues instead of simply notifying.



**Figure 4:** Action Center notification area icon and fly-out menu

The second part of the UX is the control panel, which builds upon the icon and fly-out by serving as a repository for all messages as well as providing more details about the issue and the solution. It is also action based so the layout emphasizes messages and the corresponding solutions with even more detail. Additional actions are available if you expand the UI to view them. Finally, we know that we won't always have messages about the issues a person might be having on their machine. To make sure you can solve those issues, we provide top level links to Troubleshooter and Recovery options.



**Figure 5:** Action Center Control Panel with a few messages queued up

Action Center boils down to understanding that your time is valuable and that it is your PC you want to control, not be controlled by your PC. We reduced messages, focused on solving issues not just telling you about them, and streamlined the experience so you can focus on what you want to do not what Windows needs you to do. We are aiming to get most sessions down to zero notifications from Windows itself. This reduction in notifications could significantly increase the possibility that the notification balloon will be effective in delivering its message and prompting user action as shown in the Figure 2 (notification click through).

We will of course be evangelizing to ISV the goal of following this direction and reducing notification balloons – and we believe we’ve taken the first steps to making Windows a quieter place. Hopefully you will find it less distracting and easier to work with.

*Sean Gilmour, senior program manager*

# Disk Space

Steven Sinofsky | [2008-11-19T03:00:00+00:00](#)

---

*This post is about disk space and the disk space “consumed” by Windows 7. Disk space is the sort of thing where everyone wants to use less, but the cost of using a bit more relative to the benefits has generally been a positive tradeoff. Things have changed recently with the availability of solid-state drives in capacities significantly smaller than the trend in spinning drives. Traditionally most all software, including Windows, would not hesitate to consume a 100MB on a specific (justified) need when looking at a 60GB (or 1,500GB) drive; with desirable machines shipping with 16GB of solid-state storage, we are looking carefully at the disk space used by Windows—both at setup time and also as a PC “ages”. We also had a specific [session at WinHEC](#) on solid-state drives that might be interesting to folks. This post is authored by Michael Beck, a program manager in the core OS deployment feature team. –Steven*

Let’s talk about “footprint”. For the purposes of this post, when I say “footprint” I’m talking about the total amount of physical disk space used by Windows. This includes not only the Windows binaries, but *all* disk space consumed or reserved for system operations. Later in this entry, I’ll discuss in detail how the disk footprint is consumed by various Windows technologies.

A number of comments have asked about disk footprint and what to expect in terms of Windows 7’s usage of disk space. Like many of the design issues we have talked about, disk space is also one where there are tradeoffs involved so this post goes into the details of some of those tradeoffs and also discusses some of the feedback we have received. It should be noted, that we are not at the point where we are committing to system requirements for Windows 7, so consider this background and engineering focus.

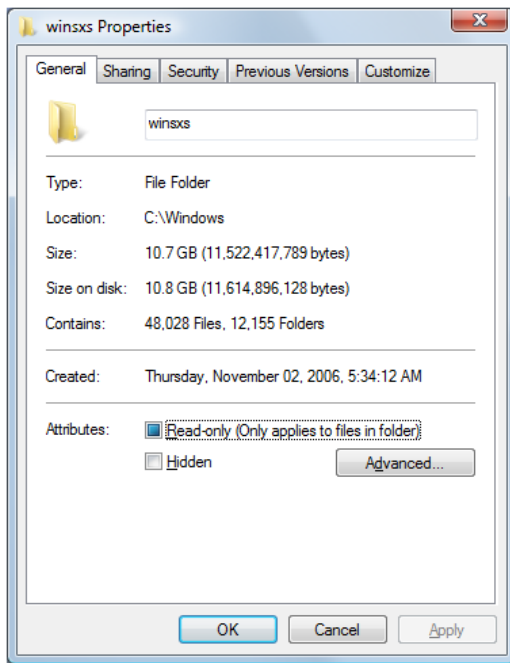
To structure this post we’ll take two important points of feedback or questions we have received:

- What does the WinSxS directory contains and why is it so big, and can I just delete it?
- Where does all the disk space go for Windows components?

We’ll then talk about the focus and engineering of Windows 7.

## WinSxS directory

We definitely get a lot of questions about the new (to Vista) Windows SxS directory (%SystemRoot%\winsxs) and many folks believe this is a big consumer of disk space as just bringing up the properties on a newly installed system shows over 3000 files and over 3.5 GB of disk consumed. Over time this directory grows to even higher numbers. Yikes--below is an example from a Steven’s home PC.



“Modularizing” the operating system was an engineering goal in Windows Vista. This was to solve a number of issues in legacy Windows related to installation, servicing and reliability. The Windows SxS directory represents the “installation and servicing state” of all system components. But in reality **it doesn’t actually consume as much disk space as it appears** when using the built-in tools (DIR and Explorer) to measure disk space used. The fact that we make it tricky for you to know how much space is actually consumed in a directory is definitely a fair point!

In practice, nearly every file in the WinSxS directory is a “hard link” to the physical files elsewhere on the system—meaning that the files are not actually in this directory. For instance in the WinSxS there might be a file called advapi32.dll that takes up >700K however what’s being reported is a hard link to the actual file that lives in the Windows\System32, and it will be counted twice (or more) when simply looking at the individual directories from Windows Explorer.

The value of this is that the servicing platform (the tools that deliver patches and service packs) in Windows can query the WinSxS directory to determine a number of key details about the state of the system, like what’s installed, or available to be installed (optional components, more on those later), what versions, and what updates are on the system to help determine applicability of Windows patches to your specific system. This functionality gives us increased servicing reliability and performance, and supports future engineering efforts providing additional system layering and great configurability.

The WinSxS directory also enables offline servicing, and makes Windows Vista “safe for imaging”. Prior to Windows Vista, inbox deployment support was through “Setup” only. IT professionals would install a single system, and then leverage any number of 3<sup>rd</sup> party tools to capture the installed state as a general image they then deployed to multiple systems. Windows wasn’t built to be “image aware”. This meant that greater than 80% of systems were deployed and serviced using a technology that wasn’t supported natively, and required IT departments to create custom solutions to deploy and manage Windows effectively. In addition, state stored in the WinSxS directory can be queried “offline”, meaning the image doesn’t have to be booted or running, and patches can be applied to it. These two features of WinSxS give great flexibility and cost reductions to IT departments who deploy Windows Vista, making it easier to create and then service standard corporate images offline.

While it’s true that WinSxS does consume some disk space by simply existing, and there are a number of metadata files, folders, manifests, and catalogs in it, it’s significantly smaller than reported. The actual amount of storage consumed varies, but on a typical system it is about 400MB. While that is not small, we think the robustness provided for servicing is a reasonable tradeoff.

So why does the shell report hard links the way it does? Hard links work to optimize disk footprint for duplicate files all over the system. Application developers can use this functionality to optimize the disk consumption of their applications as well. It’s critical that any path expected

by an application appear as a physical file in the file system to support the appropriate loading of the actual file. In this case, the shell is just another application reporting on the files it sees. As a result of this confusion and a desire to reduce disk footprint, many folks have endeavored to just delete this directory to save space.

There have been several blogs and even some “underground” tools that tell you it’s ok to delete the WinSxS directory, and it’s certainly true that after installation, you can remove it from the system and it will *appear* that the system boots and runs fine. But as described above, this is a very bad practice, as you’re removing the ability to reliably service, all operating system components and the ability to update or configure optional components on your system. Windows Vista only supports the WinSxS directory on the physical drive in its originally installed location. The risks far outweigh the gains removing it or relocating it from the system, given the data described above.

### **Where does the disk space go?**

As we all know adding new functionality consumes additional disk space--in Windows or any software. In reality, “code” takes up a relatively small percentage of the overall Windows footprint. The actual code required for a Windows Vista Ultimate install is just over 2GB, with the rest of the footprint going to “data” broadly defined. Let’s dig deeper into the use of storage in a Windows Vista installation and what we mean by “data”.

Reliability and security were core considerations during the engineering process that built Windows Vista. Much of the growth in footprint comes from a number of core reliability features that users depend on for system recovery, performance, data protection, and troubleshooting. Some of these include system restore, hibernation, page file, registry back up, and logging. Each of these represent “backup state” that is available to the system to recover from any number of situations, some planned and others not. Because we know that different customers will want to make different tradeoffs of disk space relative to recovery (especially on small footprint devices) with Windows 7 we want to make sure you have more control than you currently do to decide ahead of time how much disk space to use for these mechanisms, and we will also tune our defaults to be more sensitive to overall consumption due to the changing nature of storage.

System restore and hibernation are features that help users to confidently recover their system and prevent data loss, in a number of situations such as low battery (hibernation), bad application installation or other machine corruption (system restore). Combined, these features consume a large percentage of the footprint. Because of the amount of space these use, they are easy to identify and make decisions regarding.

System restore protects users by taking snapshots of the system prior to changes and on regular intervals. In Windows Vista, system restore, is configured to consume 300mb minimally, and up to 15% of the physical disk. As the amount of space fills up with restore points, System Restore will delete older restore points to make room for new ones. The more space you have, the greater the number of restore points you have available to “roll back” to. We have definitely heard the feedback from Windows Vista customers around system restore and recognize that it takes significant space and is not easy to tune. Some have already seen the pre-beta for Windows 7 provides an interface to manage the space better.

Hibernate is primarily used on mobile PCs and saves your work to the hard disk and puts the computer in an extremely low power state. Hibernate is used on mobile PCs when the battery drains below a certain threshold or when turning the computer off without using Shut Down to extend battery life as much as possible. On Windows Vista, Hibernate is also automatically used with Sleep on desktop PCs to keep a backup copy of open programs and work. This feature is called Hybrid Sleep and is used to save state to the hard disk in case power fails while the computer is sleeping. Hibernate writes all of the content in memory (RAM) to a file on the hard drive named Hiberfil.sys. Therefore, the size of the reserved Hiberfil.sys is equal to the amount of RAM in the machine. In the Windows Vista timeframe, the amount of RAM being built into computers has increased significantly, thus the disk footprint of Hibernate is more noticeable than before. This space must be reserved up front to guarantee that in a critical low battery situation, the system can easily write memory contents to the disk. Any mobile PC user that has experienced their computer automatically entering Hibernate when the battery is critically low can appreciate the peace of mind this footprint growth provides. While we’re talking about RAM and disk footprint in the same paragraph, Mark Russinovich has a post this week on [virtual memory](#) and how big the swapfile could/should/can be that you might find interesting.

Now it’s clear that in the description above, I don’t account for the entire footprint required by Windows Vista. For instance, we also include many sample files, videos, high resolution backgrounds that allow users to easily customize their experience, and try out new features, but we’ve covered a couple of the more common questions out there.



It's important that we consider more than just the size of the system once deployed, but we must also look at how the system grows over time as services write logs, updates, and service packs are installed, system snapshots are taken etc. For many, the “growth” over time of the installation proves to be the most perplexing—and we hear that and need to do better to (a) make smarter choices and (b) make it clearer what space is being consumed and can be reclaimed.

The following table provides one view of the installation footprint of a Windows Vista Premium/Ultimate installation. This includes the full installation, but to make it digestible this has been broken down into some logical categories and also highlights some specific features. Part of the reason to highlight specific feature is to illustrate the “costs” for items that have been raised as questions (or questionable).

Description	Vista SP1
<b>Core</b>	
Desktop code	160
Windows Components (printing, WMI, RAS, Terminal Services, DirectX, etc)	531
Non-PnP Boot Critical Drivers	12
Fonts (including global UNICODE fonts)	315
IME Components (East Asia input)	220
Internet Explorer	25
NLS (Natural Language Support)	30
Speech components	352
<b>Drivers</b>	
Boot Critical PnP Drivers	84
Printer Drivers	805
Other PnP Drivers	351
Installed Language Pack Resources	518
<b>Features</b>	
Games	100
Internet Information Server	25
Handwriting and Ink functionality	200
Windows Media Center	110
Windows Media Player	53
DVD Maker	83
.Net Framework 3.x	60
Windows Photo Gallery	15
Music / Movies / Wallpaper	250
<b>Other Files 'within one month'</b>	
System Registry	150
Windows Search Database	19
Log Files (at install time)	52
Per-User Data created by Windows	22
Installed Drivers & INF Files	103
Native .NET Framework Files	224
Filesystem Infrastructure	134
<b>Infrastructure ...WinSxS</b>	
Contents of WinSxS	378
Manifests, Packages, and Catalogs	214
Superseded by SP and not Removed	270
<b>Fixed Size Data</b>	
hiberfil.sys (MB)	1,076
pagefile.sys (MB) for 1GB of install RAM	1,330
System Restore	2,000
IE Cache	50

Here are some items worth calling out:

- ~1GB driver support. Windows Vista works with thousands and thousands of different devices. The ability to plug in almost any device, even your old printer and have it get recognized and install automatically is something customers have come to expect from Windows. We receive lots of feedback wanting to remove some or all of these and each release we carefully scrub the “in-box” device support relative to what we see from telemetry in terms of used devices. The ability to install a printer or USB device while offline is a key value, especially with laptops representing over half of all PCs being sold. In the future we can possibly assume “always go to Windows Update” but we’re not there yet in most places in the world.

- ~1GB of system growth in serviced and superseded components to support robust rollback and recovery, after installing critical security and functionality updates. We receive a lot of positive feedback about the robustness of servicing but at the same time, the desire to rollback a specific fix for any variety of reasons remains an important robustness and reliability measure. We also understand the feedback we have received regarding the disk requirements to install SP1 on top of RTM. We hope folks are aware of the [vsp1cln.exe](#) utility in the system32 directory, for those that are in need of disk space.
- ~1GB hibernation support is necessary in order to prevent data loss when a machine has been in standby for many hours. This can be removed via the Disk Cleanup wizard or via an elevated command prompt (powercfg /h off).
- ~315mb of Fonts. Windows users speak many different languages, often on the same PC, and wish Windows to “speak” to them. Windows Vista contains native font support to allow users with systems defaulted to one language to be able to read documents, or websites in another. As we know, however, fonts are easy to delete should you desire.
- ~52MB of log files. Whether it is the event log, servicing logs, or device installation logs (or more) this space is consumed and becomes critical when trying to diagnose a failure. These logs are often used by our support personnel or corporate helpdesks to diagnose a specific failure.

## Engineering Windows 7

Windows disk space consumption has trended larger over time. While not desirable, the degree to which it’s been allowed is due in large part to ever-increasing hard drive capacity, combined with a customer need and engineering focus that focused heavily on recoverability, data protection, increasing breadth of device support, and demand for innovative new features. However, the proliferation of Solid State Drives (SSDs) has challenged this trend, and is pushing us to consider disk footprint in a much more thoughtful way and take that into account for Windows 7.

This doesn’t mean that we’re going to stop adding great features or make Windows less reliable or recoverable. As we look to the future, it’s critical that as we innovate, we do so treating the disk space consumed by our work as a valuable resource, and have a clearer design for how Windows uses it. We want to make sure that we are making smart choices for the vast majority of customers and for those desiring more control providing places to fine tune these choices as appropriate. This design goal isn’t about a type of machine, or specific design, all Windows editions benefit from efforts that focus on a reduction of the overall footprint.

For example, as we consider the driver support discussed above, Windows Vista with SP1 installs almost 1GB of drivers on the system to support plug and play of devices. This local cache can get out of date as IHVs release updates to their drivers, and as a result, users are pushed to Windows update to get the latest version once the device is installed.

Why not extend the PnP user experience to include (or only use) the Windows Update cache of drivers and save some disk space? This has several benefits:

1. Because MobilePCs rarely lack a network connection, they can simply get the new driver from the web.
2. People don’t have to install the driver twice on updated devices because they do the round trip to the web anyways.

With this example it’s easy to see how engineering for a minimal footprint might actually deliver a better experience for people when attaching new devices to their systems. At the same time, we want to be careful about going too far too soon. We get a tremendous amount of feedback regarding the “plug and play” experience or feedback about costly download times (if download is at all possible). For Windows 7 we are going to continue to be deliberate in what we include based on the telemetry of real world devices and reducing the inbox set to cover the most popular devices around the world. At the same time we will continue a very significant effort around having the best available Windows Update site for all devices we can possibly support.

Windows features installed by default make sense in most cases to support many scenarios. We should consider how we make some features/components (such as Media Center) optional when they are not required rather than installing them by default on every system. We’re

committed to make more features of Windows optionally installed. As you might notice today in Windows, when you choose to add a feature that was not installed Windows does not require a source (a DVD or network location). This is because the feature is stashed away as part of a complete Windows install—this is itself a feature. We will always keep features available and will always service them even when components are not installed—that way if you add a component later you do not risk adding a piece of code that might have been exploited earlier. This is another important way we keep Windows up to date and secure, even for optional features.

System growth over time is an area where we need to provide more “transparency”. For instance, Windows will archive previous versions of updated system components to allow robust rollback. A new system will install patches as Windows Update makes them available, just as expected by design. As a Service Pack or other large update is installed that contains or supersedes any of the previous patches; we can simply recover the space used by the old updates sometime after the update is successfully installed.

Windows writes logs in many places to aid in troubleshooting and these logs can grow very large. For instance, when an application crashes, Windows will archive a very large dump file to support analysis of the failure. There are many good reasons for this behavior, but as we change our mindset towards footprint, we need to extend our scenarios to include discussions of how to manage the growth, and recover the disk space consumed whenever possible. Other areas where we are considering the default disk space reserved include System restore and hibernation. On a disk constrained system, the 1GB or more reserved to support hibernation is costly and there may be ways to shrink the size of hiberfil.sys. System restore should be configurable, and default in all cases to the minimally useful number of snapshots vs. a blanket 15% of the system disk.

At WinHEC we had several machines on display with 16GB drives/partitions and on those you could see there was plenty of free disk space. Like all the benchmarks, measuring disk space on the pre-beta is not something we’re encouraging at this time.

In conclusion, as we develop Windows 7 it’s likely that the system footprint will be smaller than Windows Vista with the engineering efforts across the team which should allow for greater flexibility in system designs by PC manufacturers. We will do so with more attention to defaults, more control available to OEMs, end-users and IT pros, and will do so without compromising the reliability and robustness of Windows overall.

-Michael Beck

# The Windows 7 Taskbar

Steven Sinofsky | [2008-11-20T03:00:00+00:00](#)

*Happy Birthday Windows! Given all the interest in the most used user-interface of Windows we thought it would be good to take a look back and see how we got to Windows 7. --Steven*

We were very excited to unveil elements of the Windows 7 desktop at this year's Professional Developers Conference (as seen in the [Welcome to the Windows 7 Desktop](#) session, among others). In previous posts ([User Interface: Starting, Launching, and Switching](#) and [Follow-up: Starting, Launching, and Switching](#)) we looked at the history, anatomy and areas for improvement of the taskbar. In this post, we will continue the conversation. Don't let looks fool you though—the UI may feel new to Windows for some of you or old hat for some of you, but rest assured it represents a careful evolution that strives to address customer feedback while retaining its familiar Windows DNA.

It was 23 years ago on November 20, 1985 when Windows first shipped. As it just so happens, this first Microsoft graphical shell actually holds relevance to this post as it surfaced one of the industry's first taskbar-like concepts.



Fig 1 Windows 1.01: Icons at the bottom of the screen represent running windows

Windows 1.0 supported zoomed (full-screen), tiled and icon (minimized) windows. Since there was no support for overlapping [that big debate between charless and bill, *Steven*], a dedicated portion of the desktop was kept visible at the bottom of the screen to surface non-tiled and non-zoomed windows. By minimizing a window or dragging it to the bottom of the screen, the person was able to populate this rudimentary taskbar with a large icon corresponding to the running window. She could then get back to this window by clicking or dragging this icon to the desktop. As simple as this mechanism seems today, it cemented an important concept that is with us even in Windows 7—when people switch between tasks, they are really switching between windows. Although it took Windows 95 to introduce a mature taskbar with launching, switching and notification functionality, the experience of surfacing and switching between windows via a dedicated region at the bottom of the screen is as ancient as Windows 1.0.

## Setting Goals

In the previous taskbar posts, we discussed some high-level principles we defined after digesting the mountain of data and feedback on the taskbar. Here's a more detailed look at the goals we identified and how we began to frame feature concepts.

## Things you use all the time are at your fingertips

*It is easy to get to the programs and destinations you use all the time, with less mouse movement and fewer clicks.*

Accessing commonly used programs within a single click required us to enrich Quick Launch by increasing its presence on the taskbar and making more top-level room for pinned items. We began looking into how Quick Launch interacted with the taskband and how launching and switching were sometimes separate and other times duplicative. For example, almost all single-instance programs in Windows interpret an attempt to re-launch them as a switch if they are already running. So, clicking Outlook's icon in Quick Launch would merely switch to the program if it was already running and present in the taskband. To make room for more items on the taskbar, we knew we had to remove some of the redundancy and free up valuable real-estate.

When researching and modeling a person's workflow, we came to realize that there were three basic steps that a person frequently seems repeats. First, she finds the program and launches it. Then, she uses the program's UI to open a file she wants to work on. Then finally, she gets to work. We asked ourselves whether we could help people jump directly to these items by skipping the first two steps. We called these files, folders, links, websites and other items that programs create or consume "destinations" as they represent where the person is ultimately is navigating to. We decided that these destinations should also be easily accessible from the taskbar. However, for real success and adoption, we needed to think through how destinations could be effectively surfaced to the person without the need for manual customization or by requiring developers to do lots of work.

## **Manage your windows with confidence**

*You can switch to the right window quickly without mistakes and effortlessly position windows the way you want them.*

This goal spoke to the very heart of the taskbar—the ability to switch between windows. This challenged us with seeking a more predictable method of surfacing windows on the taskbar, meaningful use of text and a reliable method of helping people *consistently* switch with confidence. We've had text on the taskbar for years and Vista introduced thumbnails, but customer feedback informed us that there was room for improvement. Interestingly, we found inspiration in old features such as Windows XP's window grouping and Alt-Tab's visual layout of individual windows.

During our investigation, we also spent time looking into why a person would switch windows in the first place. Two interesting scenarios emerged—one in which she needs to get some information from a window (e.g. getting a phone number) and to interact with a window's options (e.g. controlling background music). We wondered whether we could address these task switching cases in a novel way—by actually removing the need to switch completely.

## **You are in control**

*The desktop reflects your style. You get to personalize the experience, choosing what is important to you, including how and when you receive notifications.*

By far the biggest target of feedback, the Notification Area had to put control back in the hands of people. It was decided that instead of the opt-out model that required the person to clean up this area, we would start with a clean experience. Only system icons would appear by default and then people can to customize this area to their liking.

## Clean and lightweight

*The desktop experience feels organized, lightweight, open and is a pleasure to use. Visuals and animations are delighters the first time and every time.*

A successful product is more than the utility it serves—it is also an experience. From the very start we wanted the taskbar, and the desktop as a whole, to draw an emotional response from the person. This required a set of scoped delighters that demoed well and retained their appeal over time. We began to define a personality for the UI using terms such as “glass and energy,” Chi, authenticity and many others. These investigations helped define a visual and animation language that we could then apply to several aspects of Windows 7. Expect a future blog post that delves much deeper into this important design process—much of which Sam discussed in his [PDC session](#).

## The Taskbar, Evolved

The Windows 7 taskbar is about launching with ease, switching with confidence and all the while remaining in control. The UI is made up of several key features that complete common end-to-end scenarios. Let’s dive into each of these elements and how they work.

## Refreshed Look

The taskbar has undergone a facelift. We’ve enabled large icons by default (as seen in Windows 1.0 and also an option of Quick Launch since Windows 95 with IE 4). This affords a richer icon language, improves identification of programs and improves targeting for both the mouse and touch. Yet, one of the most important advantages large icons provide is a means to promote the taskbar as *the* central place to launch everyday tasks. We joke that the new taskbar is the “beachfront property of the Windows OS” and in turn, we are already seeing many people populating the UI with their commonly used programs. Somewhat if a visual trick, the taskbar is only 10 pixels (at 96 DPI) higher than its Vista counterpart (when used as a single row, since multiple rows are still supported, along with positioning around the screen edges).



Fig 2. The Windows 7 taskbar: Default settings include large icons, no text and glass surface

To mitigate its slightly increased height and the larger icons, we decided to impart the UI with a more prominent glass treatment. This also allows us to better showcase the person’s color preference (you’ll recall that in a previous post we revealed that almost 30% of sessions have personalized glass). We also changed the Vista behavior so that when a window is maximized, both the taskbar and the window’s title bar continue to remain open and translucent. We received lots of feedback on Vista that many people didn’t like these UIs turning opaque and dark.

## Pinning

You can still pin programs to the taskbar by dragging them or via a context menu, just like you have always done with Quick Launch. Destinations can also be pinned via a drag/drop, but they are designed to be surfaced differently as we'll see under the Jump List section.

## Unification

If one increases the size of Quick Launch, one must then determine what to do with the taskband. As previously discussed, we observed that under many scenarios of single-instance programs, launching and switching were equivalent. Hence, we decided to standardize this behavior and have program launchers turn into window switchers when they are launched. Effectively, we unified Quick Launch and the taskband. While some other operating systems have similar concepts, one difference with our approach is that our default experience always optimizes for a single representation on the taskbar. This means that regardless of a window's state (e.g. minimized, maximized or restored) there are no new or duplicate buttons created. Also, the default taskbar doesn't allow destinations to be pinned to the top-level which prevents duplication of a pinned file and a running window with that same file open. When we say there is "one button to rule them all" we're serious. This approach to a single, unified button keeps the taskbar uncluttered and gives the person a single place to find what she's looking for.

Combining launching and switching also made it easier to provide the most requested feature—the ability to move taskbar buttons. Quick Launch as always allowed this, but combining this mechanism with the taskband naturally extended rearrange functionality to running windows.

## Interactive, Grouped Thumbnails

Vista showed thumbnails when the user hovers on a taskbar button and Windows 7 improves upon this design. Unlike Vista, these thumbnails are now an extension of their corresponding button so the person can click on these visual aides to switch to a given window. The thumbnail is also a more accurate representation of a window complete with an icon in the top left corner, window text and even the ubiquitous close button in the top right.



Fig 3. Thumbnails: Grouped, interactive thumbnails make it easier to manage windows

One of the most important functions of the taskbar is to surface individual windows so people can easily switch between them. Having unified a

program launcher and a single window switcher, the next logical step was to determine how multiple windows of a program could be combined and presented. We looked no further than a feature introduced in Windows XP called window grouping. When the taskbar became full, windows of a program could collapse into a single menu. However, there were a few challenges with the design. First, the behavior isn't predictable. People don't really understand when this scaling mechanism is triggered. Second, a listview of windows isn't always the best way to represent these items. Finally, opening the menu always required a click, which slowed some people down. Our solution was to combine buttons by default for a predictable experience, to use grouped thumbnails and to have these thumbnails appear on hover as well as on click. Think of this approach as a contextual Alt-tab surfaced directly off the taskbar. When the person brings her mouse to a taskbar button, all the thumbnails of a program appear simultaneously making for an organized, light-weight switching model. To polish off the experience, we show a visual cue of stacked tiles that provides feedback on whether there are multiple windows running for a program. We also recognized that a set of people may still wish to see an individual buttons for each window and an option permits this behavior.

With the Windows 7 taskbar, there is a single place to go regardless of whether the program is not running, running with one window or running with several windows. Rich thumbnails provide more intuitive ways of managing and switching between windows.

## Aero Peek

Here's a riddle for you—what's the best size for a window's preview that will *guarantee* that the you can accurately identify it? Grouped thumbnails look and feel great, but we know these small previews don't always provide enough information to identify a window. Sure they work great for pictures, but not so for emails or documents. The answer is simply to show the actual window—complete with its real content, real size and real location. That's the concept behind Aero Peek.

When the taskbar doesn't offer enough information via text or a thumbnail, the person simply moves the mouse over a taskbar thumbnail and voila—the corresponding window appears on the desktop and all other windows fade away into glass sheets. Once you see the window you want, just click to restore it. Not only does this make finding a window a breeze, it may also remove the need to switch altogether for scenarios in which one just needs a quick glance to glean information. Peek also works on the desktop too. Show Desktop has been moved to the far right of the taskbar where one can still click on this button to switch to the desktop. The control enjoys a [Fitts magic corner](#) which makes it very easy to target. If you just move your mouse over the control, all windows on the desktop turn to glass allowing the desktop to be seen. It's easy to now glance at a stock or the weather gadget or to check to see if a file is on the desktop.



Fig 4. Aero Peek: Hovering over a thumbnail peeks at its corresponding window on the desktop

We spent a lot of time analyzing different aspects of Peek. For example, we recognized that when people are using the feature, they won't be necessary focused on the taskbar as they look at windows on the desktop. An early prototype triggered Peek directly off the top-level of the taskbar but this revealed issues. Moving the mouse across a small a region to trigger different previews exited Peek since the natural arc of hand motion resulted in the mouse falling off the taskbar. By only triggering Peek off the thumbnails, we gained much more room for the mouse to arc and we also reduced accidental triggers.



## Jump Lists

As far back as Windows 1.0, there has always been a system menu that shows contextual controls for running windows and their programs. This menu is accessible by right-clicking on a taskbar button or in the top left corner of most windows. By default, the menu exposes windows controls such as close. (Random trivia—ever wonder why the system menu off a taskbar button always shows close in bold when close isn't the double-click behavior? Well, the answer is that double-clicking the top left region of most windows *will* close it and the bolded option makes sense in this context. The same menu just happens to be hosted in both locations.) Over the years, some programs have extended the system menu to surface relevant tasks. For example, Command Prompt reveals tasks such as editing options, defaults and properties in its system menu. However, this is a bit of a free-for-all for programs to opt in or not, resulting in an inconsistent experience for people. Another blow to this scenario is that the system menu is only accessible when the program is running. This makes sense since the default commands are about window management, but what if you wanted to access a program's tasks even it isn't running?

As we discussed under the goals section, we thought about the various steps people have to take to accomplish tasks and whether we could reduce them. Be it getting to a destination or accessing the commands of a program, we wanted to make it easier for people to jump to the things they are trying to accomplish. Jump Lists are a new feature of the Windows 7 taskbar that accomplish just this. Think of this feature as a mini Start Menu for each program or an evolved version of the system menu. Jump Lists surface commonly used nouns (destinations) and verbs (tasks) of a program. There are several advantages this new approach provides. First, you don't need to even start the program to quickly launch a file or access a task. Second, destinations don't take up valuable space on the taskbar; they are automatically organized by their respective program in a simple list. Should one have ten programs pinned or running on her taskbar, this means she could have quick access to over 150 destinations she uses all the time, without even the need to customize the UI! Since the Jump List shows lots of text for each of its items, gone are the days of having identical icons on your taskbar that are indistinguishable without a tooltip. Should you wish to keep a specific destination around, you can simply pin it to the list.



Fig 5. Jump List: Right-clicking on Word gives quick access to recently used documents

To make sure we provide a consistent and valuable experience out-of-the-box, we decided to pre-populate Jump Lists and also allow programs to customize the experience. By default, the menu contains the program's shortcut, the ability to toggle pinning, the ability to close one or all windows and a program's recent destinations (assuming they use the Common File Dialog, register their file type or use the Recent Items API). Programs are able to replace the default MRU (Most Recently Used) list with a system-maintained MFU (Most Frequently Used) list, should their destinations be very volatile. For example, while Word will benefit from a MRU just like the one in their File Menu, Windows Explorer has opted to enable the MFU because people tend to visit many paths throughout a session. Programs are also able to provide their own custom destination list when they have a greater expertise of the person's behavior (e.g. IE exposes their own history). Still others like Windows Live Messenger and Media Player surface tasks or a mix of tasks and destinations.

In case we haven't yet impressed it upon you, the taskbar is about a single place to launch and switch. Jump Lists offer another important piece of the puzzle as it surfaces valuable destinations and tasks off a program's unified taskbar button.

## Custom Window Switchers

All the major web browsers offer tabs and a method of managing these tabs. One could argue tab toolbars are really like taskbars since they facilitate switching. These TDI (Tabbed Document Interface) and MDI (Multiple Document Interface) programs have always resorted to creating their own internal window management systems as the Windows taskbar was not optimized to help their scenarios. Some programs like Excel did custom work to surface their child windows on the taskbar, but this approach was somewhat of a hack.

Since the new taskbar already groups individual windows of a program under a single button, we can now offer a standard way for programs that have child windows to expose them. Again, the taskbar offers a single, consistent place to access real windows as well as child windows. These custom window switchers also behave as regular windows on the taskbar with rich thumbnails and even Aero Peek.

## Thumbnail Toolbars

In the earlier taskbar posts, we discussed how Windows Media Player's deskband offers valuable background music controls, but only a mere 3% of sessions ever enjoy the functionality. The new taskbar exposes a feature called Thumbnail Toolbars that surface up to seven window controls right in context of taskbar buttons. Unlike a Jump List that applies globally to a program, this toolbar is contextual to just a specific window. By embracing this new feature, Media Player can now reach a majority of people.



Fig 6. Thumbnail Toolbar: Window controls easily accessible in context of a taskbar thumbnail

Thumbnail Toolbars leave the taskbar uncluttered and allow relevant tasks to be conveniently accessible directly from a taskbar thumbnail. Surfacing tasks reduces the need to switch to a window.

## Notification Area

We're happy to announce that the Notification Area is back under your control. By default, only a select few system icons are shown while all others appear in a menu. Simply drag icons on or off the taskbar to control the experience. Better yet, every balloon tip that appears in the system has a little wrench icon that allows one to quickly "swat" an annoying alert by immediately seeing what is causing the notification and a direct way to disable it.



Fig 7. Notification Overflow: By default icons appear in an overflow area that you can then promote

Interestingly a very popular change to Notification Area isn't about reducing noise, but rather showing more information. The default taskbar now reveals both the time and the date. Finally!

## Overlay Icons and Progress Bars

Cleaning the Notification Area warrants us to consider other ways that programs can surface important information. We'll always had overlay icons throughout Windows (e.g. to show shortcuts in Explorer) so we decided to bring this functionality to the taskbar. An icon can now be shown over a program's taskbar button. Furthermore, programs can also give feedback about progress by having their taskbar button turn into a progress bar.



Fig 8. Progress Bars: Explorer utilizes taskbar progress to show a copy operation in process

A program can now easily show an icon or progress in context of its taskbar button which furthers the one place, one button philosophy of the taskbar.

## Color Hot-track

Color hot-track is a small touch that typifies the new taskbar's personality. When a person moves her mouse over a running program on the taskbar, she will be pleasantly surprised to find that a light source tracks her mouse and the color of the light is actually based on the icon itself. We calculate the most dominant RGB of the icon and dynamically paint the button with this color. Color hot-track provides a delight factor, it offers feedback that a program is running and it showcases a program's icon. We've always believed that programs light up the Windows platform and now, we're returning the favor.



Fig 9. Color Hot-track: moving the mouse across a running window reveals a dynamically colored light effect

## Start Menu

Vista introduced several changes to the Start Menu so we decided to minimize churn to this UI in Windows 7. Notable improvements include the availability of Jump Lists and a better power button that defaults to Shutdown, but makes it easy to customize.

## Different, Yet Familiar

Despite all the features of the new taskbar, it is worthwhile noting the UI retains its familiarity. We like to describe our work as evolutionary, not revolutionary. The taskbar continues to be a launch surface, a window switcher and a whisperer of notifications. Whether one is relatively new to Windows or a seasoned pro, we realize change comes at a cost. It is for this reason that we took the time to carefully evaluate feedback, we performed numerous studies to validate our designs and finally, we will continue to provide scoped settings that keep the UI flexible.

We hope this post provided more insight into the new Windows 7 taskbar. Expect future discussions on our design process, how we tested our features and advanced functionality for all you enthusiasts.

- Chaitanya

# Accessibility in Windows 7

Steven Sinofsky | [2008-11-30T03:00:00+00:00](http://www.microsoft.com/enable/)

---

*This post is from Michael Bernstein, a development lead on the User Interface Platform team where he focuses on accessibility. Accessibility is the term we apply to the APIs and features that enable Windows to be used, to be accessible, by as many people as possible so that, regardless of physical or cognitive abilities, everyone has the ability to access the functions of Windows. To enable this, Windows includes both built-in accessibility utilities as well as APIs used by third party assistive technology aids and by application developers to make sure their software is also accessible. This is a topic that is extremely important to Microsoft and one that is a key tenet in the engineering of Windows 7. Microsoft also has a corporate-wide group dedicated to making sure that PCs are easier to see, hear, and use. You can read more about Microsoft's accessibility initiatives on <http://www.microsoft.com/enable/>. –Steven*

Hi, I'm the development lead for Accessibility and Speech Recognition experiences in Windows 7, and I wanted to write about how we thought about Accessibility in Windows 7.

We wanted to make Windows 7 the most accessible operating system that Microsoft has ever produced. It became clear as we planned this release, however, that the notion of Accessibility is not as simple as it may appear. It is tempting to think about Accessibility like Security: either you have a known failure, or your system is believed to be secure/accessible. This definition turns out to be limited, though. How do you deal with the fact that the needs of customers who are blind are very different from the needs of customers who are deaf? The needs of customers who are blind are even different from those of customers with reduced vision: a magnification tool is useless for one group and crucial for the other. And what do we make of cases where something is technically accessible but practically frustrating, like a common user scenario that takes 36 keystrokes to execute? Clearly, Accessibility wasn't going to boil down to a simple yes/no question. It is really more like a particular kind of usability, but usability for a specific set of audiences with individual needs.

Since the questions we were asking were complex, the answers ended up being complex, too. We chose a four-part strategy to improve Accessibility in Windows 7.

## I. Build a firm foundation with UI Automation

In Windows Vista, Microsoft delivered a new core component for Accessibility called *UI Automation*. UI Automation enables a user's assistive technology (AT) to programmatically drive the UI of an application, and allows applications to expose their accessible functionality in a richer way than was possible in previous versions of Windows. More questions can be asked about a piece of UI, and that UI can be manipulated in richer ways. UI Automation also introduced the idea of *Control Patterns*: any given piece of UI can decide how it should be controlled. Buttons expose the *Invoke* pattern, indicating that they can be pushed; Combo Boxes expose *ExpandCollapse*, indicating that they can be opened and closed. We let different controls be different, instead of trying to force them all into the same mold. All this was introduced in Windows Vista and adoption is still ongoing.

In Windows 7, we invested in improving the performance of the UI Automation system and created a new, native-code API for UI Automation to make sure that it can be used effectively by a wide range of assistive technology software. Now applications written in C++, as well as those written using the .NET Framework, can take advantage of UI Automation.

We also did a bunch of work to make sure that the UI Automation system was integrated even more closely with the legacy Microsoft Active Accessibility (MSAA) system and developed new bridging techniques between the best of the new and the old technologies. UI Automation Clients can read Accessibility information from MSAA applications, and vice versa, to ensure maximum Accessibility regardless of which accessibility API an application used originally. Since the UI Automation and MSAA systems cooperate closely in many scenarios, we decided to name the combination of the two, calling it the *Windows Automation API*. This architecture forms the foundation for the rest of our Accessibility effort, and we're pleased to have this Accessibility foundation Windows 7.

## II. Improve our included Accessibility utilities

We also improved the Accessibility utilities that we include in the box with Windows. Microsoft works closely with many different AT software companies who deliver software to make Windows more accessible to customers with disabilities, but we also include a set of utilities to make sure that our customers' early experiences are accessible, even before installing any other software. We decided to enhance two of those utilities in Windows 7: the On-Screen Keyboard and the Magnifier.

The most noticeable change to the On-Screen Keyboard is the improved look and feel, but there are also more subtle enhancements. The appearance of this utility had not changed since Windows XP; our customers were also asking for it to be resizable. We addressed both of these by working closely with Tablet developers to share a common code base between the Tablet Soft Keyboard and the On-Screen Keyboard. Both keyboards now have an attractive appearance that is in tune with Windows 7 and both are now resizable. The keyboards still are distinct, though, because customers use them differently: Tablet users may want to switch dynamically between handwriting and typing, whereas On-Screen Keyboard users may need modes where they can hover or scan to keys, if they have disabilities that prevent them from clicking. Along these lines, we also added basic text prediction to help customers with disabilities enter text more quickly. If you have ever tried typing with an on-screen keyboard, you can appreciate how significantly text prediction can improve text entry speed.

The Magnifier came in for a deeper overhaul. The Magnifier in Windows Vista and Windows XP was not an intuitive experience: when you pointed at part of the screen, the magnified content appeared in a separate window, usually docked at top of the screen. You had to point at one place and look at another. We considered two basic solutions to this problem: you could zoom into the entire screen or you could make the magnified area follow the pointer while leaving the rest of the screen the same. These became our two primary modes for the Windows 7 Magnifier: Full-screen mode and Lens mode.

Full-screen mode is great when you want to increase the size of everything on the screen at once. As you move the mouse or keyboard focus around the middle of the screen, the view stays still; if you move towards the edge, the Magnifier scrolls the view to keep up. One downside of this mode is that you can lose track of your context. To address that usability issue, we added a context animation that zooms out to show you where your work area is relative to the whole screen, and then zooms back in.

Lens mode, on the other hand, is nice when you just want to zoom in on one particular thing. In this mode, the lens centers on the mouse pointer, which feels much like using a magnifying glass. You can re-size the lens to be very wide and short, which can be nice if you are reading a document and want to magnify it line by line. We based our design on the popular Microsoft IntelliPoint magnifier, a design you can now enjoy with any mouse.

We also addressed customer feedback about the Magnifier window taking up too much space on the screen. We moved the most commonly used controls like zoom in/out to a small toolbar, which fades out to a semi-transparent watermark when you aren't using it. The remaining options are available in an Options dialog when you need them. Last, we gave almost everything a keyboard shortcut, so if you really don't want to see the UI, you don't have to use it. Win+ will zoom you in any time you are using Windows 7.

These tools directly improve Accessibility for customers with low vision and dexterity disabilities. It should be obvious, but making the PC easier to see or interact with benefits everyone and so these two examples also show the broad appeal of AT tools – at the PDC we showed both the On-Screen Keyboard and the Magnifier and I think it is fair to see everyone saw the benefit of using these tools themselves, regardless of abilities.

### **III. Make it easier to build Accessible software**

Windows APIs cannot provide Accessibility all by themselves; it is vital for Windows-based applications to do their part in providing Accessibility data for AT programs to use. For example, a screen reader may sound excellent, but if it can't read your favorite web browser, what good is that? Assistive tools like screen readers and magnifiers are *clients* of the Accessibility system, while the applications that you want to use, like web browsers and word processors, are *providers*. It takes both to make the whole experience accessible--you need both a high-quality client and a well-written provider to have a good Accessible experience. There are more providers in the software ecosystem, so it is hard for us to work one-on-one with every provider to make sure they are well-written.

To address this challenge, our team developed the *UI Accessibility Checker* (*AccChecker* for short) and *UI Automation Verify* (*UIA Verify*) utilities, which can scan an application (a provider, really) and report on common Accessibility problems. Software developers can use *AccChecker* and *UIA Verify* to detect problems in their provider code before a customer ever uses it. Quality assurance engineers can use them to verify the quality of their firm's work. We believe this is so important that we released [AccChecker](#) and [UIA Verify](#) as open-source software to make it available to the widest possible audience. If you are not a programmer, you may never use these utilities directly, but you may well benefit from the bugs they helped to eliminate before they ever reached you.

#### **IV. Plan for Accessibility from Day 1**

To make sure that Windows features themselves were good providers, we borrowed an idea from the [Software Development Lifecycle](#), risk assessment. Before a line of code was written, each planned Windows 7 feature was rated on its Accessibility risk. Features that use more basic, off-the-shelf common controls are usually more accessible because Windows provides built-in providers for off-the-shelf components; features that do fancy, custom drawing have more work to do. This planning process made each team aware of how much accessibility risk it was taking on, so that they could plan appropriately. Once the features were all rated, the list was sorted by risk so that our team could reach out to teams with high-risk features and make sure that they had the resources and tools they needed to make their feature properly accessible. We also ensured that they received more hands-on testing and validation. As a result, most Windows features are more accessible than they have been in previous releases, making for a better overall customer experience.

To wrap up, we've emphasized Accessibility in engineering Windows 7. We've made good progress on improving the core architecture for Accessibility and enhancing the included tools like On-Screen Keyboard and Magnifier. The *AccChecker* and *UIA Verify* tools have made it much easier to validate applications to ensure that they will be compatible with current assistive tools as well as future tools based on the Windows Automation API. Our approach to Accessibility for the features and providers in Windows itself has become more thorough, consistent and integrated, thanks to the hard work of hundreds of engineers across the company. We're proud of what we have accomplished in Windows 7 and hope that it will help customers with disabilities to realize their full potential and have a more enjoyable experience with Windows.

--Michael

# Continuing our discussion on performance

Steven Sinofsky | [2008-12-15T03:00:00+00:00](#)

---

*We've talked some about performance in this blog and recently many folks have been blogging and writing about the topic as well. We thought it would be a good time to offer some more behind the scenes views on how we have been working on and thinking about performance because it such an interesting topic for the folks reading this blog. Of course I've been using some pretty low-powered machines lately so performance is top of mind for me as well. But for fun I am writing this on my early holiday present--my new home machine is a 64-bit all-in-one desktop machine with a quad core CPU, discrete graphics, 8GB of memory, and hardware RAID all running a pretty new build of Windows 7 upgraded as soon as I finished the out of box experience. Michael Fortin and I authored this post. --Steven*

Our beta isn't even out the door yet and many are already dusting off their benchmarks and giving them a whirl. As a reminder, we are encouraging folks to hold on benchmarking our pre-production builds. Yet we've come to expect it will happen, and we realize it will lead many to conclude one thing or another, and at the same time we appreciate those of you who take the time to remind folks of the pre-ship status of the code. Nevertheless we're happy that many are seeing good results thus far. We're not yet as happy as we believe we will be when we finish the product as we continue to work on all the fundamental capabilities of Windows 7 as well as all the new features folks are excited about.

Writing about performance in this blog is nearly as tricky as measuring it. As we've seen directional statements are taken further than we might intend and at the same time there are seemingly infinite ways to measure performance and just as many ways to perceive the same data. Ultimately, performance is something each individual feels is right--whether that means adequate or stellar might vary scenario to scenario, individual to individual. Some of the mail we've received has been clear about performance:

- *Boot-very very fast in all applications ( open-load applications) especially so many simultaneously!!!! Hence, massive multicore ( quad-octa core cpu) , gpgpu for all!!!!!!!!!!!!*
- *This is right time to do this properly, the users want speed, we'll give them speed.*
- *i want to be able to run windows 7 extremely fast and still look good graphically on a asus aspire one netbook with these specs-1.5 ghz intel atom processor (single core) 1gb of ram*
- *I hope that in addition to improvements in the gui and heart (I hope massive multicore + 64-bit + Directx 11 ..extreme performance, etc) for windows 7, modified the feature Flip 3d In Windows 7!!!! Try to make a Flip 3D feature, really efficient and sensible in windows 7.*
- *With regard to the performance thing, could you look at ways to reduce the penalty of having a lot of fonts installed.*
- *From performance, boot up, explorer speed and UI experience , I hope the next version of windows delivers something new and innovating. I was playing with the new UI on the HP TouchPC and I have to say they did a great 1.0 job on the touch interface controls.*
- *I do keep my fingers crossed for Windows 7 to be dramatically better in its performance than Windows Vista.*
- *The biggest feature I see a lot of people wanting is performance.*

You can also see through some of these quotes that performance means something different to different people. As user-interface folks know, perceived performance and actual performance can often be different things. I [Steven] remember when I was writing a portion of the Windows UI for Visual C++ and when I benchmarked against Borland C++ at the time, we were definitely faster (measured by seconds). However the reviews consistently mentioned Borland as being faster and providing feedback in the form of counts of lines compiled flying by. So I coded up a line count display that flashed a lot of numbers at you while compiling (literally flashy so it looked like it couldn't keep up). In clock times it actually consumed a non-zero amount of time so we got "slower" but the reviewers then started giving us credit for being faster. So in this case slower actually got faster.

There's another story from the past that is the flip side of this which is the scrolling speed in Microsoft Word for DOS (and also Excel for Windows--same dynamic). BillG always pushed hard on visible performance in the "early" days and scrolling speed was one of those things that



never seemed to be fast enough. Well clever folks worked hard on the problem and subsequently made scrolling too fast--literally to the point that we had to slow it down so you didn't always end up going from page 1 to the end of the document just because you hold down the page down key. It is great to be fast, but sometimes there is "too much speed".

We have seen the feedback about what to turn off or adjust for better performance. In many ways what we're seeing are folks hoping to find the things that cause the performance to be less than they would like. I had an email conversation with someone recently trying to pinpoint the performance issues on a new laptop. Just by talking through it became clear the laptop was pretty "clean" (~40 processes, half the 1GB of RAM free, <5% CPU at idle, etc.) and after a few back and forths it became clear it was the internet connection (dial-up) that was actually the biggest bottleneck in the system. Many encourage us to turn off animations, graphics, or even color as there is a belief that these can be the root of performance. We've talked about the registry, disk space utilization, and even color depth as topics where folks see these as potential performance issues.

It is important to consider that performance is inherently a time/space tradeoff (computer science sense, not science fiction sense), and on laptops there is the added dimension of power consumption (or CPU utilization). Given infinite memory, of course many algorithms would be very different than the ones we use. In finite memory, performance is impacted greatly by the overall working set of a scenario. So in many cases when we talk about performance we are just as much talking about reducing the amount of memory consumed as we are talking about the clock time. Some parts of the OS are much more tunable in terms of the memory they use, which then improves the overall performance of the system (because there is less paging). Other parts of the system are much more about the number of instructions executed (because perhaps every operation goes through that code path). We work a great deal on both!

The reality of measuring and improving performance is one where we are focused at several "levels" in Windows 7: micro-benchmarks, specific scenarios, system tuning. Each of these plays a critical role in how we are engineering Windows 7 and while any single one can be measured it is not the case that one can easily conclude the performance of the system from a measurement.

**Micro-benchmarks.** Micro-benchmarks are the sort of tests that stress a specific subsystem at extreme levels. Often these are areas of the code that are hard to see the performance of during usage as they go by very fast or account for a small percentage of time during overall execution. So tests are designed to stress part of the system. Many parts of the system are subjected to micro-benchmarking such as the file system, networking, memory management, 2D and 3D graphics, etc. A good example here is the work we do to enable fast file copying. There is a lot of low level code that accounts for a (very significant) number of conditions when copying files around, and that code is most directly executed through XCOPY in a command window (or an API). Of course the majority of copy operations take place through the explorer and along with that comes a progress indicator, cancellable operation, counting up bytes to copy, etc. All of those have some cost with the benefit as well. The goal of micro-benchmarks is to enable us to best understand the best possible case and then compare it to the most usable case. Advanced folks always have access to the command line for more power, control, and flexibility. It is tempting to measure the performance of the system by looking at improvements in micro-benchmarks, but time and time again this proves to be inadequate as routine usage covers a much broader code path and time is spent in many places. For Internet Explorer 8 we did a blog post on [performance](#) that went into this type issue relative to script performance. At the other end of the spectrum we definitely understand the performance of micro-benchmarks on some subsystems will be, and should be, carefully measured --the performance of directx graphics is an area that gamers rely on for example. It is worth noting that many micro-benchmarks also depend heavily on a combination of Windows OS, hardware, and specific drivers.

**Specific scenarios.** Most people experience the performance of a PC through high level actions such as booting, standby/resume, launching common applications. These are topics we have covered in previous posts to some degree. In Engineering Windows 7, each team has focused on a set of specific scenarios that are ones we wanted to make better. This type of the work should be demonstrable without any elaborate setup or additional tools. This work often involves tuning the code path for the number of instructions executed, looking at the data allocated for the common case, or understanding all the OS APIs called (for example registry lookups). One example that comes to mind is the work that we have going on to reduce the time to reinsert a USB device. This is particularly noticeable for UFD (USB flash drives) or memory cards. Windows of course allows the whole subsystem to be plumbed by unique drivers for a specific card reader or UFD, even if most of the time they are the same we still have to account for the variety in the ecosystem. At the start of the project we looked at a full profile of the code executed when inserting a UFD and worked this scenario end-to-end. Then systematically each of the "hot spots" was worked through. Another example along these lines was playback of DVD movies which involves not only the storage subsystem but the graphics subsystem as well. The neat thing about this scenario is that you also want to optimize for the CPU utilization (which you might not even notice while playing back the movie) as that dictates the power consumption.

**System tuning.** A significant amount of performance work falls under the umbrella of system tuning. To ascertain what work we do in this area we routinely look at the overall performance of the system **relative** to the same tests on previous builds and previous releases of Windows. We're looking for things that we can do to remove operations that take a lot of time/space/power or things that have "grown" in one of those dimensions.

We have build-to-build testing we do to make sure we do not regress and of course every developer is responsible for making sure their area improves as well. We left no stone unturned in terms of investigating opportunities to improve. One of the areas many will notice immediately when looking at the pre-beta or beta of Windows 7 is the memory usage (as measured by task manager, itself a measurement that can be misunderstood) of the *desktop window manager*. For Windows 7, a substantial amount of architectural work went into reducing the amount of memory consumed by the subsystem. We did this work while also maintaining compatibility with the Windows Vista drivers. We did similar work on the desktop search engine where we reduced not just the memory footprint, but the I/O footprint as well. One of the most complex areas to work on was the improvements in the taskbar and start menu. These improvements involved substantial work on critical sections ("blocking" areas of the code), registry I/O, as well as overall code paths. The goal of this work is to make sure these UI elements are always available and feel snappy.

It is worth noting that there are broad based measures of performance as well that drive the user interface of a selection of applications. These too have their place--they are best used to compare different underlying hardware or drivers with the same version of Windows. The reason for this is that automation itself is often version dependent and because automation happens in a less than natural manner, there can be a tendency to measure these variances rather than any actual perceptible performance changes. The classic example is the code path for drawing a menu drop down--adding some instructions that might make the menu more accessible or more appealing would be impossible to perceive by a human, but an automated system that drives the menu at super human speed would see a change in "performance". In this type of situation the effect of a micro-benchmark is magnified in a manner inconsistent with actual usage patterns. This is just a word of caution on how to consider such measurements.

Given this focus across different types of measurement it is important to understand that the overall goal we have for Windows 7 is for you to experience a system that is as good as you expect it to be. The perception of performance is just as important as specific benchmarks and so we have to look to a broad set of tools as above to make sure we are operating with a complete picture of performance.

In addition to these broad strategies there are some specific tools we've put in place. One of these tools, PerfTrack, takes the role of data to the next level with regard to performance and so will play a significant role in the beta. In addition, it is worth reminding folks about the broad set of efforts that go into engineering for performance:

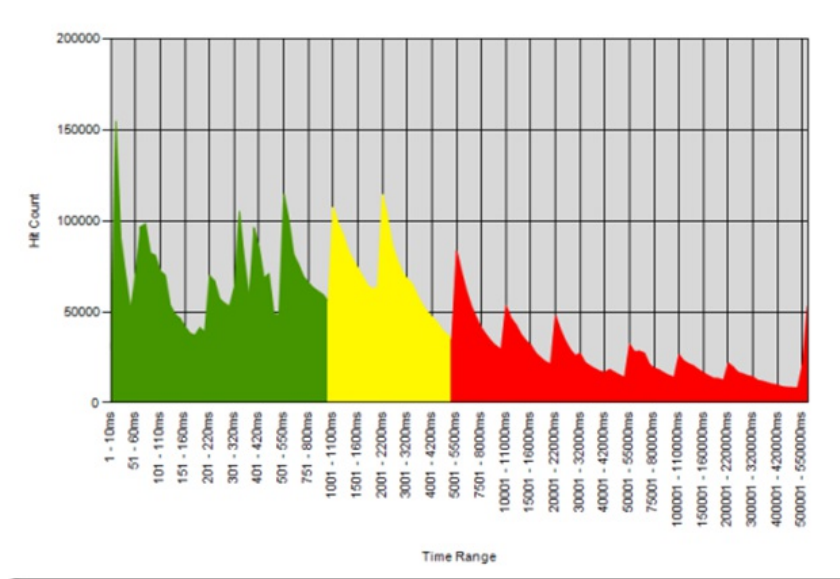
- We've been building out and maintaining a series of runs that measure thousands of little and big things. We've been running these before developer check-ins and maintaining performance and responsiveness at a level above which all that self-host our builds will find acceptable. These gates have kept the performance and responsiveness of our daily builds at a high enough level that thousands have found it possible to run their main systems on Windows 7 for extended periods of time, doing their normal daily work.
- We've been driving down footprint, reducing our service costs, improving the efficiency of key code paths, refactoring locks to improve scalability, reducing hangs, improving our I/O efficiency and much more. These are scenario driven based on real world execution paths we know from our telemetry to be common.
- We've been partnering closely with the top OEMs, ISVs and IHVs. Our tools have been made [public](#), we've held numerous training sessions, and we've been focusing heavily on shipping systems in an effort to insure customers get great performing systems out of the box, with great battery life too.
- Within the Windows dev team, we've placed a simple trace capturing tool on everyone's desktop. This desktop tool allows each person to run 24x7 with performance tracing enabled. If anything seems slow or sluggish, they can immediately save the last minute-or-so of activity and send it for automated analysis. Additionally, a team of people visually inspect the traces for new issues or issues not yet decipherable by our automation. The traces are incredibly rich and allow us to get to the root of top issues most of the time.
- For all Pre-Beta, Beta and RTM users, we've developed a new form of instrumentation and have used it to instrument over 500 locations in the operating system and inbox applications. This new instrumentation is simple in concept, but revolutionary in result. The tool is called PerfTrack, and it has helped confirm our belief that the client benchmarks aren't too informative about real user responsiveness issues.

Perfrack is a very flexible, low overhead, dynamically configurable telemetry system. For key scenarios throughout Windows 7, there exist "Start" and "Stop" events that bracket the scenario. Scenarios can be pretty much anything, including common things like opening a file, browsing to a web page, opening the control panel, searching for a document, or booting the computer. Again, there are over 500 instrumented scenarios in Windows 7 for Beta.

Obviously, the time between the Stop and Start events is meant to represent the responsiveness of the scenario and clearly we're using our telemetry infrastructure to send these metrics back to us for analysis. Perfrack's uniqueness comes not just from what it measure but from the

ability to go beyond just observing the occurrence of problematic response times. Perftrack allows us to “dial up” requests for more information, in the form of traces.

Let’s consider the distribution below and, for fun, let’s pretend the scenario is *opening XYZ*. For this scenario, the feature team chose to set some goals for responsiveness. With their chosen goals, green depicts times they considered acceptable, yellow represents times they deemed marginal, and red denotes the poor times. The times are in milliseconds and shown along the X axis. The Hit Count is shown on the Y axis.



As can be seen, there are many instances where this scenario took more than 5 seconds to complete. With this kind of a distribution, the performance team would recommend that we “dial up” a request for 100+ traces from systems that have experienced a lengthy open in the past. In our “dialed up” request, we would set a “threshold” time that we thought was interesting. Additionally, we may opt to filter on machines with a certain amount of RAM, a certain class of processor, the presence of specific driver, or any number of other things. Clients meeting the criteria would then, upon hitting the “Start” event, configure and enable tracing quickly and potentially send back to us if the “Stop” event occurred after our specified “threshold” of time.

As you might imagine, a good deal of engineering work went into making this end to end telemetry and feedback system work. Teams all across the Windows division have contributed to make this system a reality and I can assure you we’ll never approach performance the same now that we have these capabilities.

As a result of focusing on traces and fixing the very real issue revealed by them, we’ve seen significant improvements in actual responsiveness and have received numerous accolades on Windows 7. Additionally, I’d like to point out that these traces have served to further confirm what we’ve long believed to be the case.

This post provides an overview of the ways we have thought about performance with some specifics about how we measure it throughout the engineering of Windows 7. We believe that throughout the beta we will continue to have great telemetry to help make sure we are achieving our goals and that people perceive Windows 7 to perform well relative to their expectations.

We know many folks will continue to use stop watches, micro-benchmarks, or to drive automated tests. These each have their place in your own analysis and also in our engineering. We thought given all the interest we would talk more about how we measure things and how we’re engineering the product.

--Steven and Michael

# At Home with HomeGroup in Windows 7

Steven Sinofsky | [2008-12-30T03:00:00+00:00](http://www.microsoft.com/windows/windows-7)

---

*Like many places we've spent the past few weeks under quite a bit of snow, which is pretty unusual for Seattle! Most of us on the team took advantage of the snow time to install test builds of Windows 7 on our home machines as we finalize the beta for early 2009—I know I felt like I installed it on 7000 different machines. We're definitely looking forward to seeing folks kick the tires on the beta when it is available. For more information on the beta, please stay tuned to <http://www.microsoft.com/windows/windows-7> which is where we will post information about participation.*

*This post is about a Windows 7 feature that covers a lot of territory—it is about networking, user interface, sharing, media, printing, storage, search, and more. HomeGroup is a way of bringing all these features together in a way that makes it possible for a new level of coolness in a home with multiple PCs running Windows 7. A lot of us are the sysadmins for our own homes and for many others (friends and family). We set up network topologies, configure machines, and set things up so they work—HomeGroup is designed to make that easier so it can be done without a volunteer sysadmin. It makes for some challenges in how to describe the feature since the lack of such a feature has each of us creating our own private best practices or our own techniques for creating and maintaining a home network. HomeGroup is about making this easier (or possible for everyone else) and at the same time giving you the tools to customize and manage—and no matter what, under the hood the file and printer sharing, media sharing, and networking you are already familiar with is there should you wish to stick with the familiar ways. HomeGroup is a deep feature that builds on a lot of new infrastructure/plumbing new to Windows 7, though in this post we'll talk about it from the experience of setting up a network.*

*This is a feature that is one you should just use and see it working, rather than trying to read about it as it covers so much territory in writing.*

*This post is by Jerry Koh a lead program manager in the Core User Experience team, with help from a number of folks across the dev team. –Steven*

*PS: From all of us on the Windows team, we wish you a very Happy New Year!*

You probably have seen or heard about HomeGroup by now. We demonstrated it at PDC this year during Steven's keynote, it was mentioned a few times at WinHec, and some of you may have even tried it on your PCs with the PDC pre-beta build of Windows 7. HomeGroup represents a new end-to-end approach to sharing in the home, an area in which Windows has provided many features before --- the intuitive end to end is what's new. HomeGroup recognizes and groups your Windows 7 PCs in a "simple to set up" secure group that enables open access to media and digital memories in your home. With HomeGroup, you can share files in the home, stream music to your XBOX 360 or other devices, and print to the home printer without worrying about technical setup or even understanding how it all works.

This blog post is designed to give you a behind-the-scenes look at how we designed HomeGroup.

## Designed with you in mind

The HomeGroup design goal, like other Windows 7 features, is informed by customer data and input. Whether from the Customer Experience Improvement Program (CEIP), the Windows Feedback Panel, focus groups or usability sessions, the data we collect enables us to focus on key areas where people feel the most pain. To begin figuring out how to solve file and printer sharing problems in the home, we started by looking at how people interact within a home environment. We wanted to learn not only how people used computers in the home, but also what social and behavioral norms were acceptable to see if there were parallels that we could bring into our design. We found the following:

- People don't allow strangers into their homes and usually lock their exterior doors. People within the confines of the home are typically considered to be trusted.
- Within the home, doors to rooms are usually not locked, allowing members of the household to have free access. Books, photographs, magazines, CDs, and DVDs are often freely shared.
- Social norms prevent most people from snooping into areas where they shouldn't and, if needed, adding locks to rooms or drawers is relatively easy.

The social model of the home also reflected how people want to share. When we discussed file and printer sharing in the home (or the concept of doing so), we found that people classify their content generally into four different buckets: private, public, parentally sensitive, and children's stuff. Private content consists of business and financial data and is considered private mainly because people fear it will be accidentally deleted as the number of people who have access to it increases.

People are typically quick to point out that they don't have entertainment content they consider private, and they're very open to free access to this content within the family. Families with children are often concerned about parentally sensitive content (inappropriate music, videos, etc.). With digital cameras and camcorders dropping in price and being widely adopted, parents are primarily concerned about accidental deletion or loss of original copies of digital memories.

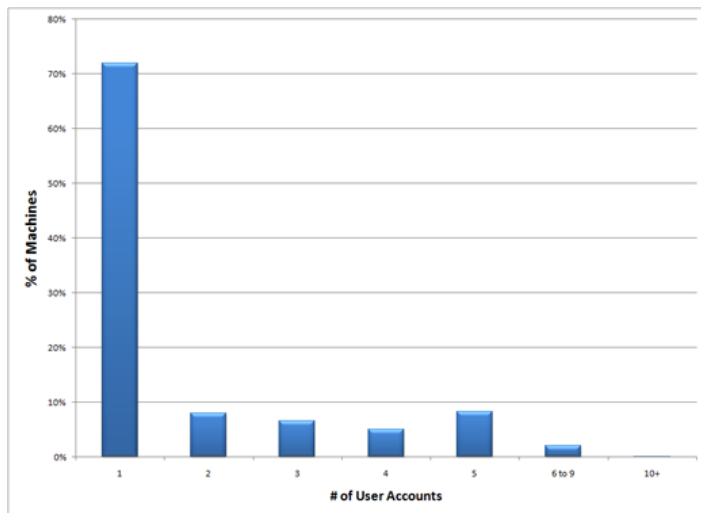
These observations were very interesting to us; a model that mirrored real-world expectations for sharing could be more natural to people than something that layered different questions around security, permissions or rights. So we approached the HomeGroup sharing model with the concept of open access in the home. But, how can we define what the "home" really is? What assumptions can we make about security?

## Wireless, user passwords and when are you “at home”?

One of the key advances we’ve had in home networking technology has been wireless. Standards like 802.11 have taken the home network by storm. Wireless router sales to consumers are higher than ever, and are projected to continue growing. As a wider segment of people buy wireless routers, concerns about security start to build up. When configured incorrectly, wireless networks can leave your entire home network vulnerable to malicious people or nosy neighbors. While there have been efforts to help people become more aware of securing wireless networks -- such as the “Windows Rally program” and various “Windows Connect Now” technologies--the general public still lags behind in setting up security for their wireless networks. We know from our customer data that more than half of all wireless networks, whether by choice or oversight are set up as unsecured and we know many of you are the first line of defense in helping your friends and families set up a secure home network. While trends all point to more awareness and improvement in the future, it isn’t clear whether we would ever reach 100% security on these networks. So how can we make sure home networks are secured?

Another interesting factor is the usage of passwords on user accounts in the home. While people are more sensitive to security than ever before, we also observed that many don’t want to set up passwords for their Windows user accounts. They feel that it is a barrier to their use of the computer and yet another thing for them to remember or lose (as an aside, passwords are often viewed as a performance bottleneck in the home). From the data we obtained from the Windows Feedback Panel, a majority of users actually don’t use passwords in the home, opting for the simple model of opening the laptop lid and using Windows quickly. This parallels usage patterns on cell phones, where setting passwords on them would just be a deal-breaker for most people.

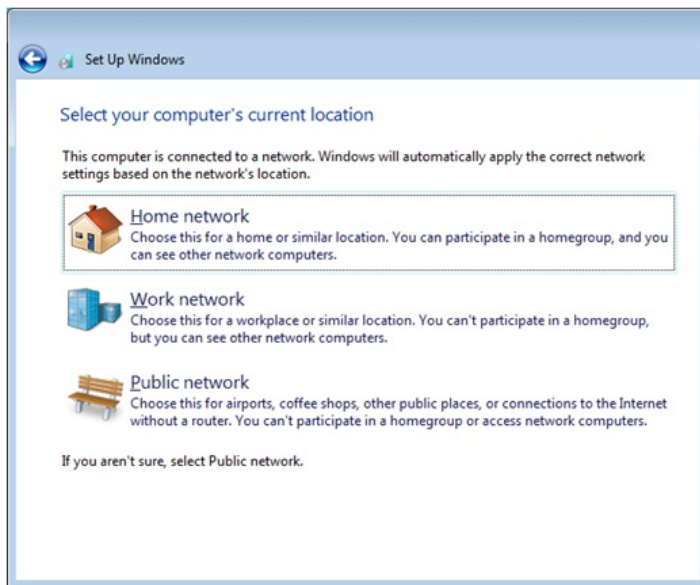
A majority of the computers in our panel only had one primary user. While we all know that laptop sales have overtaken desktop sales in the last couple of years, this data tells us that people are buying PCs more for specific people rather than for a shared location. With laptops, the mobility factor has contributed to the “one person/one computer” landscape, again mirroring cell phone ownership patterns in which users almost never share a personal cell phone. Clearly as notebook options include even less expensive options, this will only increase, though we recognize it is still rather a luxury in most of the world.



So we wanted to find a model that could secure home sharing for people who don’t use passwords and could also take into account the more personal nature of PC usage.

First we needed to figure out that people were “at home”. Luckily we didn’t need to look very far for some useful technology in this area. Windows Vista introduced a concept known as *network location awareness* (NLA). This enables the system to recognize when you’ve changed network locations, and it tags the location with a simple “Home”, “Work” or “Public” designation. While it was somewhat of a mystery in Vista in terms of what such a designation did (unless you read all the words), we will see the infrastructure has become increasingly important as we built out the HomeGroup scenario. In addition to ensuring the right firewall settings are configured for these locations, NLA also enabled us to be smarter about starting Windows services that are targeted at specific network locations. For example, the network discovery service does not start if you’re in a public location. However, Windows Vista didn’t have much distinction between the “work” and “home” network locations; they were essentially the same in terms of which firewall ports were opened and which Windows services were started.

In Windows 7, we extended the concept of NLA and made “work” and “home” more distinct. In Windows 7, when you select the “home” network profile, we know that you are “at home”, and will start the essential services required for successful file and printer sharing in the home. This provides an intuitive entry point into HomeGroup, and once you are “at home” we start looking for (via network discovery) other Win7 PCs in the home. If you already have a HomeGroup active, we offer you the ability to join it; if not, you can create one.



Now that we know your PC is at home, we need to make sure that your data is secured from prying eyes. While wireless security is full of acronyms and technical solutions to security (WEP, WPA, TKIP, etc., to name a few), the fundamental model of wireless security is fairly simple for people to understand. The use of a physical key (copied several times) to enter one's home is mirrored by the concept of typing in a shared key to gain access to the home wireless network. In the HomeGroup case, Windows will provide you with a pre-generated password out of the box, which you would hand over to any member of the home, and they could then join the group.



While a password is provided by default, people can, at any time, visit HomeGroup in Control Panel to change their password to something they prefer. This flexible system performed very well in testing. When faced with the default password, people wrote it down, and shared it with others to set up the HomeGroup. You may ask, why don't we enable people to set their own passwords by default? The answer is actually quite ironic, since that was our initial design. In testing, this concept raised quite a bit of alarm with people. It seems that most people generally have 1 or 2 passwords that they use for all their online or offline activities. When asked to input a user password for their HomeGroup, they gravitated towards using one of those, and then reacted with alarm when they realized that this password needs to be shared with other users in the home! People generally reacted better to the auto-generated password, since they knew to write it down and hand it around. The other interesting benefit we got from this was a reduction in the amount of time people would spend on the UI that introduced them to the HomeGroup concept. With a user-generated password, they had to grasp the HomeGroup concept, think about what password to set, and decide whether to accept the shared libraries default. Without having to provide a password, people had more time to understand HomeGroup, and their sharing decision – leading to a much more streamlined, private, and secure design.

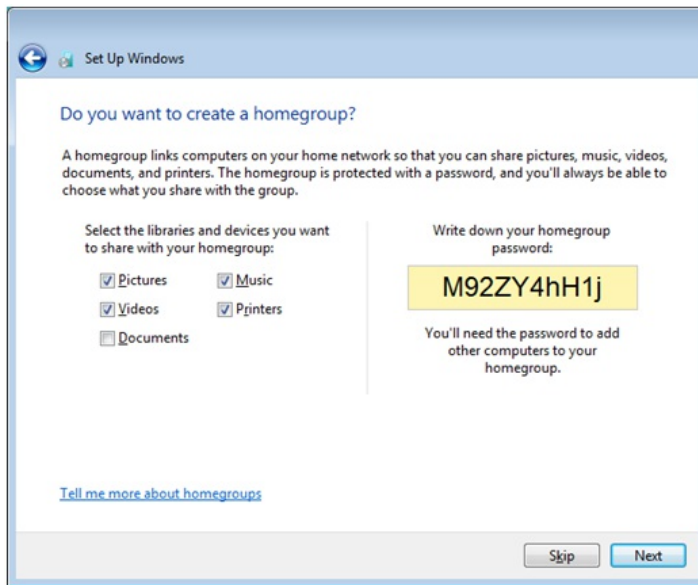
### ***A home of equals with open access to libraries***

In addition to balancing security with ease of use, we also wanted to account for PCs becoming more personal. For this reason, we adopted the concept that each person in the HomeGroup is a peer of the others. Each person can thus join and leave the HomeGroup as they wish. Each person brings with them their choice of media/memories or files to share with the rest of the home. With a system based on equals and peers, the big benefit is a lack of management overhead; you don't need one person to bear the management task of maintaining the group and dealing with membership tasks. This eliminates a primary source of complexity. All you need to gain entry is the shared password (just like the house key that each family member has).

With a home full of equals, what would they share? As mentioned above, our customers indicated a desire to share media, both music and photos, they want to quickly and easily access within the home. So that is exactly what we implemented. HomeGroup will enable sharing the pictures,

music, and video libraries from your Windows 7 PC by default. Another blog post will go into more detail on how libraries work, but in a nutshell, they provide Windows with a way to aggregate multiple physical locations on a computer into one unified view. This is a very powerful addition to the way you organize your data in Windows. Your Pictures library can now contain your <username>\pictures folder, the Public\pictures folder, as well as the f:\foo folder that contains other pictures (and perhaps is on a USB external hard drive). Viewing your picture library locally gives you a unified view of all the pictures in these locations and enables you to search, sort, and organize them in the same way you would within a folder, while also making sure you save new items to the right place physically.

In addition to media, some people might want to share their documents. We enable you to do this when you create or join a HomeGroup. This is great for people who want to collaborate with their family or in families where open access to documents is not a concern. The content is shared as “read-only” and can be selectively changed in Windows Explorer. We want the system to work the way you expect it to, with enough flexibility to do whatever you want later.



## *Easy to use*

Now that we have made it easy to set up, the next step is to make it easy to use. There were two aspects here that we want to emphasize for this post:

1. Discovery of what is shared to me
2. Access and usage of content that is shared to me

In Windows Vista, this discovery was done through the *network folder*, which provides a complete, but highly technical, view of the resources available to you on the network. In addition, the network folder also contains other devices and additional media libraries that were shared on the network. This was confusing and difficult to understand for typical people. For example, if you shared your Pictures folder, it was actually found under the computer in \\<computername>\users\<username>\pictures. Typically, people would not know to look into that path for the correct folder.

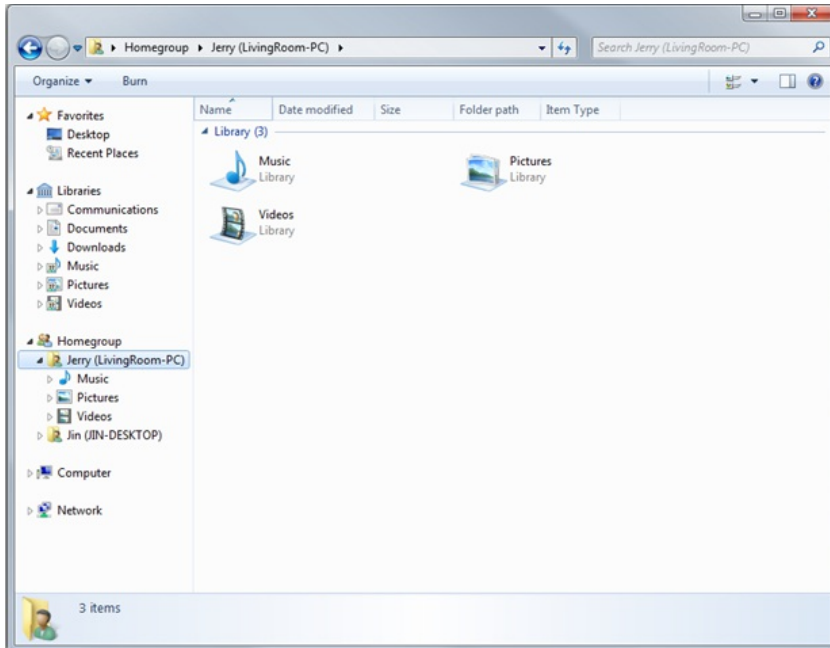
The concept of “libraries” introduced in Windows 7 gives us the design point to improve access their content across the network. While libraries aggregated the view on a local computer, if these locations were shared out to the network, they resulted in a more complicated view in the network folder for our users. Each location would be shared as a separate path, so taking the example above, sharing out the Pictures library means that you’ll see three shares under \\computername, Users\<username>\pictures, Users\public\pictures and foo. People would not benefit from the power of libraries on a network. Therefore, we use the concept of libraries to work well even across a home network. We did this in two ways.

First, people should have the same experience viewing a library whether on a local computer or across the network in a HomeGroup. We made sure that when you share the Pictures library in Windows 7, not only are all locations of the library shared, but the library resource is also shared and can be consumed by other computers in the HomeGroup. Effectively, members in a HomeGroup would see just one unified library with its aggregated views.

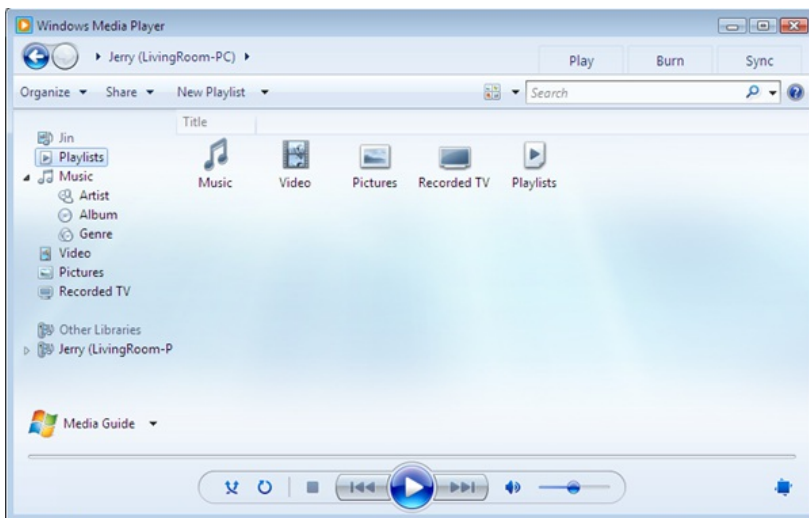
Second, we found that accessing these resources in the network folder was too many clicks away and sufficiently buried such that people would find it impossible to discover. So we created a new HomeGroup node on the navigation pane in Windows Explorer. When you join a HomeGroup, other HomeGroup Win7 PCs will appear under the HomeGroup node in the Windows Explorer navigation pane. They’re one click away and always at your fingertips. In our tests, this really opened up discovery and usage of content throughout the HomeGroup. People easily discovered music on another computer, played it back, or looked at photos. Consumption of media thus becomes something easy and habit-



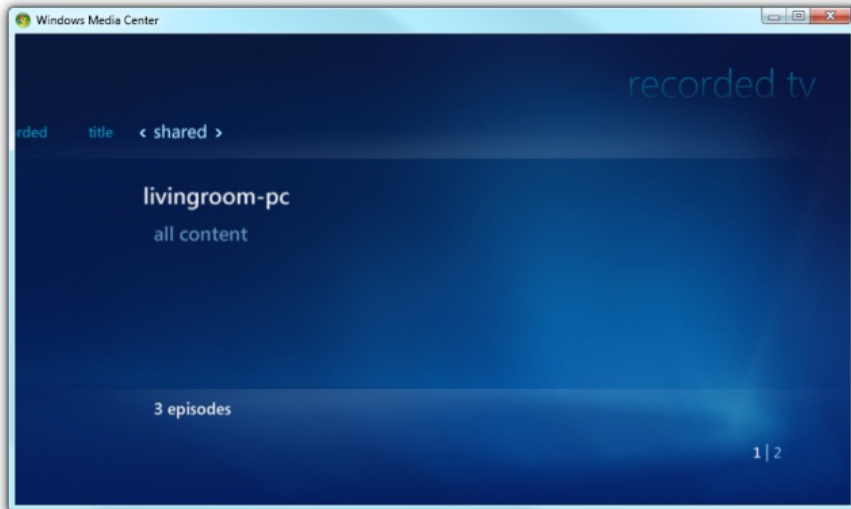
forming in the home, all by joining a HomeGroup.



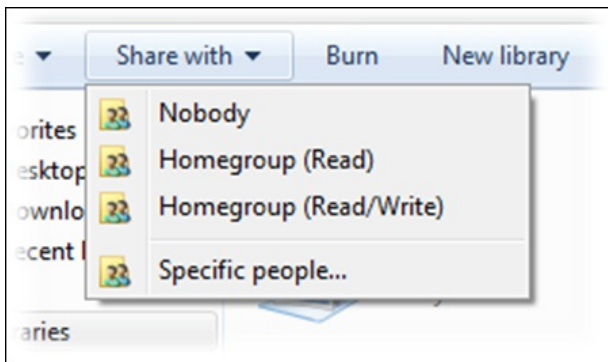
With the introduction of libraries, we also had an opportunity to remove some of the confusion between specialized media libraries that are created by Windows Media Player (WMP) or Windows Media Center (WMC). In previous versions of Windows, WMP would scan the entire hard drive on the computer to find media files and add them into a media library, but in Windows 7 this no longer has to happen. Since you already have Windows Explorer libraries, WMP and MCE just use those. If you add new locations to the libraries in Windows Explorer, WMP and MCE now automatically just pick them up since they are using the same common library for the content. We thus eliminated the need for people to manage multiple views of their data using different user experiences. In addition, WMP will also show the media libraries shared by the HomeGroup as nodes in the WMP navigation pane, mirroring the discovery and access model of Windows Explorer. So the same set of HomeGroup users you see in Explorer by default will also be shown to you in WMP as well.



Similar to WMP, in WMC, there is a new “shared” section when browsing media like recorded TV, pictures, music and video. HomeGroup computers show up in this section and can be accessed easily. The content of those libraries that have been shared with the HomeGroup will show up and be accessible in WMC. This includes music, pictures and videos, but also recorded TV--which means that you can now browse and stream non-DRM TV (that was recorded on another computer in your home) from your laptop!



In addition to sharing out your media by default, we also wanted to make sharing additional content to the HomeGroup simple. In the past you had to worry about setting access control, as well as managing user passwords to make sharing work in the home. As we better understood how people interacted and worked at home, we realized that most were OK with enabling general access to all members of the household. So we built a few shortcuts into the sharing experience to enable this. Windows Explorer now features a new “share with” menu in the command bar:



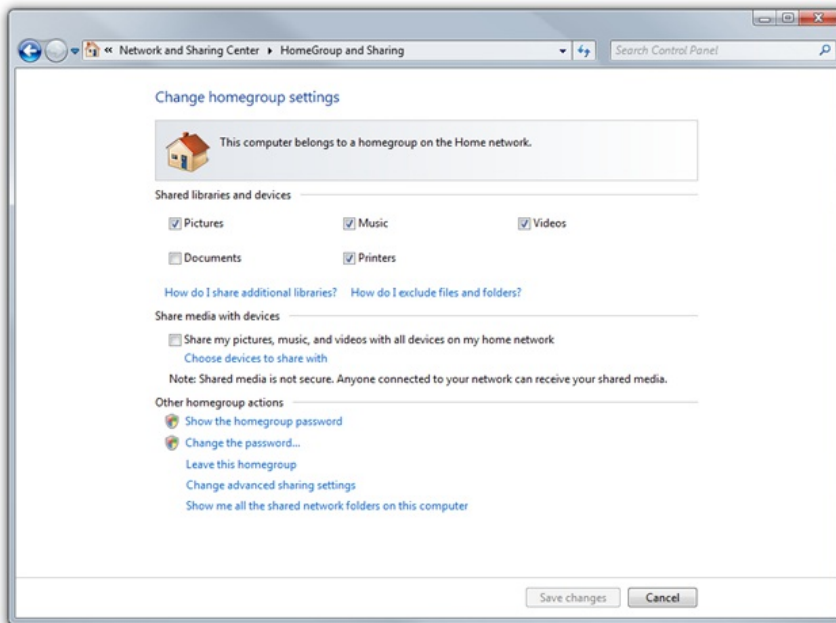
This enables you to select a library or folder and quickly share it with the home. It even enables you to make content writable by home members with one click, thus making it easy for people at home to easily collaborate on pictures or documents. This enables scenarios like importing digital photographs on one computer and editing them on another computer without making a copy. Once you share a folder with the home, it also shows up under the user in the HomeGroup node. This makes it incredibly easy to share anything on your computer to others in the home, and have them

easily find and use them. We also recognize that some people need a way to easily bring some of this content off the network quickly and easily and make it private. The “share with” menu includes a shortcut to “share with nobody.” This option removes access to any content that has been previously shared and makes it private, thus enabling us to deliver on another requirement we observed people have in the home.

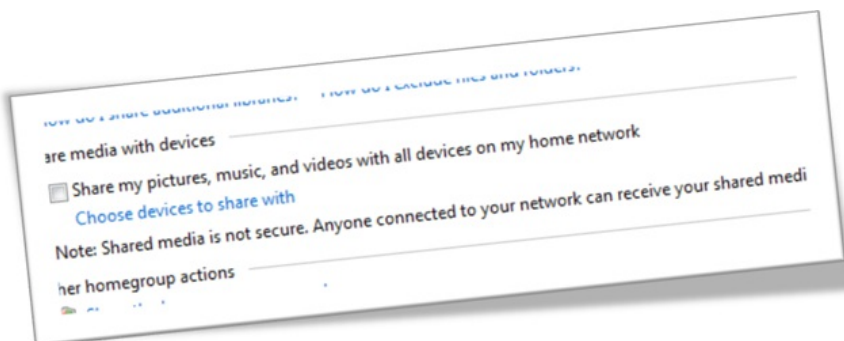
## ***Printers and other devices work with HomeGroup as well***

So what about devices? We’ve heard from you that sharing printers needs to be much simpler. While we have made it super easy to add printers to Windows, we needed to bring this simplicity to the home network. USB printers are still tied to a specific PC and can’t be shared out very easily. People typically email files to themselves to retrieve on another computer, or use USB keys to move their files to the computer with the printer. That had to change.

In a HomeGroup, if you have installed a USB printer that has a Windows logo, the other people on the HomeGroup would get this printer automatically installed on their computers. They won’t see a prompt, they won’t need to answer any questions – it would just show up, and “just work.” For non-Windows logoed printers, we need to ask the user for permission to install the printer. HomeGroup members will see a prompt that a printer has been found in the HomeGroup. Clicking on this prompt installs the driver. The reason we had to do this was to ensure that users consent to 3<sup>rd</sup> party code that hasn’t been through the rigors of the logo program. One of the big benefits of this system is that you no longer need to find, download, and install the driver manually on multiple computers. The driver (for the correct architecture) is just copied from the computer that has the physical printer attached. This saves time and network bandwidth. With a HomeGroup, there will no longer be a need to think about sharing a printer. If you attach one to a computer in the HomeGroup, everyone else will get it installed and ready to use.



In addition to printers, devices like photo frames, game consoles (such as the Xbox 360), and media receivers (like the Roku Soundbridge) can benefit from some of the easy setup, as well as all the shared media in the home. For setup, we have reduced all the UI within Windows that deals with these devices to one simple checkbox:



Once you are part of a HomeGroup, we turn on Windows Media Player streaming support, so not only will your computer detect other WMP libraries on the network and allow playback from them, devices would also be able to consume the shared media content. Another blog post will go into more detail on an exciting new feature called “play to” which would also be automatically enabled in a HomeGroup enabling you to send

media from your PC to any supported picture frame or media receiver, and never have to deal with the minimal UI you have on these devices, which you can see in the demonstration of the Day 1 keynote at WinHEC. If you check a box in HomeGroup in Control Panel, all existing and future devices in the home will detect and consume the media on the HomeGroup computer. All these previously complicated settings are now simplified with HomeGroup.

### ***Domain-joined computers can be part of a HomeGroup***

The laptop buying trend doesn't stop at home. Large corporations are also moving toward buying laptops for their employees. There is research out there that outlines productivity improvements with employees using laptops. This makes sense as most of these laptop-wielding employees bring their computers home and put in those extra email hours. However, most corporations require that their laptops be joined to a corporate domain. This enables system administrators to manage and maintain these computers. Domain-joined laptops are thus subject to more restrictions than regular home computers are. It's hard to even locate another PC on the home network to access or share files, let alone configure your domain-joined computer to print to a printer at home.

With HomeGroup, we wanted see if we could make things a little easier for these computers to come home. With more and more people working from home or having the option to these days, we wanted to see if they could enjoy some of the media content they have on the other PCs in the HomeGroup while they work. So in Windows 7, your domain-joined computer can join and participate in a HomeGroup. This enables the domain-joined computer to consume the media available on Windows 7 PCs in the home, watch TV through WMC, listen to music via WMP, or print to the printer on another HomeGroup PC all by entering the same key you provide to other computers in the HomeGroup.

The only difference is that sensitive content on the corporate laptop is never shared to the other HomeGroup computers. In essence, the domain-joined computer can see out (and consume) but no one can see in. We believe this meets the need for corporations to maintain security over documents while enabling our customers to enjoy a fun and interesting work environment at home, with access to all their media and home printers while they work. All you need is an existing HomeGroup, a domain-joined computer, and you can be rocking to your favorite tunes on your home network, while you catch up on all your important work.

Of course the ability to join a HomeGroup is a policy that can be managed by corporate domains as you would expect.

### ***Create a HomeGroup with the Beta***

Phew! I hope this post has given you some insight into some of our design decisions, as well as the capabilities of the feature. HomeGroup will highlight some of the cool capabilities Windows has had for a long time in a friendly and easy fashion and also build on some of the new plumbing and infrastructure in Windows 7, and we are very excited with its possibilities. It is important to note that none of this would be possible without the help of people around the world who have provided us with opportunities to listen to their feedback, observe their actions, and take note of their needs.

We know there will be lots of discussion around this feature once folks have had a chance to explore it. It represents a new model for something that has arguably been very difficult to set up and so for most people seeing all this work will be a first and for many of us reading this blog we'll be "mapping" our existing model to this new experience. The best thing to do is just see if you can let Windows 7 run and do the work. After some use you can then dive into the customization and configuration available to you.

To set up a HomeGroup you will need to install Windows 7 Beta on more than one PC on the same network and be sure to select Home as the network location if you want to automatically create (or join) a HomeGroup.

Thanks,

Jerry

# Windows 7 Energy Efficiency

Steven Sinofsky | [2009-01-06T03:00:00+00:00](#)

---

*Happy New Year! The following post continues our discussion of fundamentals with a focus on power management. Power Management (or energy efficiency) is something that every contributor to the PC Ecosystem must always address—the energy efficiency of a running PC is limited by the weakest component. In engineering Windows 7 we had an explicit focus on the energy usage patterns of the running system and will continue to work with hardware and software makers to realize the collective benefit of all of this work. While we talk about the balancing of needs in every area, energy consumption is probably the most easily visualized—when we test running systems we connect them to power meters and watch a very clear number change as we run tests. (If you’ve seen the film [Apollo 13](#) then you’ve seen a similar (albeit much more mission critical) struggle with a power budget.) This post is by Dean DeWhitt in program management team on our Kernel team. --Steven PS: Quite a few of us are at CES this week!*

Energy efficiency is one of the most active topics in modern computing today. As evidence, consider that processor and chipset vendors are marketing products on “performance per watt”, instead of just processor clock frequency and benchmark performance. Perhaps you have seen a press release for one of the many industry consortiums focused on “Green Computing”—reducing the power consumption and environmental impact of computing. Finally, battery life continues to be a major purchasing and usability factor for mobile PCs. These related energy efficiency efforts in the PC industry result in an ever-increasing interest in how Windows manages power.

In engineering Windows 7, our goal is to deliver the capabilities and features users want from a Windows PC while reducing power consumption over previous releases. Windows already provides a rich set of energy saving features, including the ability to turn off the display and automatically put the system to sleep when the user is not interacting with the computer. For Windows 7, we are building upon the investments in these areas by extending the existing capabilities and focusing on reducing power consumption when the system is idle. Although Windows is responsible for managing the power state of many devices, including the processor, hard drive and display, the remaining devices and software running on the computer have just as much (if not more) impact on power consumption and battery life. This is a challenge for everyone contributing to the PC experience.

When we talk about energy efficiency and power consumption, we like to break down the problem area into 3 main components:

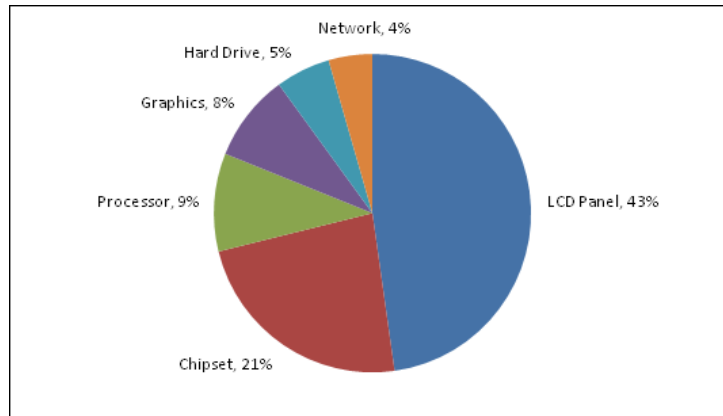
- **Base Hardware Platform:** The processor, chipset and memory and in the case of mobile platforms this also includes the battery capacity. The base hardware platform can have a significant impact on the power consumption of the platform—maximum processor speed, the number of cores, if the processor is designed for mobile devices, and the amount of RAM are all factors.
- **Windows:** The PC operating system is responsible for managing many of the devices in the system, making smart tradeoffs between performance and power consumption based on usage and allowing the end-user to dictate power management policy through power plans and settings. The challenges in this area are to properly manage device power and to ensure new Windows features are as efficient as possible in the amount of system resources (CPU, memory and disk) they use.
- **Extensions:** Extensions is a general category which includes other devices, drivers, services and applications. Devices, drivers and other software can have a significant impact on power consumption and a single application can impact battery life by 20% or more.

Realizing great energy efficiency from a Windows PC requires efforts in each of these areas. A problem with any single component in any area can have a significant impact on power consumption. Thus approaching energy efficiency from a platform approach and paying special attention to each component on the platform is required.

## Base Hardware Platform

The base hardware platform is really dictated by the system manufacturer. The customer gets the ultimate choice when they buy a system—the customer can buy a system with ultra-efficient hardware components or can buy a system with components that favor performance over power consumption. There are desktop and mobile PCs in all kinds of form factors, with varying capabilities and power consumption levels. Some mobile PCs have a normal 3 or 6-cell battery, while others have an extended 9-cell battery or another external battery that can be added to the computer.

The challenge for Windows is to be energy efficient across the wide range of hardware in the Windows ecosystem. Looking at a modern laptop, here is where the power goes:

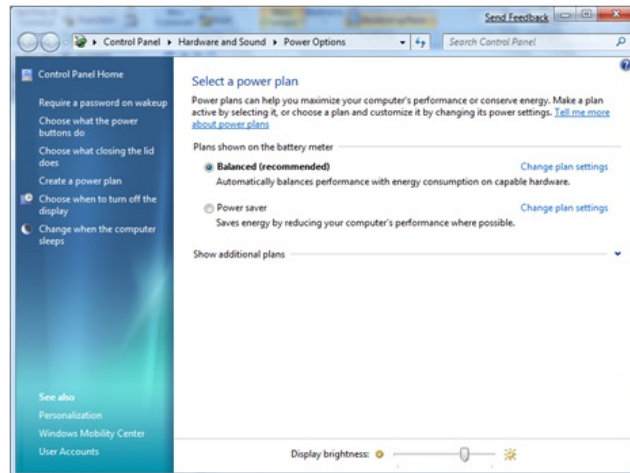


Desktops will have a similar power distribution although higher in watts. The display is a large amount of the energy consumed in using your desktop PC as well.

## Operating System

The Windows operating system can have just as big an influence as any other component in the platform. In engineering Windows 7 our goal is to make sure Windows provides a great foundation and energy saving opportunities within the operating system starting with configuration of power policy settings.

The first place most users encounter Windows power management is through Power Options in control panel, or the battery meter on a mobile PC. For as long as Windows has had power management, Windows has had power schemes or power plans. The power plans allow you to easily change from one set of power settings to another, depending on your preferences.



Within a power plan, you can change a variety of Windows power-saving features, including inactivity timers for turning off the display, automatically putting the system to sleep or even creating a new custom power plan for the exact settings you want. The display and sleep idle features are very important for power savings and battery life. As above, the display can consume approximately 40% of the power budget on the typical mobile PC and anywhere from 30-100+ Watts on a desktop PC.

PC OEMs, especially makers of laptops, will often develop a custom set of power schemes that work to take advantage of differentiated hardware and unique software available on a specific model. So often you will see power schemes that carry the name of your PC OEM in the title. These have been developed by the OEM who is just as committed to energy efficiency.

*Quick tips:* The easiest way to save power on a desktop PC is reduce the display idle timeout to something very aggressive, such as 2 or 5 minutes. If you have a screen saver enabled, disable it to allow the display to turn off. On a mobile PC, the easiest way to extend battery life is to reduce the brightness of the display in addition. Also note that many of the new all-in-one machines use laptop components and thus from a power management perspective look like laptops.

Windows manages the processor performance and changes it dynamically based on the current usage to provide performance boost when required and conserve power based on the current workload. For example, when the system is mostly idle, such as when I'm typing this blog post, there is no need to be running the processor in the maximum performance mode, instead the processor voltage and frequency can be reduced to a lower value to save power. Similarly, the hard disk drive and a variety of other devices can be placed in low-power modes or turned off completely to save power when not in use.

For Windows 7, we're refining the user experiences for power management, focusing on reducing idle power consumption and supporting new device power modes.

There are two reasons to optimize idle power consumption on the system. First there are various times throughout the day when the PC is idle and the more the system gets to idle and stays idle, the less power it uses. Second, idle power consumption is the 'base' power consumption for all other workloads. A system which consumes 15W at Idle will consume additional power over the idle power consumption while in use for other workloads. By reducing the idle power consumption on the platform we will improve most other scenarios as well.

The first step in reducing idle power is optimizing the amount of processor, memory and disk utilization. Reducing processor utilization is the most important, because the processor has a wide range of power consumption. When truly idle, the processor power consumption can be as low as 100-300mW. But, when fully busy, the processor can consume up to 35W. This large range means that even small amounts of processor activity can have a significant impact on overall power consumption and battery life. There are several areas of investment in Windows 7 that help reduce processor utilization and thereby enabling longer periods of time where the processor can enter into low power modes. One of these investments is in the area of services that are running on the platform and having those services only start when they are required referred to as "Trigger-Start". While these services are efficient and have minimal impact by themselves, the additive effect of several services can add up. We are looking at smart ways to manage these services both within Windows but making our investments in this area extensible for others who are writing services to take advantage of this infrastructure. (Also note these are the same features that contribute to improvements in boot time as well).

To further help reduce idle power, we are focusing on core processor power management improvements. Windows scales processor performance based on the current amount of utilization, and making sure Windows only increases processor performance when absolutely required can have a big impact on power consumption.

We have made several investments in the area of device power management including enhancements to USB device classes to enable selective suspend across a broad range of devices including audio, biometrics, scanners, and smart cards. These investments available in Windows 7 enable more energy efficient PC designs. We have also invested in improvements to power management for networking devices, both wired and wireless.

While many of our investments in the core infrastructure improves energy efficiency across several scenarios, in Windows 7 we also focused on several key customer scenarios to identify resource utilization improvements to extend battery life on mobile platforms. One of these scenarios that we identified was media playback. The optimizations for DVD playback include reducing processor and graphics utilization, audio improvements, and optical disk drive enhancements. These improvements are already paying off and showing significant increase in battery life across a broad range of mobile platforms which we demonstrated at the WinHEC conference.

## **Extensions**

Graphics devices, USB devices, device drivers, background services and installed applications are all extensions to Windows. Large improvements in power consumption and energy efficiency can be realized by improving the efficiency of platform extensions.

For example, consider a single USB device that does not support Selective Suspend. That USB device itself may have very low power consumption (e.g., a fingerprint reader), but until that device enters the suspend state, the processor and chipset must poll the device at a very high frequency to see if there is new data. That polling prevents the processor from entering low power idle states, and on a typical business-class notebook reduces battery life by 20-25%.

Devices are not the only area that require efforts for great energy efficiency. Application and service software can also have a big impact on power consumption. Take for example an application that increases the platform timer resolution using the `timeBeginPeriod` API. The platform timer tick resolution will be increased and the processor will not be able to efficiently use low power idle modes. We have observed a single application that keeps the timer resolution increased to 1ms can have up to a 10% impact on battery life on a typical notebook PC.

We're committed to helping improve the energy efficiency of Windows platform extensions by working closely with our partners. The strategy we're employing is to provide rich tools to identify energy efficiency problems in hardware and software. For Windows 7, we've added a new inbox utility that provides an HTML report of energy efficiency issues—a "Top 10" checklist of power problems. If you want to try it out on Windows 7, run `powercfg /energy` at an elevated command prompt. Be sure to close any open applications and documents before running `powercfg`—this utility is designed to find energy efficiency problems when the system is idle. `powercfg` with the `/energy` parameter can detect USB devices that are not suspending and applications that have increased the platform timer resolution.

For more advanced analysis, we have provided the Windows Performance Toolkit. The Performance Toolkit <http://www.microsoft.com/whdc/system/sysperf/perftools.mspx> makes it very easy for software developers to observe the resource utilization of their applications, resolve performance bottlenecks and identify issues impacting energy efficiency.

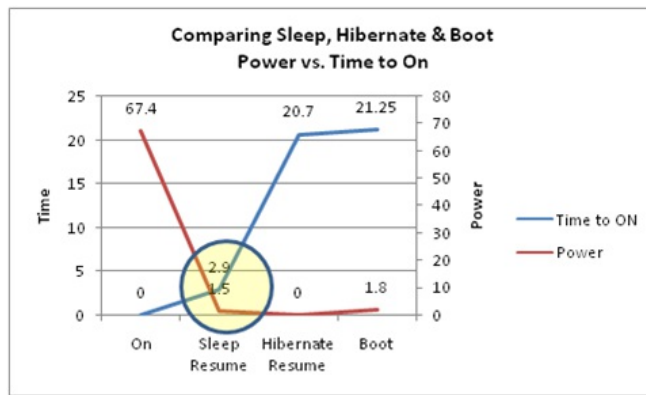
## What about turning my PC off?

So far, we have been talking about how to save power while the PC is ON. But, there are power savings to gain by entering low power modes when the PC is not in use. Many users simply Shut Down their computer when it is not in use, yet others use Sleep and sometimes Hibernate on mobile PCs. Windows features each of these power-saving modes so you can choose the right mode for how you use the system:

- **Sleep** : All of the open programs, documents and files are preserved in system RAM and the rest of the system is powered off. Because only memory is powered, Sleep consumes a very small amount of power—typically less than 1W on a mobile PC and typically less than 3W on a desktop PC. The primary benefit of Sleep is that resume is very fast—most systems resume from sleep in less than 2 seconds.
- **Hibernate**: All of the open programs, documents and files are copied from system RAM to the hard drive. The resulting file is called the *Hiberfile*. After RAM is copied into the Hiberfile, all of the PC is powered off. Hibernate is most often used on mobile PCs because it consumes nearly 0W on most laptops, and even if the battery does eventually drain, all of the open programs and documents are saved in the Hiberfile. As RAM continues to grow, and as some PCs have limited storage, Hibernate might not be the best option for folks. (As a quick tip, the disk cleanup wizard, or `powercfg -hibernate off`, can remove the disk space pre-allocated to hibernate).
- **Shut Down** - This is a normal Windows shutdown, nothing is saved to memory or disk, and the system boots again the next time the system is powered on.

Using an example desktop PC, we measured power consumption for Sleep, Hibernate, Shut Down and the basic ON state, with just the desktop shown and no open programs. We also measured resume latency—the amount of time to get the system back to the ON state.





The chart makes it pretty clear why we focus on Sleep reliability and performance, and encourage most people to use it when they are not using their computer. Sleep consumes nearly the same amount of power as Shut Down, but resumes the system in less than 2 seconds, instead of going through the boot process. You can see that boot takes a significant amount of power so when considering whether to turn off your machine to save power or to put it into a low power state, think about how long your machine will be out of use. Nevertheless, as we've talked about in previous blogs boot (and shutdown) are obviously very important performance scenarios as we engineer Windows 7.

## Next Steps

We are committed to continuously improving the energy efficiency of Windows PCs, and have made significant improvements to core platform power management for Windows 7, as well as tools to identify where power is consumed. We still have more work to do, and look forward to our upcoming Beta release and monitoring incoming CEIP telemetry for energy efficiency and power management results. Of course we continue to work very closely with the other members of the ecosystem as we all have much to contribute to energy efficiency—from the manufacturing, usage, and end of life of a PC, software, and peripherals.

--Dean

# Primer on Device Support and Testing for Windows 7

Steven Sinofsky | [2009-01-10T03:00:00+00:00](#)

---

*As most folks (finally) get the beta and start to set aside some time to install and try out Windows 7, we thought it would be a good idea to start to talk about how we support devices through testing and work across the PC ecosystem. This is a big undertaking and one that we take very seriously. As we talked about at the PDC, this is also an area where we learned some things which we want to apply to Engineering Windows 7. While this is a massive effort across the entire Windows organization, Grant George, the VP of Test for the Windows Experience, is taking the lead in authoring this post. We think this is a deep topic and I know folks want to know more so consider this a kick-off for more to come down the road. –Steven*

## Devices and Drivers in Windows

One of the most important responsibilities in a release of Windows is our support of, and compatibility with, all of the devices and their associated drivers that our users have. The abstraction layer in Windows to connect software and hardware is a crucial part of the operating system. That layer is surfaced through our driver model, which provides the interface for all of our partners in the multi-faceted hardware ecosystem. Windows supports a vast range of devices today – audio devices (speakers, headsets...), display devices (monitors...), print, fax and scan devices, connectivity to digital cameras, portable media devices of all shapes, sizes and functions, and more. Windows is an open platform for companies across the globe who develop and deliver these devices to the marketplace and our users – and our job is to make sure we understand that ecosystem and those choices and verify those devices and drivers work well for our customers – which includes partnering with those device providers throughout the engineering of Windows 7.

Drivers provide the interface between a device and the Windows operating system – and are citizens of the WDM ([Windows Driver Model](#)). WDM was initially created as an intermediary layer of kernel mode drivers to ease the authoring of drivers for Windows. There are different types of drivers. Class drivers (which are hardware device drivers that supports an array of devices of a similar hardware class where hardware manufacturers make their products compatible with standard protocols for interaction with the operating system) and device-specific drivers (provided by the device manufacturer for a specific device and sometimes a specific version of that device) are the two most common.

## Partner Support

Support for our hardware partners comes in the form of the [Windows Driver Kit](#) (WDK) and for certification, the [Windows Logo Kit](#) (WLK). The WDK enables the development of device drivers and as of Vista replaced the previous Windows Driver Development Kit (DDK). The WDK contains all of the DDK components plus Windows Driver Foundation (WDF) and the Installable File System kit (IFS). The Driver Test Manager (DTM) is another component here, but is separate from the WDK. The Windows Logo Kit (WLK) aids in certifying devices for Windows (it contains automated tests as well as a run-time framework for those tests). These tests are run and passed by our hardware vendor partners in order to use the Microsoft “Designed for Windows™” logo on devices. This certification process helps us and our hardware partners ensure a specific level of quality and compatibility for devices interacting with the Windows operating system. Hardware devices and drivers that pass the logo kits tests qualify for the Windows logo, driver distribution on Windows Update, and can be referenced in the online Windows Marketplace.

## Validation and Testing

With Windows 7 we have modified driver model validation, new and legacy device testing, and driver testing. Compared to Vista, we now place much more emphasis on validating the driver platform and verifying legacy devices and their associated drivers throughout our product engineering cycle. Data based on installed base for each device represents an integral part of testing, and we gather this data from a variety of sources including the voluntary, opt-in, anonymous telemetry in addition to sources such as sales data and IHV roadmaps. We have centralized and standardized the testing mechanics of the lab approach to this area of the product in a way that yields much earlier issue/bug discovery than in past releases. We have also ramped up our efforts to communicate platform or interface changes earlier with our external hardware partners to help them ensure their test cycles align with our schedule. In addition, we draw a more robust correlation between the real-world usage data, including recent trends, and prominence of each device and the prioritization it is given in our test labs. This is especially important for new and emerging devices that will come to market right before and just after we release Windows 7 to our customers.

Another important element in bringing a high quality experience to our Windows 7 users in device and driver connectivity and capability is the staging of our overall engineering process in Windows 7. For this release all of our engineering teams have followed a well structured and staged development process. The development/coding of new features and capabilities in Windows 7 was broken out in to 3 distinct phases (milestones) with dedicated integration and stabilization time at the end of each of these three coding phases. This included ensuring our code base remained highly stable throughout the development of Windows 7 and that our device and driver test validation was a constant part of those milestones. Larry discussed this in his [post](#) as some might recall. Program Managers, Developers and Testers all worked in super close partnership throughout the coding phases. Our work with external partners – especially our device manufacturer partners – was also enhanced through early forums we provided for them to learn about the changes in Windows 7 and also work closely with us on validation. Much more focus has been put on planning and then executing - planning the work and then working the plan. Our belief is that this yields much more predictability to developing and delivering our new features in Windows 7 both from a feature content and overall schedule standpoint. We recognize that this raised the bar on how our external partners see us execute and deliver on that plan when we say we will, but we also hope it increases their confidence in how they engage with us in validating the device experience during our development and delivery of Windows 7.

## Determining Which Devices to Test

Our program management team helps us drive device market share analysis. Most of their data comes from our [Customer Experience Improvement Program](#). This gives us data on the actual hardware in use across our customer base. For example there are over 16,000 unique 4-part hardware IDs for display devices alone. Like many things, we understand that it only takes a single device not functioning well to degrade an overall Windows experience or upgrade—we definitely want to re-enforce this shared understanding.

New devices typically have a small initial user base, but the driver will often be mostly new code (or the first time a code-base has seen a new device). As the device enters the mainstream, market share grows and most manufacturers continue to develop and improve their drivers. This is where for our customers, and our own testing, it's important to always have the latest drivers for a given device.

Over a device's lifetime, we work closely with our external device partners and represent as faithfully as possible in our test labs, a prioritized way of ensuring old and new devices continue to work well with Windows. By paying very close attention to trends in the market place across our device classes, we can make guided decisions in the context of these areas:

- Critical and mainstream devices we must support out-of-the-box
- Which drivers we must make available on Windows Update
- On which devices and drivers to focus our testing

Another benefit of close market tracking is creating an **equivalence-based view** of a device family.

## Equivalence Classes

We use the notion of equivalence classes to help us define and prioritize our hardware (device) test matrix. Creating equivalence classes involves grouping things into sets based on equivalent properties across related devices. For example, imagine if we worked for a chemical company and it was our job to test a car polish additive on actual automobiles. Given a fixed test budget, we would want to maximize the number of makes and models we test our product on. We begin by analyzing the current market space so we can make the best choices for our test matrix.

Let's say the first test car we analyze is a blue 2003 Ford Mustang. We also know that the same blue paint is used on all of Ford's 2003 and 2004 models and is also used on all of Mazda's 2005 models. This means our first automobile represents several entries in our table based on equivalence:

---

Test ID	Make	Model	Color	Year
1	Ford	Mustang	Blue	2003
2	Ford	*	Blue	2004
3	Mazda	*	Blue	2005

Now let's look at a silver 2001 Mercedes C240. We know that Mercedes and Chrysler have a relationship and upon further investigation we find Chrysler used the same silver paint on their 2006 through 2009 models. Now our equivalence class based test matrix looks like this:

---

Test ID	Make	Model	Color	Year
1	Ford	Mustang	Blue	2003
2	Ford	*	Blue	2004
3	Mazda	*	Blue	2005
4	Mercedes	C240	Silver	2001
5	Chrysler	*	Silver	2006

6	Chrysler	*	Silver	2007
7	Chrysler	*	Silver	2008
8	Chrysler	*	Silver	2009

By carefully analyzing each actual automobile, we have established an equivalence relationship that we can leverage to maximize implicit test coverage. Testing one make and model is theoretically equivalent to testing many. Of course we recognize in the real world different companies might use different techniques for applying paint, as one variable, so there are subtleties that require additional information to property class attributes for testing purposes.

Testing computer devices is very similar. Even though there are thousands of different devices on the market, many of them share major components, are die-shrinks of a previous revision, or differ only in terms of memory, clock-rate, pixel count, connector, or even the type of heat sink. Take for example display devices. There are over 16,000 display devices on the market. But the *equivalence* view reveals that 90% of the market is represented by about 60 different GPUs. By adding a few more to a carefully constructed test matrix based on equivalence it is possible to represent over 99% of all GPUs. Driver writers also leverage equivalence by targeting drivers at a range of hardware. Driver install packages indicate devices they support via hardware IDs.

All modern computer devices are assigned a unique hardware ID based on the device vendor, type, and class. Most IDs (PCI, PC Card, USB, and IEEE 1394 devices) are assigned by the industry standards body associated with that device type.

Let's look at the device ID of my display adapter:

```
PCI\VEN_10DE&DEV_0611&SUBSYS_C8013842&REV_A2
```

If I visit [PCI-SIG](#) (the standards body associated with all PCI device ID assignment) and do a search on 10DE, I'm told I this is an NVidia PCI ID. If I look further on my system in

```
C:\Windows\System32\DriverStore\FileRepository
```

I can find NVidia drivers (folders that start with nv\_lh). If I open one of the driver .INF files on my machine I see this tell-tale line:

```
NVIDIA_G92.DEV_0611.1 = "NVIDIA GeForce 8800 GT"
```

Further inspection of the driver .INF file tells me that the same G92 GPU is used for all of these devices:

- NVIDIA GeForce 8800 GTS 512
- NVIDIA GeForce 8800 GT
- NVIDIA GeForce 9800 GX2
- NVIDIA GeForce 8800 GS
- NVIDIA GeForce 9600 GSO
- NVIDIA GeForce 8800 GT
- NVIDIA GeForce 9800 GTX
- NVIDIA Quadro FX 3700

A bit of online research reveals other interesting information: “The 8800 GT, codenamed G92, was released on October 29, 2007. The card is the first to transition to 65 nm process, and supports PCI-Express 2.0.[13] It has a single-slot cooler as opposed to the double slot cooler on the 8800 GTS and GTX, and uses less power than GTS and GTX due to its 65 nm process.” -[Wikipedia](#)

So in theory, if I was to run a test on my display adapter, there’s a good chance I’d get the same results as I would on any of these other related devices.

## Driver Goals for Windows 7

One of our primary goals for Windows 7 is compatibility with all Vista certified drivers and to ensure that people have a seamless upgrade experience. This breaks down into several requirements that guide how we test:

- Drivers for basic functionality are in-box (by in-box we mean available as part of the installation of Windows). This includes drivers for mainstream storage, network, input, and display devices so the OS can be installed and user can get online where, if needed, additional drivers can be acquire from Windows Update.
- Drivers update and/or install with minimal end user effort.
- When drivers are upgraded, there aren’t problems with the new drivers.
- Drivers are reliable.

One question we are asked about quite a bit is the availability of drivers. There are three primary reasons drivers end up looking for folks: clean installation of Windows, attaching device to a new computer, wanting the updated driver. We definitely recognize that for the readers of this blog, both as enthusiasts and often the support/IT infrastructure for corporations, friends, and families, that the ability to acquire drivers and reliably update machines is something of a “hobby” we all love to hate. We all want the latest and greatest—no more and no less.

A clean installation is one we are all definitely valuing during the beta phase of Windows 7. It should be clear that a clean install, as important as it is to many of us, is not a routine/mainstream experience. Nevertheless, the combination of in-box drivers and those available via Windows Update will serve a very broad set of PCs (for example, you should see most of the drivers installed for the new Atom-based machines if you do a clean install). On the other hand, some drivers for PCs are only available from the PC maker and for a variety of reasons are not available for download from Windows Update or even the device manufacturer’s site. For example, mobile graphics drivers are generally available only from the PC maker and not from the graphics component maker—this is a decision they make because of the way these chipsets are delivered for each PC maker.

Obviously attaching an existing device to a new PC is a common occurrence. In this case you may have long ago lost the CD/DVD that came with a device and you just plug it in (because you ignored the warning saying “please run the setup program first”). Again, our goal is to provide these via Windows Update. Often IHVs have updates or significantly large downloads that for a number of reasons are not appropriate to deliver via Windows Update. In that case we can also alert you, with a link many times, to seek the driver from the vendor of the device.

Updating drivers is something we are all familiar with as we often read “get the latest driver” to address issues. We all see this particularly in the enthusiast gamer space where newer drivers also improve performance or offer more features, in addition to improving overall. The primary way to get updated drivers is generally through optional updates in Windows Update, though again many times the latest and greatest must be downloaded directly from an IHV (independent hardware vendor) site.

Our goal is clearly to make sure that drivers for the broadest set of devices are available and high quality. There are many equal partners that contribute to delivering a PC and all the associated devices and we work hard to develop a systematic way to reach the broadest set of customers with high quality software and support.

## **Scale of Device and Driver Testing in Windows 7**

The table below provides *examples* of some of the explicit devices we have directly tested thus far during the development of Windows 7. This is just a sampling of that direct testing - many more devices have been directly tested that are not shown here or are covered through equivalence classing.

This information is available in many sources, such as the WHQL web site that lists all qualified devices. For the purposes of this blog we thought it would be fun to provide a list here which we think will most certainly serve as the basis for discussion.













































































<b>Manufacturer</b>	<b>Description</b>	<b>Family</b>
Altec Lansing	T515	Audio
AMD (ATI)	Radeon 9200	Display
AMD (ATI)	FireGL 3100	Display
AMD (ATI)	Radeon X300/X550/X1050 Series	Display
AMD (ATI)	Radeon 9800 Pro	Display

AMD (ATI)	FireGL V3100	Display
AMD (ATI)	Radeon Xpress Series	Display
AMD (ATI)	Radeon Xpress Series	Display
AMD (ATI)	Radeon Xpress 1200	Display
AMD (ATI)	Radeon X700 PRO	Display
AMD (ATI)	Radeon X1200	Display
AMD (ATI)	Radeon X800 CrossFire Edition	Display
AMD (ATI)	Mobility Radeon X300	Display
AMD (ATI)	Radeon X850 CrossFire Edition	Display
AMD (ATI)	Radeon X1550	Display
AMD (ATI)	Radeon X1950 Series	Display
AMD (ATI)	Mobility Radeon X1300	Display
AMD (ATI)	Mobility Radeon X1400	Display
AMD (ATI)	Mobility Radeon HD3200	Display
AMD (ATI)	Radeon HD 2600 XT	Display
AMD (ATI)	Radeon HD 3850	Display
AMD (ATI)	Radeon HD 3870	Display

AMD (ATI)	Radeon HD 3200	Display
AMD (ATI)	Radeon HD 2400	Display
AMD (ATI)	FireGL 6000	Display
AMD (ATI)	FireGL 8200	Display
AMD (ATI)	Radeon HD 2900 XT	Display
AMD (ATI)	Radeon HD 2600	Display
AMD (ATI)	Radeon HD 4850	Display
AMD (ATI)	Radeon HD4670	Display
AMD (ATI)	ATI Technologies, Inc. RAGE XL PCI	Display
AMD (ATI)	RADEON 7000 Series	Display
Analog Devices	AD1884	Audio
Analog Devices	AD1984	Audio
Analog Devices	AD1981	Audio
Analog Devices	ADI1986A	Audio
Analog Devices	ADI1988B	Audio
Analog Devices Inc.	ADI AC97	Audio
Apple	iPhone headset	Audio

Apple	iSight 640x480 Firewire	VidCap
Archos	Archos605(WiFi)	Portable Device
ATI	ATI HDMI	Audio
BlueAnt	X5 Stereo BT Headset	Audio
Brother	HL-5140	Print / Scan
Brother	HL-2070	Print / Scan
Brother	MFC-8440	Print / Scan
Brother	MFC-5840c	Print / Scan
Brother	HL-5150	Print / Scan
Brother	MFC-8840	Print / Scan
Brother	HL-6050D	Print / Scan
Brother	IntelliFax-5750e	Print / Scan
Brother	IntelliFax-5750	Print / Scan
Canon	Canon A720IS	Portable Device
Canon	Digital Rebel XT	Portable Device
Canon	A420/410	Portable Device
Canon	SD430	Portable Device

Canon	Pixma MP140	Print / Scan
Canon	Pixma iP1800	Print / Scan
Canon	Pixma iP1700	Print / Scan
Canon	Pixma iP2500	Print / Scan
Canon	Pixma MP210	Print / Scan
Canon	Pixma MP160	Print / Scan
Canon	Pixma iP1500	Print / Scan
Canon	Pixma iP1600	Print / Scan
Canon	Pixma iP4200	Print / Scan
Canon	Pixma iP3500	Print / Scan
Canon	Pixma iP4500	Print / Scan
Canon	Pixma MP180	Print / Scan
Canon	Pixma iP2000	Print / Scan
Canon	i475D	Print / Scan
Canon	Pixma MP150	Print / Scan
Canon	i250	Print / Scan
Canon	Pixma MP520	Print / Scan

Canon	S450	Print / Scan
Canon	MultiPass MP390	Print / Scan
Canon	Pixma MP500	Print / Scan
Canon	Pixma MX300	Print / Scan
Canon	Pixma iP1000	Print / Scan
Canon	Pixma MP610	Print / Scan
Canon	MultiPass MP190	Print / Scan
Canon	Pixma iP6210D	Print / Scan
Canon	Pixma iP5200	Print / Scan
Canon	Pixma iP3300	Print / Scan
Canon	Pixma iP3000	Print / Scan
Canon	Pixma MP510	Print / Scan
Canon	Pixma iP90	Print / Scan
Canon	i350	Print / Scan
Canon	Pixma iP6600D	Print / Scan
Canon	Pixma MP830	Print / Scan
Canon	BJC-6000	Print / Scan

Canon	i550	Print / Scan
Canon	Pixma MP170	Print / Scan
Canon	Pixma MP460	Print / Scan
Canon	Pixma MP600	Print / Scan
Canon	Pixma iP4300	Print / Scan
Canon	i860	Print / Scan
Canon	Pixma MP110	Print / Scan
Canon	i320	Print / Scan
Canon	Pixma iP6220D	Print / Scan
Canon	Pixma MP130	Print / Scan
Canon	Pixma iP6310D	Print / Scan
Canon	i960/i965	Print / Scan
Canon	Pixma MP950	Print / Scan
Canon	Selphy Series	Print / Scan
Canon	i560	Print / Scan
Canon	Pixma iP8500	Print / Scan
Canon	MultiPass MP370	Print / Scan



Canon	Pixma iP4000	Print / Scan
Canon	i9900	Print / Scan
Canon	Pixma iX4000	Print / Scan
Canon	i865	Print / Scan
Canon	Pixma mini260	Print / Scan
Canon	Pixma iX5000	Print / Scan
Canon	i850	Print / Scan
Canon	S530D	Print / Scan
Canon	Pixma MP800R	Print / Scan
Canon	Pixma iP5200R	Print / Scan
Canon	i470D Photo Printer	Print / Scan
Canon	S600	Print / Scan
Canon	BJC-85	Print / Scan
Canon	Pixma iP6000	Print / Scan
Canon	S9000	Print / Scan
Canon	Pixma MP750	Print / Scan
Canon	Pixma MP780	Print / Scan

Canon	S630	Print / Scan
Canon	MultiPass MP1000	Print / Scan
Canon	S520	Print / Scan
Canon	Pixma MP810	Print / Scan
Canon	Pixma iP5000	Print / Scan
Canon	Pixma iP6700D	Print / Scan
Canon	Pixma iP80	Print / Scan
Canon	SD600	Portable Device
Canon Inc.	PowerShot A720 IS	Portable Device
CASIO COMPUTER CO.,LTD.	EX-Z1200	Portable Device
Chrontel	Chrontel HDMI	Audio
Conexant	Venice	Audio
Creative	MP3+ (SB0270)	Audio
Creative	Xmod	Audio
Creative	Live! Cam Optia AF	VidCap
Creative	WebCam Live! USB	VidCap
Creative	Webcam Notebook 640x480 USB	VidCap

Creative	WebCam Instant 352x288 USB	VidCap
Creative	WebCam NX Pro 640x480 USB	VidCap
Creative	WEBCAM NX	VidCap
Creative	Live! Cam Notebook Pro 640K USB 2.0	VidCap
Creative	Live! Cam Video IM Pro VGA USB 2.0	VidCap
Creative	Webcam Live Ultra 640x480 USB 2.0 Manual Focus Ring	VidCap
Creative Labs, Inc.	Live! Series	Audio
Creative Labs, Inc.	Audigy Series	Audio
Creative Labs, Inc.	X-Fi Series	Audio
Creative Technology Ltd	Nano Plus	Portable Device
Creative Technology Ltd	NOMAD MuVo TX	Portable Device
Creative Technology Ltd	Zen Vision M	Portable Device
Creative Technology Ltd	Vision W	Portable Device
Creative Technology Ltd	Sleek	Portable Device
Creative Technology Ltd	PMC v2	Portable Device
Dell	Axim X51v	Portable Device
Dell	AiO 810	Print / Scan

Dell	A924	Print / Scan
Dell	J740	Print / Scan
Dell	1600n	Print / Scan
Dell	A922	Print / Scan
Dell	A940	Print / Scan
Dell	LP 1720dn	Print / Scan
Dell	3100cn	Print / Scan
Dell	W5300N	Print / Scan
Denon	S-52	Media Sharing
Dixim	media server	Media Sharing
Dlink	DSM-210	Media Sharing
Dlink	DSM - 520	Media Sharing
Dlink	DSM - 510	Media Sharing
Drobo	Drobo NAS	Media Sharing
Epson	Stylus Color C88+	Print / Scan
Epson	Stylus Color C84/C85	Print / Scan
Epson	Stylus Color C86/C87	Print / Scan

Epson	Stylus Color C64	Print / Scan
Epson	Stylus Photo R265	Print / Scan
Epson	LQ-570/670	Print / Scan
Epson	FX-880	Print / Scan
Epson	Stylus Photo R220	Print / Scan
Epson	LQ-300	Print / Scan
Epson	Stylus Photo R320	Print / Scan
Epson	Stylus CX6600/6500/6900	Print / Scan
Epson	Stylus CX5400	Print / Scan
Epson	Stylus Photo 1270	Print / Scan
Epson	LQ-1070+	Print / Scan
Epson	Stylus Photo R200	Print / Scan
Epson	Stylus Photo 1280/1290	Print / Scan
Epson	Stylus Color 900/N	Print / Scan
Epson	Stylus Color C62	Print / Scan
Epson	ActionPrinter 5000+	Print / Scan
Epson	Stylus Photo 820	Print / Scan

Epson	Stylus Color 660	Print / Scan
Epson	Stylus Color 640	Print / Scan
Epson	AcuLaser 2600N	Print / Scan
Epson	FX-2170	Print / Scan
Epson	FX-2190	Print / Scan
FujiFilm	F30	Portable Device
General Electric	EasyCam USB PC Camera 640x480	VidCap
GN\Jabra	GN9330	Audio
GN\Jabra	GN9350	Audio
GN\Jabra	GN2000USB	Audio
HP	HD TV	Media Sharing
HP	Photosmart R717	Portable Device
HP	Deskjet D1400 series	Print / Scan
HP	Deskjet F380	Print / Scan
HP	Deskjet F4100	Print / Scan
HP	LaserJet 1018	Print / Scan
HP	LaserJet 1020	Print / Scan

HP	Photosmart C3180	Print / Scan
HP	Deskjet D2400 Series	Print / Scan
HP	LaserJet P2015	Print / Scan
HP	Officejet K550	Print / Scan
HP	PSC 1410	Print / Scan
HP	Deskjet F2100 series	Print / Scan
HP	PSC 1315	Print / Scan
HP	Deskjet 5440	Print / Scan
HP	Color LaserJet 2600	Print / Scan
HP	Officejet 5700	Print / Scan
HP	PSC 1510	Print / Scan
HP	Photosmart C4200	Print / Scan
HP	Deskjet 5150	Print / Scan
HP	Deskjet 930C/935C	Print / Scan
HP	Deskjet 5940	Print / Scan
HP	Photosmart C4180	Print / Scan
HP	Deskjet D2330	Print / Scan

HP	LaserJet 1022	Print / Scan
HP	Deskjet 3745	Print / Scan
HP	Deskjet 5550	Print / Scan
HP	Photosmart C5200	Print / Scan
HP	Officejet 5610	Print / Scan
HP	Deskjet D2360	Print / Scan
HP	Deskjet 3900 Series	Print / Scan
HP	Photosmart C5180	Print / Scan
HP	Deskjet 5740	Print / Scan
HP	Deskjet D4200 Series	Print / Scan
HP	Deskjet 6122	Print / Scan
HP	Deskjet 950C	Print / Scan
HP	Deskjet 940C	Print / Scan
HP	PSC 1610	Print / Scan
HP	Photosmart D5160	Print / Scan
HP	Officejet 6200 Series	Print / Scan
HP	Deskjet 3845	Print / Scan



HP	Deskjet 3650	Print / Scan
HP	PSC 2355	Print / Scan
HP	Officejet 6300 Series	Print / Scan
HP	LaserJet P2014	Print / Scan
HP	LaserJet 1300	Print / Scan
HP	Officejet Pro L7500	Print / Scan
HP	Officejet Pro L7600	Print / Scan
HP	PSC 1350	Print / Scan
HP	Deskjet 9800	Print / Scan
HP	Photosmart 2575	Print / Scan
HP	Deskjet 450ci	Print / Scan
HP	Officejet 4215	Print / Scan
HP	LaserJet 1160	Print / Scan
HP	Deskjet 5650	Print / Scan
HP	Officejet 7400 Series	Print / Scan
HP	Deskjet 3740	Print / Scan
HP	Officejet 5510 Series	Print / Scan

HP	Photosmart 3210	Print / Scan
HP	Officejet 7300 Series	Print / Scan
HP	Photosmart 7850	Print / Scan
HP	Deskjet 832C	Print / Scan
HP	Deskjet 1220C	Print / Scan
HP	LaserJet 3030 MFP	Print / Scan
HP	Photosmart A616	Print / Scan
HP	LaserJet 3055	Print / Scan
HP	Deskjet 720C	Print / Scan
HP	Photosmart 7260	Print / Scan
HP	Deskjet 3320	Print / Scan
HP	Deskjet 970C	Print / Scan
HP	Photosmart A440	Print / Scan
HP	Deskjet 695C/697C	Print / Scan
HP	Photosmart A516	Print / Scan
HP	Deskjet 6540	Print / Scan
HP	Deskjet 6940	Print / Scan

HP	PSC 2510	Print / Scan
HP	Officejet 6100 Series	Print / Scan
HP	Deskjet 6840	Print / Scan
HP	Photosmart A430	Print / Scan
HP	Photosmart 7450	Print / Scan
HP	Deskjet 812C/815C	Print / Scan
HP	Photosmart 375	Print / Scan
HP	Officejet V40 Series	Print / Scan
HP	Deskjet 840/843/845	Print / Scan
HP	Photosmart D7400 Series	Print / Scan
HP	PSC 950 Series	Print / Scan
HP	Officejet G Series	Print / Scan
HP	LaserJet 1015	Print / Scan
HP	Photosmart 7960	Print / Scan
HP	Deskjet 895C	Print / Scan
HP	Photosmart 8450	Print / Scan
HP	Photosmart Pro B8350	Print / Scan

HP	Deskjet 1180c	Print / Scan
HP	LaserJet 4345 MFP	Print / Scan
HP	LaserJet 4250	Print / Scan
HP	LaserJet P3005	Print / Scan
HP	LaserJet 5200	Print / Scan
HP	LaserJet 4350n	Print / Scan
HP	Color LaserJet 4700	Print / Scan
HP	LaserJet 2300	Print / Scan
HP	LaserJet 4000	Print / Scan
HP	Color LaserJet 5550	Print / Scan
HP	Color LaserJet 3800	Print / Scan
HP	LaserJet 4050	Print / Scan
HP	Color LaserJet 3600	Print / Scan
HP	LaserJet 9050	Print / Scan
HP	LaserJet 2100	Print / Scan
HP	LaserJet 4240	Print / Scan
HP	LaserJet 2200	Print / Scan

HP	Color LaserJet 3000	Print / Scan
HP	LaserJet 4100	Print / Scan
HP	LaserJet 5000	Print / Scan
HP	Business Inkjet 1200D	Print / Scan
HP	Color LaserJet 4550	Print / Scan
HP	Color LaserJet 4600	Print / Scan
HP	Color LaserJet CP4005	Print / Scan
HP	Color LaserJet 3700	Print / Scan
HP	Color LaserJet 3500	Print / Scan
HP	LaserJet 9000 MFP	Print / Scan
HP	LaserJet 4 Plus	Print / Scan
HP	LaserJet III	Print / Scan
HP	LaserJet 6MP	Print / Scan
HP	Color LaserJet 1500L	Print / Scan
HP	PSC 1315	Print / Scan
HP	Officejet 5610	Print / Scan
HP	PSC 1350	Print / Scan

HP	LaserJet 4345 MFP	Print / Scan
HTC	TyTN II	Portable Device
IDT	STAC9220(9223)7680	Audio
IDT	STAC9220(9223)7681	Audio
IDT	STAC9227X(D)7618	Audio
IDT	STAC9227X(D)7619	Audio
IDT	STAC9225(Sony)7662	Audio
IDT	STAC9225(Sony)7664	Audio
IDT	STAC9225(Sony)7661	Audio
IDT	STAC9200	Audio
IDT	STAC9228	Audio
IDT	STAC9205	Audio
IDT	STAC9250	Audio
Insignia	NS-BTHDP	Audio
Insignia	NS-DV4G	Portable Device
Insignia	NS-DA2G	Portable Device
Intel	Intel HDMI	Audio

Intel	i965GX/G35	Display
Intel	G3x	Display
Intel	i4G	Display
Intel	i45GM	Display
Intel	i915GM	Display
Intel	i915G	Display
Intel	i945G	Display
Intel	i945GM	Display
Intel	Q3x	Display
Intel	i965G	Display
Intel	i965GM	Display
Iriver	ClixGen2	Portable Device
Iriver	IriverClix2_FWv1.14	Portable Device
Iriver	U10 Series	Portable Device
Iriver	Clix	Portable Device
Jabra	BT620S	Audio
Jabra	BT8010	Audio

Jabra	BT3030	Audio
Jasco	Minicam Pro	VidCap
Kodak	Easysshare LS420	Portable Device
Konica Minolta	magicolor 5450	Print / Scan
Kyocera Mita	FS-6900	Print / Scan
LABTEC	LABTEC WEBCAM PRO 961358	VidCap
LABTEC	Web Cam Plus 352x288 USB 2.0 Manual Focus Motion Detection	VidCap
Lexmark	Z845	Print / Scan
Lexmark	Z1300	Print / Scan
Lexmark	X2550	Print / Scan
Lexmark	X1270	Print / Scan
Lexmark	X2470	Print / Scan
Lexmark	Z735	Print / Scan
Lexmark	E120n	Print / Scan
Lexmark	X3550	Print / Scan
Lexmark	Z715	Print / Scan
Lexmark	Z42 Color JetPrinter	Print / Scan



Lexmark	X5470	Print / Scan
Lexmark	Z816	Print / Scan
Lexmark	Z615	Print / Scan
Lexmark	X2250	Print / Scan
Lexmark	P915	Print / Scan
Lexmark	X7170	Print / Scan
Lexmark	X4550	Print / Scan
Lexmark	X6170	Print / Scan
Lexmark	X6150	Print / Scan
Lexmark	E232	Print / Scan
Lexmark	2490	Print / Scan
Lexmark	P3150	Print / Scan
Lexmark	X5150	Print / Scan
Lexmark	E323	Print / Scan
Lexmark	P315	Print / Scan
Lexmark	Z25 Color JetPrinter	Print / Scan
Lexmark	2491	Print / Scan

Lexmark	X215	Print / Scan
Lexmark	X4250	Print / Scan
Lexmark	E321	Print / Scan
Lexmark	Z45 Color JetPrinter	Print / Scan
Lexmark	X83	Print / Scan
Lexmark	C524	Print / Scan
Lexmark	E450D	Print / Scan
Lexmark	T640	Print / Scan
Lexmark	X634	Print / Scan
Lexmark	W840	Print / Scan
Lexmark	X632	Print / Scan
Lexmark	X620	Print / Scan
Lexmark	X630	Print / Scan
Lexmark	T642	Print / Scan
Lexmark	W812	Print / Scan
Lexmark	X1270	Print / Scan
LG	HBS-200	Audio

Logitech	QuickCam Pro 9000	Audio
Logitech	QuickCam Pro 9000	VidCap
Logitech	Quickcam Communicate STX VGA Fixed Focus USB 2.0	VidCap
Logitech	QuickCam Chat VGA w/Image Capture USB 2.0	VidCap
Logitech	961400-0403 QuickCam Notebook Deluxe 1.3MP MF USB 2.0	VidCap
Logitech	QuickCam Pro 4000 640x480 USB 2.0	VidCap
Logitech	QuickCam Pro 5000 640x480 USB 2.0	VidCap
Logitech	Quickcam Vision Pro1	VidCap
Logitech	Quickcam Vision Pro2	VidCap
Logitech	961403 QuickCam Fusion 1.3MP USB 2.0	VidCap
Logitech	QuickCam Messenger 640x480 USB	VidCap
Logitech	QuickCam Messenger Refresh 640x480 USB	VidCap
Logitech	QuickCam Notebooks Pro 1.3MP USB 2.0	VidCap
Logitech	QuickCam Zoom 640x480 USB	VidCap
Logitech	QuickCam Communicate 640x480 USB 2.0	VidCap
Logitech	QuickCam Orbit MP 1.3MP USB 2.0	VidCap

Logitech	QUICKCAMFORNB	VidCap
Logitech	QuickCam Orbit 640x480 USB 2.0	VidCap
Logitech	QuickCam for Notebooks Pro	VidCap
Lubix	UBHS-LC1	Audio
Matrox	M9120	Display
Microsoft	NX-3000	Audio
Microsoft	VX-7000	Audio
Microsoft	NX-6000	Audio
Microsoft	VX-6000	Audio
Microsoft	VX-3000	Audio
Microsoft	VX-1000	Audio
Microsoft	LX-3000	Audio
Microsoft	ZX-6000	Audio
Microsoft	Mic Array	Audio
Microsoft	XBox 360	Media Sharing
Microsoft	LifeCam VX-1000 VGA USB 2.0	VidCap
Microsoft	Lifecam NX-6000	VidCap

Microsoft	LifeCam VX-6000 1.3MP USB 2.0	VidCap
Microsoft	LifeCam VX-3000 1.3MP USB 2.0	VidCap
Microsoft	Xbox Live Vision (Xbox 360)	VidCap
Microsoft	LifeCam VX-7000	VidCap
Microsoft	LifeCam NX-3000	VidCap
Momento	Wireless Picture Frame	Media Sharing
Motorola	S9	Audio
Motorola	HT820	Audio
Motorola	H670	Audio
Motorola	HS850	Audio
Motorola	H500	Audio
Motorola	DJ S805	Audio
NEC	UTR-UC-1	Audio
Nero8 Home Media	media server	Media Sharing
Nikon	CoolPix S1	Portable Device
Nokia	BH800	Audio
Nokia	N95	Media Sharing

Nokia	N95	Portable Device
Nokia	5300	Portable Device
nVidia	nVidia HDMI	Audio
Nvidia	GeForce 7600GT	Display
Nvidia	GeForce 7800GT	Display
Nvidia	Geforce 8200	Display
Nvidia	GeForce 7400 Go	Display
Nvidia	Geforce 7950 GX2	Display
Nvidia	Geforce 8800GTS	Display
Nvidia	Geforce 8800GTX	Display
Nvidia	Geforce 8400 GS	Display
Nvidia	GeForce 8400M GS	Display
Nvidia	Geforce 8600 GT	Display
Nvidia	Quadro NVS 130m	Display
Nvidia	Quadro 570	Display
Nvidia	Quadro 570m	Display
Nvidia	GeForce 9600 GT	Display

Nvidia	GeForce 8800 GT	Display
Nvidia	Geforce 8400GS (G98)	Display
Nvidia	Geforce 9800 X2	Display
Nvidia	Geforce GTX 260	Display
Nvidia	GeForce4 MX 420	Display
Nvidia	GeForce FX 5200	Display
Nvidia	Geforce FX 5900	Display
Nvidia	GeForce 6150	Display
Nvidia	GeForce 6100	Display
Nvidia	GeForce 6200	Display
Nvidia	GeForce 7050	Display
Nvidia	GeForce 6800	Display
Nvidia	GeForce Go 6150	Display
Oki	Microline 320/Turbo	Print / Scan
Oki	Microline 184 Turbo	Print / Scan
Oki	Microline 391/Turbo	Print / Scan
Oki	Microline 321/Turbo	Print / Scan

Oki	Microline 590	Print / Scan
Panasonic	KX-P2130	Print / Scan
Panasonic	KX-P2023	Print / Scan
Parrot	Boombox	Audio
Philips	Stereo Mic	Audio
Philips	GoGear 30GB	Portable Device
Plantronics	Pulsar 590A/E	Audio
Plantronics	Pulsar 260	Audio
Plantronics	Discovery 655 or 665	Audio
Plantronics	SupraPluc DA45	Audio
Polycom	CX400	Audio
Realtek	Realtek 262 HD Audio codec	Audio
Realtek	Realtek 268 HD Audio codec	Audio
Realtek	Realtek 660 HD Audio codec	Audio
Realtek	Realtek 862 HD Audio codec	Audio
Realtek	Realtek 883 HD Audio codec	Audio
Realtek	Realtek 888 HD Audio codec	Audio



Realtek	Realtek 885 HD Audio codec	Audio
Realtek	Realtek 882 HD Audio codec	Audio
Realtek	Realtek 861 HD Audio codec	Audio
Realtek	Realtek 662 HD Audio codec	Audio
Realtek Semiconductor Corp	Realtek AC97	Audio
Rhapsody	music Jukebox	Media Sharing
RIO	Rio Carbon	Portable Device
Roku	Radio Soundbridge	Media Sharing
Roku	SoundbridgeM1000	Media Sharing
S3	GammaChrome G700	Display
S3	GammaChrome G700	Display
S3	S3 Graphics Chrome 440/430 Series	Display
S3	S3 Graphics Chrome 440/430 Series	Display
Samsung	WEP-210	Audio
Samsung	YP-Z5	Portable Device
Samsung	ML-1610	Print / Scan
Samsung	SF-5100	Print / Scan

Samsung	ML-1710	Print / Scan
SanDisk Corporation	Sansa E260	Portable Device
SanDisk Corporation	Sansa View Mp3 Player	Portable Device
SanDisk Corporation	Sansa m250	Portable Device
SI	1392 HDMI	Audio
SigmaTel, Inc.	Sigmatel AC97	Audio
SiS	Xabre	Display
SiS	Mirage3	Display
Sonos	Zone player ZP80	Media Sharing
Sony	DR-BT22	Audio
Sony	PS3	Media Sharing
Sony	DSC-T200	Portable Device
Sony Corporation	WALKMAN NWZ-A816	Portable Device
Sony Ericsson	W910i	Portable Device
Toshiba	Gigabeat	Portable Device
Toshiba	Gigabeat V2 PMC	Portable Device
Turtle Beach	Audio Advantage Micro	Audio

Tiversity Inc	media server	Media Sharing
Twonky Media	media server	Media Sharing
Via	DeltaChrome G700	Display
Western Digital	External harddrive	Media Sharing
Xerox	Phaser 6120	Print / Scan
Xerox	Phaser 4510	Print / Scan

# User Account Control (UAC) – quick update

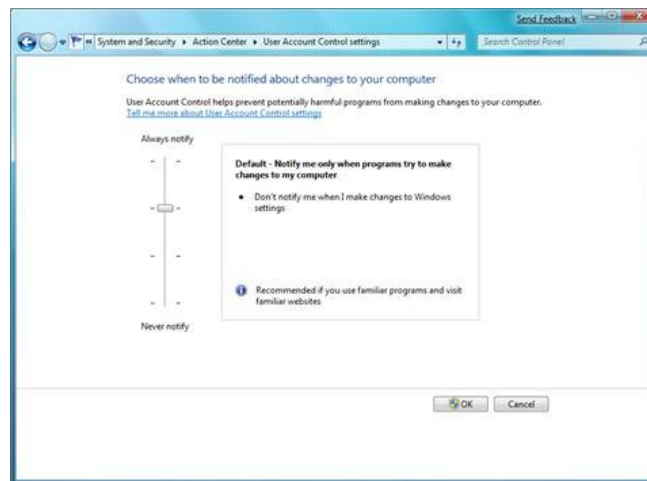
Steven Sinofsky | [2009-01-15T03:00:00+00:00](#)

*There's been a ton of interest in how we have improved user account control (UAC) and so we thought we'd offer a quick update for folks. We know most of you have discovered this and picked a setting that works for you, and we're happy with the feedback we've seen. This just goes into the details on the choice of defaults. –Steven*

In an earlier [blog post](#) we discussed the why of UAC and its implications for Windows, the ecosystem, and our customers. We also talked about what we needed to do moving forward to address the data and feedback we've received. This blog post will provide additional detail on our response and what you can expect to see in the upcoming beta build in early 2009.

As mentioned in our previous post, and your comments supported this, the goals for UAC are good and important ones. User Account Control was created with the intention of putting you in control of your system, reducing cost of ownership over time, and improving the software ecosystem. It is important not to abandon these goals. Instead, we want to address feedback we've received and build on the telemetry we have using those to improve the overall experience without losing sight of the goals with which we agree.

For those of you using 6801 you have started to see the benefits of prompt reduction and our new and improved dialog designs. You also have seen our efforts to give the user greater control of their system—the new UAC Control Panel. The administrator now has more control over the level of notification received from UAC. Look for the UAC Control Panel to appear in Start Search, Action Center, Getting Started, and even directly from the UAC prompt itself. Of course, the familiar ways to access it from Vista are still present.



**Figure 1:** UAC Control Panel

The UAC Control Panel enables you to choose between four different settings:

1. **Always notify on every system change.** This is Vista behavior – a UAC prompt will result when any system-level change is made (Windows settings, software installation, etc.)
2. **Notify me only when programs try to make changes to my computer.** This setting does not prompt when you change Windows settings, such as control panel and administration tasks.
3. **Notify me only when programs try to make changes to my computer, without using the Secure Desktop.** This is the same as #2,

but the UAC prompt appears on the normal desktop instead of the Secure Desktop. While this is useful for certain video drivers which make the desktop switch slowly, note that the Secure Desktop is a barrier to software that might try to spoof your response.

4. **Never notify.** This turns off UAC altogether.

We know from the feedback we've received that our customers are looking for a better balance of control versus the amount of notifications they see. As we mentioned in our last post we have a large number of admin (aka developer) customers looking for this balance, our data shows us that most machines (75%) run with a single account with full admin privileges.



**Figure 2.** Percentage of machines (server excluded) with one or more user accounts from January 2008 to June 2008.

For the in-box default, we are focusing on these customers, and we have chosen number 2, “Notify me only when programs try to make changes to my computer”. This setting does not prompt when you change Windows settings (control panels, etc.), but instead enables you to focus on administrative changes being requested by non-Windows applications (like installing new software). For people who want greater control in changing Windows settings frequently, without the additional notifications, this setting results in fewer overall prompts and enables customers to zero in on the key remaining notifications that they do see.

This default setting provides the right degree of change notification that a broad range of customers’ desire. At the same time we’ve made it easy and readily discoverable for the administrator to adjust the setting to provide more or fewer notifications via the new control panel (and policy). As with all of our default choices we will continue to closely monitor the feedback and data that come in through beta before finalizing for ship.

--UAC, Kernel, and Security program managers

# Engineering the Windows 7 “Windows Experience Index”

Steven Sinofsky | [2009-01-19T03:00:00+00:00](#)

---

*We're busy going through tons of telemetry from the many people that have downloaded and installed the Windows 7 beta around the world. We're super excited to see the excitement around kicking the tires. Since most folks on the beta are well-versed in the hardware they use and very tuned into the choices they make, we've received a few questions about the [Windows Experience Index \(WEI\)](#) in Windows 7 and how that has been changed and improved in Windows 7 to take into account new hardware available for each of the major classes in the metric. In this post Michael Fortin returns to dive into the engineering details of the WEI.*

*The WEI was introduced in Windows Vista to provide one means across PCs to measure the relative performance of key hardware components. Like any index or benchmark, it is best used as a relative measure and should not be used to compare one measure to another. Unlike many other measures, the WEI merely measures the relative capability of components. The WEI only runs for a short time and does not measure the interactions of components under a software load, but rather characteristics of your hardware. As such it does not (nor cannot) measure how a system will perform under your own usage scenarios. Thus the WEI does not measure performance of a system, but merely the relative hardware capabilities when running Windows 7.*

*We do want to caution folks in trying to generalize an “absolute” WEI as necessary for a given individual. We each have different tolerances or more importantly expectations for how a PC should perform and the same WEI might mean very different things to different individuals. To personalize this, I do about 90% of my work on a PC with a WEI of 2.0, primarily driven by the relatively low score for the gaming graphics component on my very low cost laptop. I run Outlook (with ~2GB of email), Internet Explorer (with a dozen tabs), Excel (with long list of people on the development team), PowerPoint, Messenger (with video), and often I am running one of several LOB applications written in .NET. I feel with this type of workload and a PC with Windows 7 and that WEI my own brain and fingers continues to be my “bottleneck”. At the other end of the spectrum is my holiday gift machine which is a 25” all-in-one with a WEI of 5.1 (though still limited by gaming graphics, with subscores of 7.2, 7.2, 6.2, 5.1, 5.9). This machine runs Windows 7 64-bit and I definitely don't keep it very busy even though I run MediaCenter in a window all the time, have a bunch of desktop gadgets, and run the PC as our print server (I use about 25% of available RAM and the CPU almost never gets above 10%).*

–Steven

The overall Windows Experience Index (WEI) is defined to be the lowest of the five top-level WEI subscores, where each subscore is computed using a set of rules and a suite of system assessment tests. The five areas scored in Windows 7 are the same as they were in Vista and include:

- Processor
- Memory (RAM)
- Graphics (general desktop work)
- Gaming Graphics (typically 3D)
- Primary Hard Disk

Though the scoring areas are the same, the ranges have changed. In Vista, the WEI scores ranged from 1.0 to 5.9. In Windows 7, the range has been extended upward to 7.9. The scoring rules for devices have also changed from Vista to reflect experience and feedback comparing closely rated devices with differing quality of actual use (i.e. to make the rating more indicative of actual use.) We know during the beta some folks have noticed that the score changed (relative to Vista) for one or more components in their system and this tuning, which we will describe here, is responsible for the change.

For a given score range, we hope our customers will be able to utilize some general guidelines to help understand the experiences a particular PC can be expected to deliver well, relatively speaking. These Vista-era general [guidelines](#) for systems in the 1.0, 2.0, 3.0, 4.0 and 5.0 ranges still apply to Windows 7. But, as noted above, Windows 7 has added levels 6.0 and 7.0; meaning 7.9 is the maximum score possible. These new

levels were designed to capture the rather substantial improvements we are seeing in key technologies as they enter the mainstream, such as solid state disks, multi-core processors, and higher end graphics adapters. Additionally, the amount of memory in a system is a determining factor.

For these new levels, we're working to add guidelines for each level. As an example for gaming users, we expect systems with gaming graphics scores in the 6.0 to 6.9 range to support DX10 graphics and deliver good frames rates at typical screen resolutions (like 40-50 frames per second at 1280x1024). In the range of 7.0 to 7.9, we would expect higher frame rates at even higher screen resolutions. Obviously, the specifics of each game have much to do with this and the WEI scores are also meant to help game developers decide how best to scale their experience on a given system. Graphics is an area where there is both the widest variety of scores readily available in hardware and also the widest breadth of expectations. The extremes at which CAD, HD video, photography, and gamers push graphics compared to the average business user or a consumer (doing many of these same things as an avocation rather than vocation) is significant.

Of course, adding new levels doesn't explain why a Vista system or component that used to score 4.0 or higher is now obtaining a score of 2.9. In most cases, large score drops will be due to the addition of some new disk tests in Windows 7 as that is where we've seen both interesting real world learning and substantial changes in the hardware landscape.

With respect to disk scores, as discussed in our recent post on [Windows Performance](#), we've been developing a comprehensive performance feedback loop for quite some time. With that loop, we've been able to capture thousands of detailed traces covering periods of time where the computer's current user indicated an application, or Windows, was experiencing severe responsiveness problems. In analyzing these traces we saw a connection to disk I/O and we often found typical 4KB disk reads to take longer than expected, much, much longer in fact (10x to 30x). Instead of taking 10s of milliseconds to complete, we'd often find sequences where individual disk reads took many hundreds of milliseconds to finish. When sequences of these accumulate, higher level application responsiveness can suffer dramatically.

With the problem recognized, we synthesized many of the I/O sequences and undertook a large study on many, many disk drives, including solid state drives. While we did find a good number of drives to be excellent, we unfortunately also found many to have significant challenges under this type of load, which based on telemetry is rather common. In particular, we found the first generation of solid state drives to be broadly challenged when confronted with these commonly seen client I/O sequences.

An example problematic sequence consists of a series of sequential and random I/Os intermixed with one or more flushes. During these sequences, many of the random writes complete in unrealistically short periods of time (say 500 microseconds). Very short I/O completion times indicate caching; the actual work of moving the bits to spinning media, or to flash cells, is postponed. After a period of returning success very quickly, a backlog of deferred work is built up. What happens next is different from drive to drive. Some drives continue to consistently respond to reads as expected, no matter the earlier issued and postponed writes/flushes, which yields good performance and no perceived problems for the person using the PC. Some drives, however, reads are often held off for very lengthy periods as the drives apparently attempt to clear their backlog of work and this results in a perceived "blocking" state or almost a "locked system". To validate this, on some systems, we replaced poor performing disks with known good disks and observed dramatically improved performance. In a few cases, updating the drive's firmware was sufficient to very noticeably improve responsiveness.

To reflect this real world learning, in the Windows 7 Beta code, we have capped scores for drives which appear to exhibit the problematic behavior (during the scoring) and are using our feedback system to send back information to us to further evaluate these results. Scores of 1.9, 2.0, 2.9 and 3.0 for the system disk are possible because of our current capping rules. Internally, we feel confident in the beta disk assessment and these caps based on the data we have observed so far. Of course, we expect to learn from data coming from the broader beta population and from feedback and conversations we have with drive manufacturers.

For those obtaining low disk scores but are otherwise satisfied with the performance, we aren't recommending any action (Of course the WEI is not a tool to recommend hardware changes of any kind). It is entirely possible that the sequence of I/Os being issued for your common workload and applications isn't encountering the issues we are noting. As we've said, the WEI is a metric but only you can apply that metric to your computing needs.

Earlier, I made note of the fact that our new levels, 6 and 7, were added to recognize the improved experiences one might have with newer hardware, particularly SSDs, graphics adapters, and multi-core processors. With respect to SSDs, the focus of the newer tests is on random I/O rates and their avoidance of the long latency issues noted above. As a note, the tests don't specifically check to see if the underlying storage device is an SSD or not. We run them no matter the device type and any device capable of sustaining very high random I/O rates will score well.

For graphics adapters, both DX9 and DX10 assessments can be run now. In Vista, the tests were specific to DX9. To obtain scores in the 6 or 7 ranges, a graphics adapter must obtain very good performance scores, support DX10 and the driver must be a WDDM 1.1 driver (which you might have noticed are being downloaded in beta during the Windows 7 beta). For WDDM 1.0 drivers, only the DX9 assessments will be run, thus capping the overall score at 5.9.

For multi-core processors, both single threaded and multi-threaded scenarios are run. With levels 6 and 7, we aim to indicate that these systems will be rarely CPU bound for typical use and quite suitable for demanding processing tasks and multi-tasking. As examples, we anticipate many quad core processors will be able to score in the high 6 to low 7 ranges, and 8 core systems to be able to approach 7.9. The scoring has taken into account the very latest micro-processors available.

For many key hardware partners, we've of course made available additional details on the changes and why they were made. We continue to actively work with them to incorporate appropriate feedback.

--Michael Fortin



# Follow-up: Accessibility in Windows 7

Steven Sinofsky | [2009-01-21T03:00:00+00:00](#)

---

*We've seen some comments recently posted on a previous [post](#) on accessibility and a member of the User Interface Platform team wanted to offer some thoughts on the topic. Brett is a senior test lead who leads our efforts testing the Accessibility of Windows 7. -- Steven*

Hi, my name is Brett and I am the test lead for the Windows 7 Accessibility team. Back in November my colleague Michael wrote a blog post about the work our team is doing for Windows 7, I'm following up to that and some recent comments about our new screen Magnifier. On a personal note I would like to mention that I'm a person with low vision and depend on some of the technologies that my team produces to help me in my work.

I've been using Windows 7 for my day-to-day work for several months, this is something we call "dogfooding", which is using our own pre-release products long before the public ever sees a beta. I've been using Windows 7 as my primary operating system and have found our new Magnifier to be very useful to me.

Now, about our Magnifier, as you can imagine, the appeal of the many features in Windows varies from person to person, we often say that it is like making pizza for a billion people. The same is true for the features my team owns. I've read many comments since we released our Windows 7 beta about magnifier, some are from people that have really benefited from our new work, some have suggestions, and others have concerns. I will say thanks for the feedback, we appreciate all types. Those of you that have benefited are mostly people that need basic magnification and appreciate the easy ability to zoom in and out as needed; I fall into this category myself. Those of you that need magnification in combination with custom colors, high-contrast or some screen readers probably haven't been able to benefit from the new Magnifier, for you we've made sure that the Vista magnifier continues to work. Let me explain a little more about what we've done in Windows 7.

To go into more detail about our implementation I need to start with our graphics system in Windows. Over the last several years GPU technology has made huge advances and in Vista we finally made the leap to a modern hardware accelerated graphics system, what we call Aero, which takes advantage of the GPU. We often use the term Aero to refer to the specific elements of Windows visuals, such as transparency and gradients. In practice it is more than that, the modern graphics rendering (technically the *desktop window manager* along with the DirectX APIs) is not just for aesthetics but for all forms of rendering including text, 2D, and 3D all using modern hardware assisted graphics and a much richer API. It takes time, however, for the diverse ecosystem to adopt this technology, perhaps even over the course of several OS releases. It also takes time for Windows and time for software developers and hardware manufacturers to adopt new technologies; so for a time we will have (and fully support) a mix of both old and new. For example, some screen readers do the great things they do by capturing the data that goes through the original Windows graphics system (GDI) and building their off-screen UI models which is why they need to turn off the new rendering. On the other hand, our new Magnifier is integrated deeply into the desktop window manager ("Aero") to leverage this graphics horsepower and deliver smooth full-screen multi-monitor magnification.

While, as this demonstrates, these advances aren't seamless, in Windows 7 my team has worked to make sure that we maintain Vista functionality and compatibility while making new investments. Magnifier is an example of this, we utilize the power of the GPU where we can to bring new capabilities to a broad spectrum of customers, and when Aero needs to be off, whether for screen readers, high-contrast or other needs, we maintain the existing capabilities in the product. And by maintaining compatibility as much as possible, many of the tools you depend on today will continue to work with Windows 7.

So, is Magnifier better for everyone? Not everyone, but certainly for many people, but more than that I can honestly say that we have made advances to accessibility for everyone in Window 7. As Michael noted in his posting, we invested in several areas, there's not only the Magnifier and on-screen keyboard work, there is also significant work to the underlying accessibility APIs. We also actively support the community and recently made a grant to NV Access to help them improve their open source screen reader support for Windows 7 and Internet Explorer 8.

Thanks for reading, and thanks for your comments,

-Brett

# Disk Defragmentation – Background and Engineering the Windows 7 Improvements

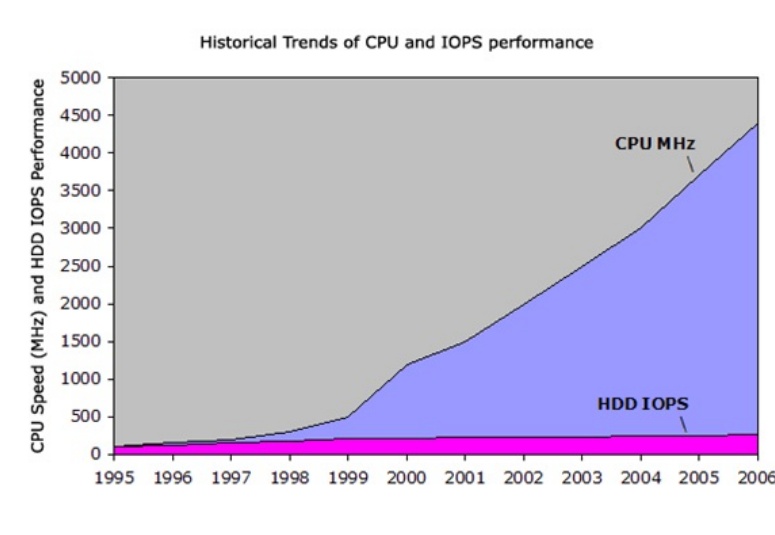
Steven Sinofsky | [2009-01-25T03:00:00+00:00](#)

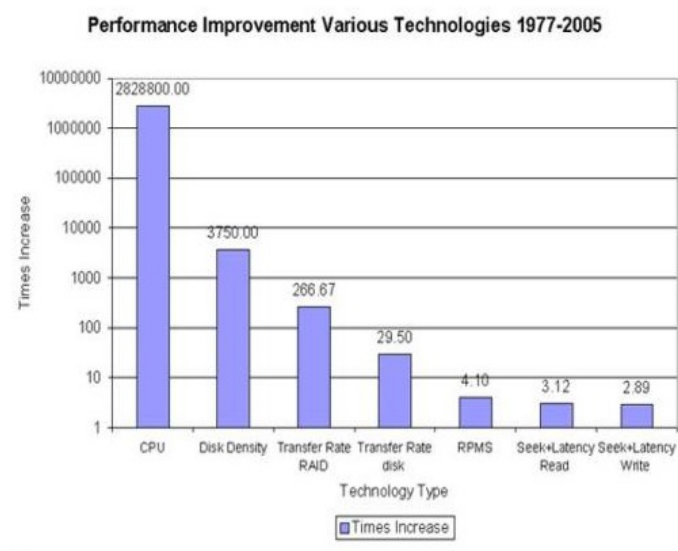
*One of the features that you've been pretty clear about (I've received over 100 emails on this topic!) is the desire to improve the disk defrag utility in Windows 7. We did. And from blogs we saw a few of you noticed, which is great. This is not as straight forward as it may appear. We know there's a lot of history in defrag and how "back in the day" it was a very significant performance issue and also a big mystery to most people. So many folks came to know that if your machine is slow you had to go through the top-secret defrag process. In Windows Vista we decided to just put the process on autopilot with the intent that you'd never have to worry about it. In practice this turns out to be true, at least to the limits of automatically running a process (that is if you turn your machine off every night then it will never run). We received a lot of feedback from knowledgeable folks wanting more information on defrag status, especially during execution, as well as more flexibility in terms of the overall management of the process. This post will detail the changes we made based on that feedback. In reading the mail and comments we received, we also thought it would be valuable to go into a little bit more detail about the process, the perceptions and reality of performance gains, as well as the specific improvements. This post is by Rajeev Nagar and Matt Garson, both are Program Managers on our File System feature team. --Steven*

In this blog, we focus on disk defragmentation in Windows 7. Before we discuss the changes introduced in Windows 7, let's chat a bit about what fragmentation is, and its applicability.

Within the storage and memory hierarchy comprising the hardware pipeline between the hard disk and CPU, hard disks are relatively slower and have relatively higher latency. Read/write times from and to a hard disk are measured in milliseconds (typically, 2-5 ms) – which sounds quite fast until compared to a 2GHz CPU that can compute data in less than 10 nanoseconds (on average), once the data is in the L1 memory cache of the processor.

This performance gap has only been increasing over the past 2 decades – the figures below are noteworthy.





In short, the figures illustrate that while disk capacities are increasing, their ability to transfer data or write new data is not increasing at an equivalent rate – so disks contain **more** data that takes **longer** to read or write. Consequently, fast CPUs are relatively idle, waiting for data to do work on.

Significant research in Computer Science has focused on improving overall system I/O performance, which has led to two principles that the operating system tries to follow:

1. Perform less I/O, i.e. try and minimize the number of times a disk read or write request is issued.
2. When I/O is issued, transfer data in relatively large chunks, i.e. read or write in bulk.

Both rules have reasonably simply understood rationale:

1. Each time an I/O is issued by the CPU, multiple software and hardware components have to do work to satisfy the request. This contributes toward increased latency, i.e., the amount of time until the request is satisfied. This latency is often directly experienced by users when reading data and leads to increased user frustration if expectations are not met.
2. Movement of mechanical parts contributes substantially to incurred latency. For hard disks, the “*rotational time*” (time taken for the disk platter to rotate in order to get the right portion of the disk positioned under the disk head) and the “*seek time*” (time taken by the head to move so that it is positioned to be able to read/write the targeted track) are the two major culprits. By reading or writing in large chunks, the incurred costs are amortized over the larger amount of data that is transferred – in other words, the “per unit” data transfer costs decrease.

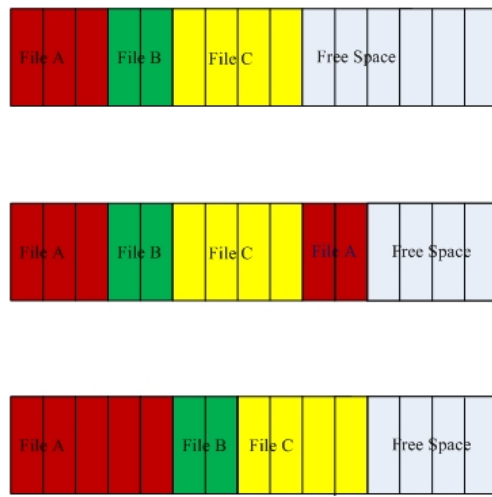
File systems such as NTFS work quite hard to try and satisfy the above rules. As an example, consider the case when I listen to the song “Hotel California” by the Eagles (one of my all time favorite bands). When I first save the 5MB file to my NTFS volume, the file system will try and find enough contiguous free space to be able to place the 5MB of data “together” on the disk. Since logically related data (e.g. contents of the same file or directory) is more likely to be read or written around the same time. For example, I would typically play the entire song “Hotel California” and not just a portion of it. During the 3 minutes that the song is playing, the computer would be fetching portions of this “related content” (i.e. sub-portions of the file) from the disk until the entire file is consumed. By making sure the data is placed together, the system can issue read requests in larger chunks (often pre-reading data in anticipation that it will soon be used) which, in turn, will minimize mechanical movement of hard disk drive components and also ensure fewer issued I/Os.

Given that the file system tries to place data contiguously, when does fragmentation occur? Modifications to stored data (e.g. adding, changing, or deleting content) cause changes in the on-disk data layout and can result in fragmentation. For example, file deletion naturally causes space de-allocation and resultant “holes” in the allocated space map – a condition we will refer to as “fragmentation of available free space”. Over time, contiguous free space becomes harder to find leading to fragmentation of newly stored content. Obviously, deletion is not the only cause of fragmentation – as mentioned above, other file operations such as modifying content in place or appending data to an existing file can eventually lead to the same condition.

So how does defragmentation help? In essence, defragmentation helps by moving data around so that it is once again placed more optimally on the hard disk, providing the following benefits:

1. Any logically related content that was fragmented can be placed adjacently
2. Free space can be coalesced so that new content written to the disk can be done so efficiently

The following diagram will help illustrate what we’re discussing. The first illustration represents an ideal state of a disk – there are 3 files, A, B, and C, and all are stored in contiguous locations; there is no fragmentation. The second illustration represents a fragmented disk – a portion of data associated with File A is now located in a non-contiguous location (due to growth of the file). The third illustration shows how data on the disk would look like once the disk was defragmented.



Nearly all modern file systems support defragmentation – the differences generally are in the defragmentation mechanism, whether, as in Windows, it’s a separate, schedulable task or, whether the mechanism is more implicitly managed and internal to the file system. The design decisions simply reflect the particular design goals of the system and the necessary tradeoffs. Furthermore, it’s unlikely that a general-purpose file system could be designed such that fragmentation never occurred.

Over the years, defragmentation has been given a lot of emphasis because, historically, fragmentation was a problem that could have more significant impact. In the early days of personal computing, when disk capacities were measured in megabytes, disks got full faster and fragmentation occurred more often. Further, memory caches were significantly limited and system responsiveness was increasingly predicated on disk I/O performance. This got to a point that some users ran their defrag tool weekly or even more often! Today, very large disk drives are available cheaply and % disk utilization for the average consumer is likely to be lower causing relatively less fragmentation. Further, computers can utilize more RAM cheaply (often, enough to be able to cache the data set actively in use). That together, with improvements in file system allocation strategies as well as caching and pre-fetching algorithms, further helps improve overall responsiveness. Therefore, while the performance gap between the CPU and disks continues to grow and fragmentation does occur, combined hardware and software advances in other areas allow Windows to mitigate fragmentation impact and deliver better responsiveness.

So, how would we evaluate fragmentation given today's software and hardware? A first question might be: *how often does fragmentation actually occur and to what extent?* After all, 500GB of data with 1% fragmentation is significantly different than 500GB with 50% fragmentation. Secondly, *what is the actual performance penalty of fragmentation, given today's hardware and software?* Quite a few of you likely remember various products introduced over the past two decades offering various performance enhancements (e.g. RAM defragmentation, disk compression, etc.), many of which have since become obsolete due to hardware and software advances.

The incidence and extent of fragmentation in average home computers varies quite a bit depending on available disk capacity, disk consumption, and usage patterns. In other words, there is no general answer. The actual performance impact of fragmentation is the more interesting question but even more complex to accurately quantify. A meaningful evaluation of the performance penalty of fragmentation would require the following:

- Availability of a system that has been "aged" to create fragmentation in a typical or representative manner. But, as noted above, there is no single, representative behavior. For example, the frequency and extent of fragmentation on a computer used primarily for web browsing will be different than a computer used as a file server.
- Selection of meaningful disk-bound metrics e.g. boot and first-time application launch post boot.
- Repeated measurements that can be statistically relevant

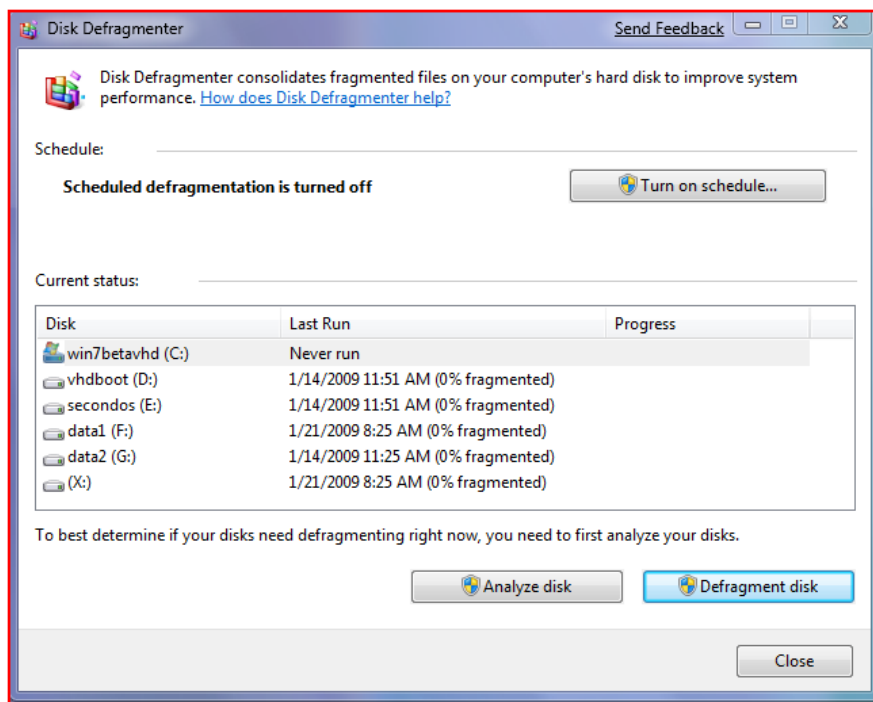
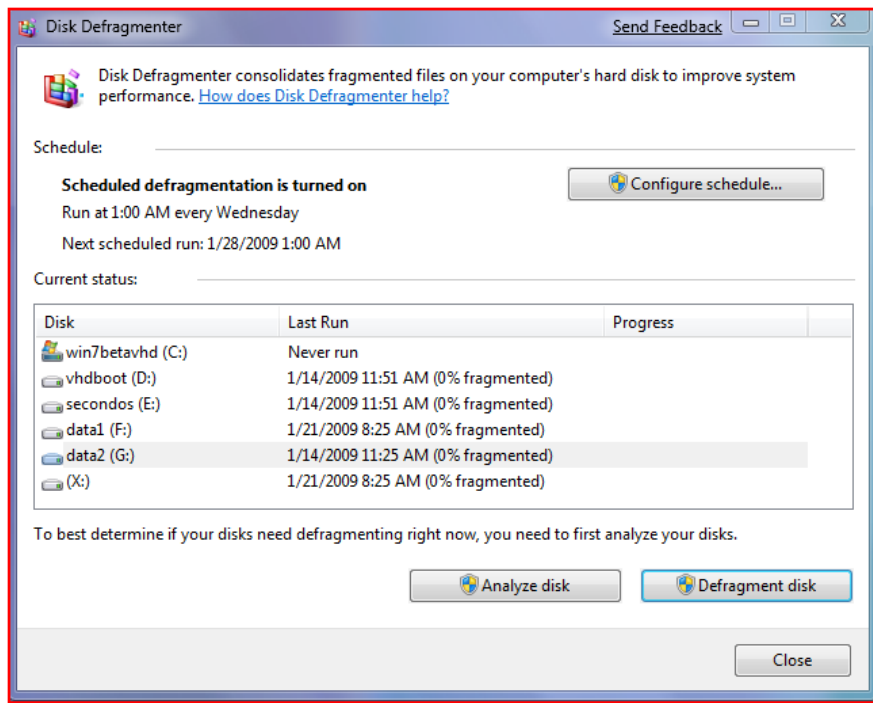
Let's walk through an example that helps illustrate the complexity in directly correlating extent of fragmentation with user-visible performance.

In Windows XP, any file that is split into more than one piece is considered fragmented. Not so in Windows Vista if the fragments are large enough – the defragmentation algorithm was changed (from Windows XP) to ignore pieces of a file that are larger than 64MB. As a result, defrag in XP and defrag in Vista will report different amounts of fragmentation on a volume. So, which one is correct? Well, before the question can be answered we must understand why defrag in Vista was changed. In Vista, we analyzed the impact of defragmentation and determined that the most significant performance gains from defrag are when pieces of files are combined into *sufficiently large chunks* such that the impact of disk-seek latency is not significant relative to the latency associated with sequentially reading the file. This means that there is a point after which combining fragmented pieces of files has no discernible benefit. In fact, there are actually negative consequences of doing so. For example, for defrag to combine fragments that are 64MB or larger requires significant amounts of disk I/O, which is against the principle of minimizing I/O that we discussed earlier (since it decreases total available disk bandwidth for user initiated I/O), and puts more pressure on the system to find large, contiguous blocks of free space. Here is a scenario where a certain amount of fragmentation of data is just fine – doing nothing to decrease this fragmentation turns out to be the right answer!

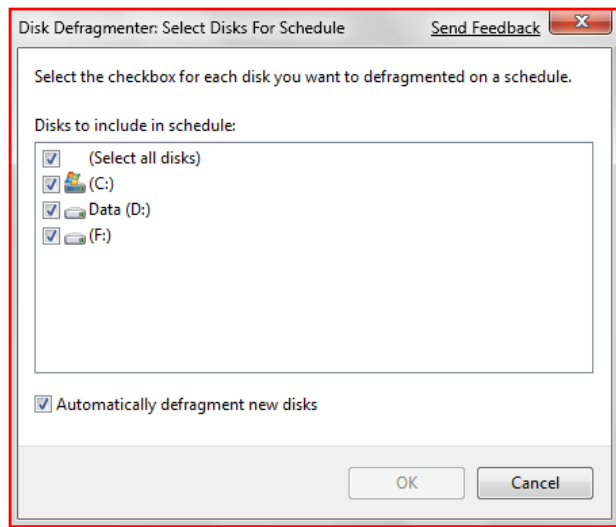
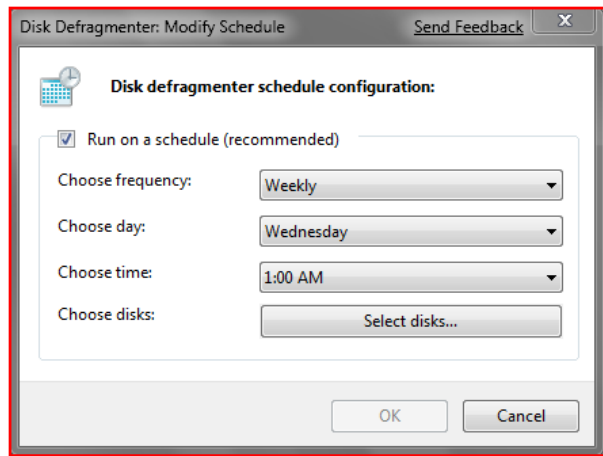
Note that a concept that is relatively simple to understand, such as the amount of fragmentation and its impact, is in reality much more complex, and its real impact requires comprehensive evaluation of the entire system to accurately address. The different design decisions across Windows XP and Vista reflect this evaluation of the typical hardware & software environment used by customers. Ultimately, when thinking about defragmentation, it is important to realize that there are many additional factors contributing towards system responsiveness that must be considered beyond a simple count of existing fragments.

The defragmentation engine and experience in Windows 7 has been revamped based on continuous and holistic analysis of impact on system responsiveness:

In Windows Vista, we had removed all of the UI that would provide detailed defragmentation status. We received feedback that you didn't like this decision, so we listened, evaluated the various tradeoffs, and have built a new GUI for defrag! As a result, in Windows 7, you can monitor status more easily and intuitively. Further, defragmentation can be safely terminated any time during the process and on all volumes very simply (if required). The two screenshots below illustrate the ease-of-monitoring:



In Windows XP, defragmentation had to be a user-initiated (manual) activity i.e. it could not be scheduled. Windows Vista added the capability to schedule defragmentation – however, only one volume could be defragmented at any given time. Windows 7 removes this restriction – multiple volumes can now be defragmented in parallel with no more waiting for one volume to be defragmented before initiating the same operation on some other volume! The screen shot below shows how defragmentation can be concurrently scheduled on multiple volumes:



Among the other changes under the hood in Windows 7 are the following:

- Defragmentation in Windows 7 is more comprehensive – many files that could not be re-located in Windows Vista or earlier versions can now be optimally re-placed. In particular, a lot of work was done to make various NTFS metadata files movable. This ability to relocate NTFS metadata files also benefits volume shrink, since it enables the system to pack all files and file system metadata more closely and free up space “at the end” which can be reclaimed if required.
- If solid-state media is detected, Windows disables defragmentation on that disk. The physical nature of solid-state media is such that defragmentation is not needed and in fact, could decrease overall media lifetime in certain cases.
- By default, defragmentation is disabled on Windows Server 2008 R2 (the Windows 7 server release). Given the variability of server workloads, defragmentation should be enabled and scheduled only by an administrator who understands those workloads.

Best practices for using defragmentation in Windows 7 are simple – you do not need to do anything! Defragmentation is scheduled to automatically run periodically and in the background with minimal impact to foreground activity. This ensures that data on your hard disk drives is efficiently placed so the system can provide optimal responsiveness and I can continue to enjoy glitch free listening to the Eagles :-).

Rajeev and Matt



# Showcasing Windows 7 Platform with Applets

Steven Sinofsky | [2009-01-28T03:00:00+00:00](#)

---

*About every decade we make the big decision to update what we refer to as the applets (note we'll use applet, application, program, and tool all interchangeably as we write about these) in Windows—historically Calc (Calculator), Paint (or MS Paint, Paint Brush) and WordPad (or Write), and also the new Sticky Notes applet in Windows 7. As an old-timer, whenever I think of these tools I think of all the history behind them and how they came about. I'm sure many folks have seen the now "classic" [video](#) of our (now) CEO showing off Windows to our sales force (the last word of this video is the clue that this video was done for inside Microsoft). Windows 7 seems like a great time to update these tools. The motivation for updating the applets this release is not the 10-year mark or just time to add some applet-specific features, but the new opportunities for developers to integrate their applications with the Windows 7 desktop experience. While many use the applets as primary tools, our view of these is much more about demonstrating the overall platform experience and to provide guidance to developers about how to integrate and build on Windows 7, while at the same time providing "out of box" value for everyone. There's no real "tension" over adding more and more features to these tools as our primary focus is on showing off what's new in Windows—after all there are many full-featured tools available that provide similar functionality for free. So let's not fill the comments with request for more bitmap editing features or advanced scientific calculator features :-).*

*The APIs discussed in this post are all described on MSDN in the updated developer area for Windows 7 where you can find the [Windows 7 developer guide](#). Each of the areas discussed is also supported by the [PDC](#) and [WinHEC](#) sessions on those sites.*

*This post was written by several folks on our applications and gadgets team with Riyaz Pishori, the group program manager, leading the effort. –Steven*

This blog post discusses some of the platform innovations in Windows 7 and how Windows 7 applications have adopted and showcased these innovations. This post details some of the platform features that developers and partners can expect in Windows 7 and how Windows 7 programs have showcased these innovations. This post also discusses how applications have been given a facelift both in terms of their functionality as well as their user experience by focusing on key Windows design principles and platform innovations. Finally, this post can serve as a pointer or guide to application developers and ISVs to get themselves familiar with some of the key new Windows platform innovations, see them in action and then figure out how they can build on these APIs for their own software.

The post is organized by each subsystem, and how Windows applets are using that particular subsystem.

## Windows Ribbon

The Windows Ribbon User Interface is the next generation rich, new user interface for Windows development. The Windows Ribbon brings the now familiar Office 2007 Ribbon user interface to Windows 7, making it available to application developers and ISVs.

There are several advantages of adopting the Windows Ribbon user interface, many of which have been talked about in the Office 2007 [blogs](#). The Ribbon provides a rich, graphical user interface for all commands in a single place, without the need to expose various functions and commands under different menus or toolbars. The Ribbon UI is direct and self-explanatory, and has a labelled grouping of logically related commands. While using an application that is built on the Ribbon UI platform, the user only needs to focus on his workflow and the context of his task, rather than worry about where a particular function is located or accessible. The Ribbon UI also takes care of layout and provides consistency as compared to toolbars which the user can customize in terms of their sizes, location and contents. It also has built-in and improved keyboard accessibility, and making the application DPI and theme aware becomes easier by using the Ribbon. Finally, development and changes to the user interface is very quick and rapid due to the XML-markup based programming model for the Ribbon User Interface.

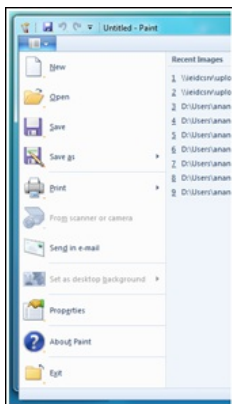
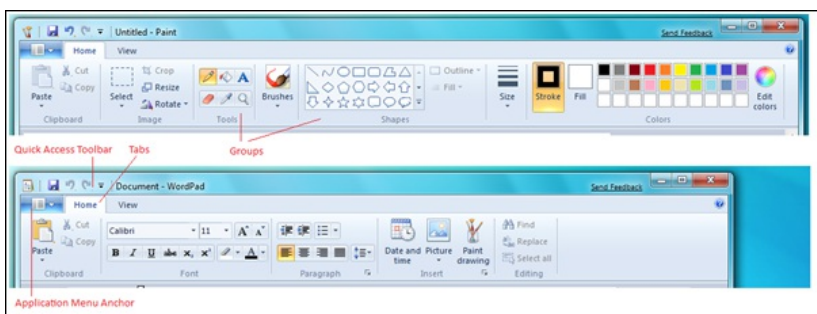
Paint and Wordpad are two of the first consumers of the Windows Ribbon UI Platform. In Windows 7, both these applications are enhanced with

a set of new features, and the user interface of these applications also required to be brought up to the Windows 7 experience and standards. The Windows Ribbon UI is a great fit for these applications to revamp their user experience and make it consistent, and make these applications rich, fun and easy to use. The tasks and commands in these applications were amenable to be applied to the Ribbon UI framework, and it also served as an opportunity for popular native Windows applications to showcase the Windows Ribbon UI platform to consumers, as well as developers and ISVs. Many has asked about the Windows Explorer and IE also using the ribbon, which we did not plan on for Windows 7. Our Windows 7 focus was on the platform and demonstrating the platform for document-centric applications such as Paint and Wordpad.

Both these applications showcase several elements of the Windows UI Ribbon. The Application Menu of both Paint and Wordpad exposes Application related commands that are typically available thru the 'File' menu of an application. Both the applications have a core tab set that consists of 'Home', which exposes most of the commands in the application, and 'View' which exposes the image or document viewing options in the application. The commands in both these tabs are laid out logically in groups of related functionality.

A quick access toolbar (QAT) is provided by both Paint and Wordpad, which comes with certain defaults like Save, Undo and Redo that are meaningful to the application. The user can customize the QAT by using the QAT drop-down, or right-click on any command or group in the ribbon and add it to the QAT.

Several ribbon commands are used in both these applications, like command buttons, split buttons, galleries, drop-downs, check boxes and toggle buttons.



Further, both applications provide a 'Print preview' mode which shows a print preview of the image or the document in context. In a mode, all the core tabs are removed and only the mode is displayed for the user to interact with. On exiting a mode, the user is returned to the core tab set.

Paint also exposes a contextual tab for the Text tool, which is displayed only when a text control is drawn on canvas. A contextual tab shown next to the core tab set when the text tool is in focus, and removed when the text is applied to the image on the canvas. The contextual tab set contains the tools that are specific and relevant only to the text tool.

Both the applications provide live previews through ribbon galleries, for example the font size and font name for Wordpad and Paint while formatting text, bullets and lists in Wordpad, and color selection, outline size selection and outline and fill styles for shapes in Paint. A live preview allows the user to see the changes instantaneously on mouse hover, and then apply those changes on a selection. These previews are one of the key elements of the ribbon UI and demonstrate why the metaphor is much more than a “big toolbar” but a new interaction style.

By adopting the Ribbon User Interface, both the applications inherit built-in keyboard accessibility support using ribbon Keytips, have tooltips on all commands, and have ready support for DPI and Windows themes.

Paint and Wordpad can serve as examples of how the Ribbon UI can be easily used in MFC applications. The Windows Ribbon presents new opportunities and options for developers and ISVs to develop applications with the Ribbon User Interface. The Windows Scenic Ribbon programming model and architecture emphasizes the separation of the markup file and the C++ code files to help developers decouple the presentation and customization of the UI from the underlying application code. The platform also promotes developer-designer workflow, where the developer can focus on the application logic, while the designer can work on the UI presentation and layout. The ribbon UI is a significant investment for us and you should expect to see us continue to use it more throughout Microsoft, including an implementation in the .NET Framework as was demonstrated by Scott Guthrie at the PDC, which will be built on Windows 7 natively in the future.

## Multi-touch platform

Windows 7 provides support for multi-touch input data, as well as supporting multi-touch in Win32 via Windows messages. The investments in the multi-touch platform include the developer platform that exposes touch APIs to applications, enhancing the core user interface in Windows 7 to optimize for touch experiences, and providing multi-touch gestures for applications to consume. Developers on Windows 7 can build on these APIs decide on the appropriate level of touch support they would like to provide in their software.

Wordpad enhances the document reading experience by using the multi-touch platform and using the zoom and pan gestures. Zooming, panning and inertia lets the user get to a particular piece of content very quickly in an intuitive fashion. By using the zoom gesture, the user can zoom in or zoom out of the document which is akin to using the zoom slider at the right of the Wordpad status bar. On multi-touch capable hardware, the user can zoom in and out of the document by placing his fingers anywhere within the document window and executing the zoom gesture. Wordpad also supports the pan gesture to pan thru the pages of a document that is open in Wordpad. By executing the pan gesture, the user can scroll-down or scroll-up a document similar to using the scroll bar of the Wordpad application.

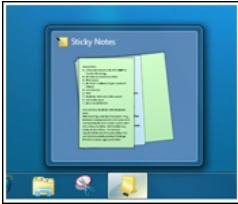
In Paint multi-touch data is used to allow users to paint with multiple fingers. It is an example of an application that allows multi-touch input without the usage of gestures. For Paint’s functionality, providing multiple finger painting ability was more compelling and enriching than allowing for zoom, pan, rotate or other gestures that act on the picture in a read-only mode and not in an edit-mode. New brushes in Paint are multi-touch enabled, and they handle touch inputs via multiple fingers and allow the user to simultaneously draw strokes on canvas on finger drag. These brushes are also pressure-sensitive, thereby providing a realistic experience with touch by varying the stroke width based on the pressure on the screen. While adopting the multi-touch platform to enhance the end-user experience in Paint, conscious design decisions were made to preserve the single touch experience for functionalities where a multi-touch scenario does not apply such as the color picker, magnifier and text tool.

By building with the multi-touch APIs, Paint and Wordpad have created more natural and intuitive interfaces on touch-enabled hardware and show “out of the box” how different capabilities can be exposed by developers in their software.

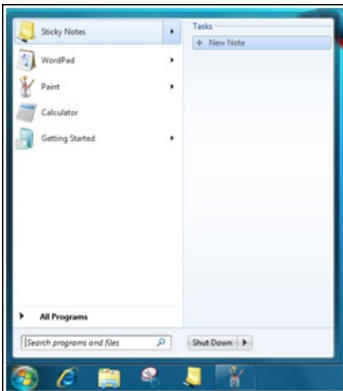
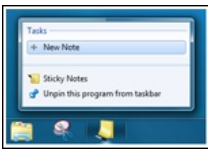
## Taskbar

Sticky Notes (or just Notes) is an extension of a TabletPC applet available in Windows 7. One of the things which was key to the Notes experience on the desktop was to have the ability to quickly take all the notes away and get them back, but still making sure it is really easy to create a new note. We achieved this by having a single top level window for the sticky notes application. You can minimize all your notes and view a stack of notes in the preview on the command bar with a single click. The stacked preview has been achieved using the new thumbnail preview

APIs that enable apps to override the default taskbar previews that are essentially a redirected snapshot of the top level application window, and provide their own. This enables applications to decouple their previews from the top level application window and provide a more productive preview based on the scenario. For example, this was very valuable in Sticky Note scenarios where a quick peek at a note that was last touched provides for quite a productive workflow. Taskbar also caches the preview thumbnail images so once the preview is given to the Taskbar, the application does not need to keep it around – the application does however need to send an updated preview whenever it changes.



Another nifty customization end-point on the task bar is the destination menu (aka jumplist). This menu comes up when a user right clicks on the application in the taskbar or hovers over the application icon in the Start Menu. The Sticky Notes application does not have a single main application window – this makes the application feel really light weight and fits in well into the Windows 7 philosophy of creating simple and powerful user experiences. The challenge then was exposing functionality such as the ability to create a new note from a central location or potentially other custom “tasks”. The destination menu helped exposing these scenarios in a simple yet discoverable way.

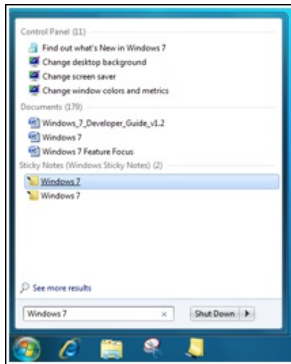


The new taskbar functionality and extensibility built in that has the potential to make it a lot easier for people to work with applications/scenarios in a more productive and efficient manner when developers integrate their software with the Windows desktop.

## Search

Building on the long history of Search in Windows and the significant enhancements in Windows 7, there are APIs available to developers to deeply integrate their content types with the desktop search user experience affordances in Windows 7. Sticky Notes shows one example of how these APIs can be used.

The Sticky Notes application now allows users to get back to their notes by simply searching for content through Windows Inline search within the start menu. This is in line with allowing users to reach the relevant note as quickly as possible even when the application is closed. Even though search could be done for both Text and Ink content, it is restricted to text because of lower success rates with varied handwriting styles in ink. The application registers a protocol handler that generates a URL for each Note. The Sticky Notes Filter handler gets asked for the content associated with each note that is then indexed by the Search infrastructure. These indexes are then used to perform a quick lookups when the user searches the Search interfaces provided by the Windows Shell. When a user clicks on a result, Search invokes the associated application with the URL corresponding to the one that the protocol handler had generated that the Filter handler associated with the content it sent to the Search indexer.



The search platform also has the ability to enable the filter handler to specify the language of each chunk of content passed on to it that overrides the default Search heuristics used to compute the language - this increases Search accuracy manifold and thereby enhances internationalization support of the entire ecosystem.

The reason Sticky Notes implemented a protocol handler in addition to a Filter handler was because it implements its own integrated storage schema on top of the Windows File system - all the notes are represented by a single .snt file. The protocol handler generates URL's to individual entities (in this case - notes); the filter handler picks out content for each of these URL's and gives it to Search for indexing.

This demonstrates the ease in which applications can plug into the search platform in Windows 7, and add search handlers which can enhance the overall user experience from the App as well as the platform.

## Real-Time Stylus

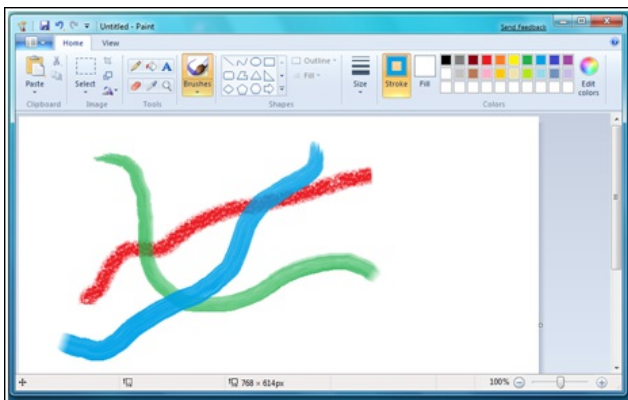
Real-Time Stylus (RTS) is infrastructure that provides access to the stylus events coming from pen or touches digitizers. It provides information about strokes and points and provides access to ink-related events. Using RTS, applications can get access to stylus information and develop compelling end-user scenarios and experiences.

Sticky Notes now allows the users to Ink and Type on notes depending on the availability of inking hardware. Users can use keyboard input to type on notes and use the stylus to ink on notes. Though the experience has been designed keeping in mind that users will either use either ink or text on a particular note, it does allow users to ink and type on the same note. However these surfaces are maintained agnostic to each other. Sticky Notes also auto-grows the note while inking on the note, providing a real-time experience of the note adjusting its size to fit the inked content.

Real Time Stylus (RTS) is used for inking features provided in Sticky Notes. Inking gestures are also available to applications, and the scratch out gesture has been implemented in Sticky Notes to delete content.



In addition, Paint uses RTS to get a stream of positional input from mouse, stylus or touch which are used for drawing strokes on canvas. Paint also captures additional input variables like pressure and touch surface area when such input is available from the digitizer, and maps these inputs into the stroke algorithms that are used to generate Paint strokes on canvas. Using this algorithm, the user is able to modulate stroke width and other parameter based on the pressure or touch area on canvas.



Using RTS allows the development of applications and software that can build on the inking platform and provide ways to interact with the application that go beyond mouse or keyboard. Using stylus, inking and gestures, developers can create interactive experiences for end-users.

## Restart and Recovery

The Windows Error Reporting (WER) infrastructure is a set of feedback technologies that is built into Windows 7 and other earlier versions of Windows client and server. WER allows applications to register for application failures and capture this data for end-users who agree to report it. This data can be accessed and analyzed and can be used to monitor error trends and download debug information to help developers and ISVs determine the root cause for application failures.

WER can add value to software development at various stages: during development, during beta testing by getting early feedback from end-users, after the release of the product by analysing and prioritizing the top fixes, and at end of life of the product.

Related to failure recovery, Applications can also register with WER for restart on application of a Windows patch that terminates the application and on application of an update that reboots the computer, as well as failure caused due to an application crash or hang or not responding state. Applications can optionally register for recovery of lost data, can develop their own mechanism for recovery.

Several Windows applications adopt the WER infrastructure to collect and analyze data. Calculator, Paint and Wordpad register for restart and additionally recover the current data in the sessions of the application that were running. Sticky Notes also registers for restart and recovery, and returns the user to the set of notes open on the desktop. Using WER, end-users would allow Windows to capture and collect problem data and then would be returned to the applications in the same state that they were in earlier.

As you can see, our primary effort for the applets in Windows 7 is to showcase some of the new platform APIs and innovations available to developers. As you get to try out these applications you will see that while showcasing the Windows 7 platform innovations, we have also added some commonly requested features and functionality. Some of them are: Check and correct, calculation modes and templates in **Calculator**, New brushes, shapes and multi-touch support in **Paint**, Open standards support in **Wordpad** and Ink and text, taskbar and search integration in **Sticky notes**. Maybe we won't wait 10 years to update these again ??

--Riyaz Pishori and team

# Our Next Engineering Milestone

Steven Sinofsky | [2009-01-30T03:00:00+00:00](#)

---

Many posts start with a thank you and I want to start this post with an extra special thank you on behalf of the entire Windows team for all the installs and usage we are seeing of the Windows 7 Beta. We've had millions of installations of Windows 7 from which we are receiving telemetry, which is simply incredible. And from those who click on the "Send Feedback" button we are receiving detailed bug reports and of course many suggestions. There is simply no way we could move from Beta through Final Release of Windows 7 without this type of breadth coverage and engagement from you in the development cycle. There's been such an incredible response, with many folks even blogging about how they have moved to using Windows 7 Beta on all their machines and have been super happy. The question we get most often is "if the Beta expires in August what will I do—I don't want to return to my *old* [sic] operating system." For a Beta release, that is quite a complement and we're *very* appreciative of such a kind response.

This post is about the path from where we are today, Beta, to our RTM (Release To Manufacturing), building on the discussion of this topic that started at the PDC. *This post is in no way an announcement of a ship date, change in plans, or change in our previously described process, but rather it provides additional detail and a forward looking view of the path to RTM and General Availability.* The motivation for this, in addition to the high level of interest in Windows 7, is that we're now seeing how releasing Windows is not something that Microsoft does "solo", but rather is something that we do as one part of the overall PC ecosystem. Obviously we have a big responsibility to do our part, one we take very seriously of course. The last stages of a Windows release are a partnership across the entire ecosystem working to make sure that the incredible variety of choices you have for PCs, software, and peripherals work together to bring you a complete and satisfying Windows 7 experience.

The next milestone for the development of Windows 7 is the Release Candidate or "RC". Historically the Release Candidate has signaled "we're pretty close and we want people to start testing the release, especially because all the features are done." As we have said before, with Windows 7 we chose a slightly different approach which we were clear up front about and are all now experiencing together and out in the open. The *Pre-Beta* from the PDC was a release where we said it was substantially API complete and even for the areas that were not in the release we detailed the APIs and experience in the sessions at the PDC. At that time we announced that the Beta test in early 2009 would be both API and feature complete, widely available, and would be the only Beta test. We continued this dialog with our hardware partners at WinHEC. We also said that many ecosystem partners including PC makers, software vendors, hardware makers will, as has been the case, continue to receive interim builds on a regular basis. This is where we stand today. We've released the feature complete Beta and have made it available broadly around the world (though we know folks have requested even more languages). As a development team we're doing just what many of you do, which is choosing to run the Beta full time on many PCs at home and work (personally I have at least 9 different machines running it full time) and we're running it on many thousands of individual's machines inside Microsoft, and thousands of machines in our labs as well.

All the folks running the Beta are actively contributing to fixing it. We're getting performance telemetry, application compatibility data, usage information, and details on device requirements among other areas. This data is very structured and very actionable. We have very high-bandwidth relationships with partners and good tools to help each other to deliver a great experience. One thing you might be seeing is that hardware and software vendors might be trying out updated drivers / software enhanced for Windows 7. For example, many of the anti-virus vendors already have released compatibility packs or updates that are automatically applied to your running installation. You might notice, for example, that many GPU chipsets are being recognized and Windows 7 downloads the updated WDDM 1.1 drivers. While the Windows Vista drivers work as expected, the new 1.1 drivers provide enhanced performance and a reduced memory footprint, which can make a big difference on 1GB shared memory machines. You might insert a device and receive a recently updated version of a driver as I did for a Logitech QuickCam. Another example some of you might have seen is that the Beta requires an updated version of Skype software currently in testing. When you go to install the old version you get an error message and then the problem and solutions user interface kicks in and you are redirected to the Beta site. This type of error handling is deployed in real time as we learn more and as the ecosystem builds out support. It is only because of our partnerships across the ecosystem that such efforts are possible, even during the Beta.

Of course, it is worth reiterating that our design point is that devices and software that work on Windows Vista and are still supported by the manufacturer will work on Windows 7 with the same software. There are classes of software and devices that are Windows-version specific for a variety of reasons, as we have talked about, and we continue to work together to deliver great solutions for Windows 7. The ability to provide people the vast array of choices and the openness of the Windows platform make all of this a massive undertaking. We continue to work to improve this while also making sure we provide the opportunities for choice and differentiation that are critical to the health and variety of the overall ecosystem. This data and the work we're doing together with partners is the critical work going on now to reach the Release Candidate phase.

We're also looking carefully at all the quality metrics we gather during the Beta. We investigate crashes, hangs, app compat issues, and also real-



world performance of key scenarios. A very significant portion of our effort from Beta to RC is focused on exclusively on quality and performance. We want to fix bugs experienced by customers in real usage as well as our broad base of test suites and automation. A key part of this work is to fix the bugs that people really encounter and we do so by focusing our efforts on the data we receive to drive the ordering and priority of which bugs to fix. As Internet Explorer has moved to Release Candidate, we've seen this at work and also read about it on [IE Blog](#).

Of course the other work we're doing is refining the final product based on all the real-world usage and feedback. We've received a lot of verbatim feedback regarding the user experience—whether that is default settings, keyboard shortcuts, or desired options to name a few things. Needless to say just working through, structuring, and “tallying” this feedback is a massive undertaking and we have folks dedicated to doing just that. At the peak we were receiving one “Send Feedback” note every 15 seconds! As we've talked about in this blog, we receive a lot of feedback where we must weigh the opinions we receive because we hear from all sides of an issue—that's to be expected and really the core design challenge. We also receive feedback where we thought something was straight forward or would work fine, but in practice needed some tuning and refinement. Over the next weeks we'll be blogging about some of these specific changes to the product. These changes are part of the process and part of the time we have scheduled between Beta and RC.

So right now, every day we are researching issues, resolving them, and making sure those resolutions did not cause regressions (in performance, behavior, compatibility, or reliability). The path to Release Candidate is all about getting the product to a known and shippable state both from an internal and external (Beta usage and partner ecosystem readiness) standpoint.

We will then provide the Release Candidate as a refresh for the Beta. We expect, based on our experience with the Beta, a broad set of folks to be pretty interested in trying it out.

With the RC, this process of feedback based on telemetry then repeats itself. However at this milestone we will be very selective about what changes we make between the Release Candidate and the final product, and very clear in communicating them. We will act on the most critical issues. The point of the Release Candidate is to make sure everyone is ready for the release and that there is time between the Release Candidate and our release to PC makers and manufacturing to validate all the work that has gone on since the pre-Beta. Again, we expect very few changes to the code. We often “joke” that this is the point of lowest productivity for the development team because we all come to work focused on the product but we write almost no code. That's the way it has to be—the ship is on the launch pad and all the tools are put away in the toolbox to be used only in case of the most critical issues.

As stated up front, this is a partnership and the main thing going on during this phase of the project is really about ecosystem readiness. PC makers, software vendors, hardware makers all have their own lead times. The time to prepare new products, new configurations, software updates, and all the collateral that goes with that means that Windows 7 cannot hit the streets (so to speak) until everyone has time to be ready together. Think of all those web sites, download pages, how-to articles, training materials, and peripheral packages that need to be created—this takes time and knowing that the Release Candidate is the final code that we're all testing out in the open is reassuring for the ecosystem. Our goal is that by being deliberate, predictable, and reliable, the full PC experience is available to customers.

We also continue to build out our compatibility lists, starting with logo products, so that our <http://www.microsoft.com/windows/compatibility> site is a good resource for people starting with availability. All of these come together with the PC makers creating complete “images” of Windows 7 PCs, including the full software, hardware, and driver loads. This is sort of a rehearsal for the next steps.

At that point the product is ready for release and that's just what we will do. We might even follow that up with a bit of a celebration!

There's one extra step which is what we call *General Availability* or GA. This step is really the time it takes literally to “fill the channel” with Windows PCs that are pre-loaded with Windows 7 and stock the stores (online or in-person) with software. We know many folks would like us to make the RTM software available right away for download, but this release will follow our more established pattern. GA also allows us time to complete the localization and ready Windows for a truly worldwide delivery in a relatively small window of time, a smaller window for Windows 7 than any previous release. It is worth noting that the Release Candidate will continue to function long enough so no one should worry and everyone should feel free to keep running the Release Candidate.

So to summarize briefly:

- **Pre-Beta** – This release at the PDC introduced the developer community to Windows 7 and represents the platform complete release and disclosure of the features.
- **Beta** – This release provided a couple of million folks the opportunity to use feature complete Windows 7 while also providing the telemetry and feedback necessary for us to validate the quality, reliability, compatibility, and experience of Windows 7. As we said, we are working with our partners across the ecosystem to make sure that testing and validation and development of Windows 7-based products begins to enter final phases as we move through the Beta.
- **Release Candidate (RC)** – This release will be Windows 7 as we intend to ship it. We will continue to listen to feedback and telemetry with the focus on addressing only the most critical issues that arise. We will be very clear in communicating any changes that have a visible impact on the product. This release allows the whole ecosystem to reach a known state together and make sure that we are all ready together for the Release to Manufacturing. Once we get to RC, the whole ecosystem is in “dress rehearsal” mode for the next steps.
- **Release to Manufacturing (RTM)** – This release is the final Windows 7 as we intend to make available to PC makers and for retail and volume license products.
- **General Availability (GA)** – This is a business milestone and represents when you can buy Windows 7 pre-installed on PCs or as full packaged product.

The obvious question is that we know the Pre-Beta was October 28, 2008, and the Beta was January 7th, so when is the Release Candidate and RTM? The answer is forthcoming. We are currently evaluating the feedback and telemetry and working to develop a robust schedule that gets us the right level of quality in a predictable manner. Believe me, we know many people want to know more specifics. We're on a good path and we're making progress. We are taking a quality-based approach to completing the product and won't be driven by imposed deadlines. We have internal metrics and milestones and our partners continue to get builds routinely so even when we reach RC, we are doing so together as partners. And it relies, rather significantly, on all of you testing the Beta and our partners who are helping us get to the finish line together.

Shipping Windows, as we hoped this shows, is really an industry-wide partnership. As we talked about in our first post, we're promising to deliver the best release of Windows we possibly can and that's our goal. Together, and with a little bit more patience, we'll achieve that goal.

We continue to be humbled by the response to Windows 7 and are heads down on delivering a product that continues to meet your needs and the needs of our whole industry.

--Steven on behalf of the Windows 7 team

# UAC Feedback and Follow-Up

Steven Sinofsky | [2009-02-05T18:00:00+00:00](#)

---

When we started the “E7” blog we were both excited and also a bit uneasy. The excitement is obvious. The unease is because at some point we knew we would mess up. We weren’t sure if we would mess up because we were blogging about a poorly designed feature or mess up because we were blogging poorly about a well-designed feature. To some it appears as though with the topic of UAC we’ve managed to do both. Our dialog is at that point where many do not feel listened to and also many feel various viewpoints are not well-informed. That’s not the dialog we set out to have and we’re going to do our best to improve.

This post is an attempt to get both the blog right and the feature right. We don’t like where we are in terms of how folks are feeling and we don’t feel good – Windows 7 is too much fun and folks are having too much fun for us to be having the dialog we’re having. We hope this post allows us to get back to having fun!

To start we’ll just show representative comments from the spectrum of feedback. We’ll then talk about the changes we’re making and also make sure we’re all on the same page regarding how we move forward. In terms of comments we’ve heard the following:

@sroussey says:

You have 95% of the people out there think you got it wrong, even if they are the ones that got it wrong. The problem is that they are the one's that buy and recommend your product. So do you give them a false sense of increased security by implementing the change (not unlike security by obscurity) and making them happy, or do you just fortify the real security boundaries?

And @Thack says:

Jon,

Thanks for sharing your thoughts. I understand your points.

Now, I want add my voice to the call for one very simple change:

Treat the UAC prompting level as a special case, such that ANY change to it, whether from the user or a program, generates a UAC prompt, regardless of the type of account the user has, and regardless of the current prompting level.

That is all we are asking. No other changes. Leave the default level as it is, and keep UAC as it is. We're just talking about the very specific case of CHANGES to the UAC prompting level.

It will NOT be a big nuisance - most people only ever change the UAC level once (if at all).

Despite your assurances, I REALLY WANT TO KNOW if anything tries to alter the UAC prompting level.

The fact that nobody has yet demonstrated how the putative malware can get into your machine is NO argument. Somebody WILL get past those other boundaries eventually.

Even if you aren't convinced by my argument, then the PR argument must be a no-brainer for Microsoft.

PLEASE, Jon, it's just a small change that will gain a LOT of user confidence and a LOT of good PR.

Thack

With this feedback and a lot more we are going to deliver **two changes to the Release Candidate** that we’ll all see. First, the **UAC control panel will run in a high integrity process**, which requires elevation. That was already in the works before this discussion and doing this prevents all the mechanics around SendKeys and the like from working. Second, **changing the level of the UAC will also prompt for confirmation**.

@mdaria510 says:

Sometimes, inconsistency with your own ideals is a good thing. Make an exception, if only to put people's fears to rest.

That sums up where we are heading. The first change was a bug fix and we actually have a couple of others similar to that—this is a beta still, even if many of us are running it full time. The second change is due directly to the feedback we’re seeing. This “inconsistency” in the model is exactly the path we’re taking. The way we’re going to think about this that the UAC setting is something like a password, and to change your password you need to enter your old password.

The feedback is that UAC is special, because it can be used to disable silently future warnings if that change is not elevated and so to change the UAC setting an elevation will be required. To the points in the comments, we also don’t want to create a sense or expectation of security that is not there—you should still not download code and run it unless you trust the source. HTML, EXE, VBS, BAT, CMD and more are all code and all have the potential to alter the environment (user settings, user files) running as a standard user or an administrator. We’re focused on helping

people make sure that code doesn't get on the machine without consent and many third party tools can help more as well. We want people to be comfortable with the new UAC control and the new default setting, so we'll make the changes outlined above as the feedback has been clear.

While we're discussing this we want to make sure we're all on the same page going forward in terms of how we will evaluate the security of Windows 7. Aside from the UAC setting, the discussion of the vulnerability aspects of the Windows 7 Beta have each started with getting code on the machine, which the mechanisms of Windows have prevented in the cases shown. We have also heard of security concerns that involve multiple steps to demonstrate a potential exploit. It is important to look at the first step—if the first step is "first get code running on the machine" then nothing after that is material, whether it is changing settings or anything else. We will treat very seriously the ability to get code on a machine and run without consent. As Jon's post highlighted briefly, the work in Windows 7 is about the increased protections in place to secure your PC from acquiring and running code without your consent, and of course we continue to make sure Windows code is secure from both tampering or circumventing the protections in the system.

We want to reiterate the security of the system overall. Windows 7 is SD3+C and is designed to be more secure than Vista—that's our priority. None of us want to have Windows 7 be perceived as being less secure than Vista in any way, because our design point is to make sure it is more secure than Windows Vista, by default.

We said we thought we were bound to make a mistake in the process of designing and blogging about Windows 7. We want to continue the dialog and hopefully everyone recognizes that engineering, perhaps especially engineering Windows 7, is sometimes going to be a lively discussion with a broad spectrum of viewpoints expressed. We don't want the discussion to stop being so lively or the viewpoints to stop being expressed, but we do want the chance to learn and to be honest about what we learned and hope for the same in return. This blog has almost been like building an extra product for us, and we're having a fantastic experience. Let's all get back to work and to the dialog about Engineering Windows 7. And of course most importantly, we will continue to hear all points of view and share our point of view and work together to deliver a Windows 7 product that we can all feel good about.

--Jon and Steven

# Update on UAC

Steven Sinofsky | [2009-02-05T03:00:00+00:00](#)

---

Hi, Jon DeVaan here to talk to you about the recent UAC feedback we've been receiving.

Most of our work finishing Windows 7 is focused on responding to feedback. The UAC feedback is interesting on a few dimensions of engineering decision making process. I thought that exploring those dimensions would make for an interesting e7 blog entry. This is our third discussion about UAC and for those interested in the evolution of the feature in Windows it is worth seeing the two previous posts ([post #1](#) and [post #2](#)) and also reading the comments from many of you.

We are flattered by the response to the Windows 7 beta so far and working hard at further refining the product based on feedback and telemetry as we work towards the Release Candidate. For all of us working on Windows it is humbling to know that our work affects so many people around the world. The recent feedback is showing us just how much passion people have for Windows! Again we are humbled and excited to be a part of an amazing community of people working to bring the value of computing to a billion people around the world. Thank you very much for all of the thoughts and comments you have contributed so far.

UAC is one of those features that has a broad spectrum of viewpoints with advocates staking out both "ends" of the spectrum as well as all points in between, and often doing so rather stridently. In this case we might represent the ends of the spectrum as "security" on one end and "usability" on the other. Of course, this is not in reality a bi-polar issue. There is a spectrum of perfectly viable design points in between. Security experts around the world have lived with this basic tension forever, and there have certainly been systems designed to be so secure that they are secure from the people who are supposed to benefit from them. A personal example I have, is that my bank recently changed the security regimen on its online banking site. It is so convoluted I am switching banks. Seriously!

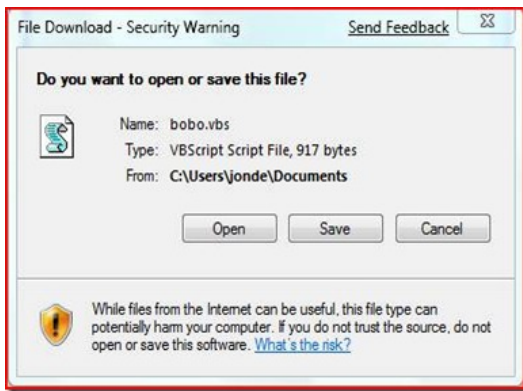
## Clarifying Misperceptions

As people have commented on our current UAC design (and people have commented on those comments) it is clear that there is conflation of a few things, and a set of misperceptions that need to be cleared up before we talk about the engineering decisions made on UAC. These engineering decisions have been made while we carry forth our [secure development lifecycle principles](#) pioneered in Windows XP SP2, and most importantly the principle of "secure by default" as part of SD3+C. Windows 7 upholds those principles and does so with a renewed focus on making sure everyone feels they are in control of their PC experience as we have talked about in many posts.

The first issue to untangle is about the difference between malware making it onto a PC and being run, versus what it can do once it is running. There has been no report of a way for malware to make it onto a PC without consent. All of the feedback so far concerns the behavior of UAC once malware has found its way onto the PC and is running. Microsoft's position that the reports about UAC do not constitute a vulnerability is because the reports have not shown a way for malware to get onto the machine in the first place without express consent. Some people have taken the, "it's not a vulnerability" position to mean we aren't taking the other parts of the issue seriously. Please know we take all of the feedback we receive seriously.

The word "vulnerability" has a very specific meaning in the security area. Microsoft has one of the leading security agencies in the world in the [Microsoft Security Response Center](#) ([secure@microsoft.com](mailto:secure@microsoft.com)) which monitors the greater ecosystem for security threats and manages the response to any threat or vulnerability related to Microsoft products. By any definition that is generally accepted across the world wide security community, the recent feedback does not represent a vulnerability since it does not allow the malicious software to reach the computer in the first place.

It is worth pointing out the defenses that exist in Windows Vista that keep malware from getting on the PC in the first place. In using Internet Explorer (other browsers have similar security steps as well) when attempting to browse to a .vbs file or .exe file, for example, the person will see the prompts below:

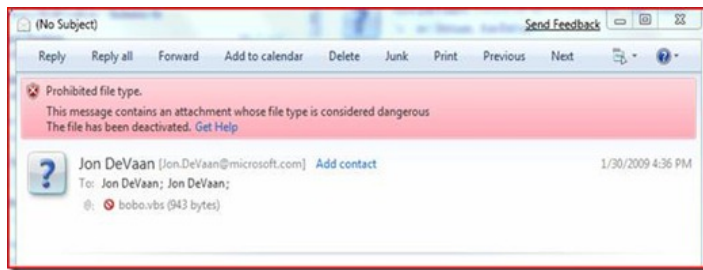


Internet Explorer 8 has also introduced many new features to thwart malware distribution (see <http://blogs.msdn.com/ie/archive/2008/08/29/trustworthy-browsing-with-ie8-summary.aspx>). One of my favorites is the [SmartScreen® Filter](#) which helps people understand when they are about to visit a malicious site. There are other features visible and hidden that make getting malware onto a PC much more difficult.



A SmartScreen® display from IE 8

Additionally, if one attempts to open an attachment in a modern email program (such as Windows Live Mail) the malware file is blocked:



Much of the recent feedback has failed to take into account the ways that Windows 7 is better than Windows Vista at preventing malware from reaching the PC in the first place. In Windows 7 we have continued to focus on improving the ability to stop malware before it is installed or running on a PC.

The second issue to untangle is about the difference in behavior between different UAC settings. In Windows 7, we have four settings for the UAC feature: “Never Notify,” “Notify me only when programs try to make changes to my computer (without desktop dimming),” “Notify me only when programs try to make changes to my computer (with desktop dimming),” and “Always Notify.” In Windows Vista there were only two choices, the equivalent of “Never Notify” and “Always Notify.” The Vista UI made it difficult for people to choose “Never Notify” and thus choosing between extremes in the implementation. Windows 7 offers you more choice and control over this feature, which is particularly interesting to many of you based on the feedback we have received.

The recent feedback on UAC is about the behavior of the “Notify me only when programs try to make changes to my computer” settings. The feedback has been clear it is not related to UAC set to “Always Notify.” So if anyone says something like, “UAC is broken,” it is easy to see they are mischaracterizing the feedback.

### **The Purpose of UAC**

We are listening to the feedback on how “Notify me only when...” works in Windows 7. It is important to bring in some additional context when explaining our design choice. We choose our default settings to serve a broad range of customers, based on the feedback we have received about improving UAC as a whole. We have learned from our customers participating in the Customer Experience Improvement Program, Windows Feedback Panel, user surveys, user in field testing, and in house usability testing that the benefit of the information provided by the UAC consent dialog decreases substantially as the number of notifications increases. So for the general population, we know we have to present only key information to avoid the reflex to “answer yes”.

One important thing to know is that UAC is not a security boundary. UAC helps people be more secure, but it is not a cure all. UAC helps most by being the prompt before software is installed. This part of UAC is in full force when the “Notify me only when...” setting is used. UAC also prompts for other system wide changes that require administrator privileges which, considered in the abstract, would seem to be an effective counter-measure to malware after it is running, but the practical experience is that its effect is limited. For example, clever malware will avoid operations that require elevation. There are other human behavior factors which were discussed in our earlier blog posts ([post #1](#) and [post #2](#)).

UAC also helps software developers improve their programs to run without requiring administrator privileges. The most effective way to secure a system against malware is to run with standard user privileges. As more software works well without administrator privileges, more people will run as standard user. We expect that anyone responsible for a set of Windows 7 machines (such as IT Administrators or the family helpdesk worker (like me!)) will administer them to use standard user accounts. The recent feedback has noted explicitly that running as standard user works well. Administrators also have Group Policy at their disposal to enforce the UAC setting to “Always Notify” if they choose to manage their machines with administrator accounts instead of standard user accounts.

Recapping the discussion so far, we know that the recent feedback does not represent a security vulnerability because malicious software would already need to be running on the system. We know that Windows 7 and IE8 together provide improved protection for users to prevent malware from making it onto their machines. We know that the feedback does not apply to the “Always Notify” setting of UAC; and we know that UAC is not 100% effective at stopping malware once it is running. One might ask, why does the “Notify me only when...” setting exist, and why is it the

default?

### **Customer-Driven Engineering**

The creation of the “Notify me only when...” setting and our choice of it as the default is a design choice along the spectrum inherent in security design as mentioned above. Before we started Windows 7 we certainly had a lot of feedback about how the Vista UAC feature displayed too many prompts. The new UAC setting is designed to be responsive to this feedback. A lot of the recent feedback has been of the form of, “I’ll set it to ‘Always Notify,’ but ‘regular people’ also need to be more secure.” I am sure security conscious people feel that way, and I am glad that Windows 7 has the setting that works great for their needs. But what do these so called “regular people” want? How to choose the default, while honoring our secure design principles, for these people is a very interesting question.

In making our choice for the default setting for the Windows 7 beta we monitored the behavior of two groups of regular people running the M3 build. Half were set to “Notify me only when...” and half to “Always Notify.” We analyzed the results and attitudes of these people to inform our choice. This study, along with our data from the Customer Experience Improvement Program, Windows Feedback Panel, user surveys, and in house usability testing, informed our choice for the beta, and informed the way we want to use telemetry from the beta to validate our final choice for the setting.

A key metric that came out of the study was the threshold of two prompts during a session. (A session is the time from power up to power down, or a day, whichever is shorter.) If people see more than two prompts in a session they feel that the prompts are irritating and interfering with their use of the computer. In comparing the two groups we found that the group with the “Always Notify” setting was nearly four times as likely to have sessions with more than two prompts (a 1 in 6.7 chance vs a 1 in 24 chance). We gathered the statistic for how many people in the sample had malware make it onto their machine (as measured by defender cleaning) and found there was no meaningful difference in malware infestation rates between the two groups. We will continue to collect data during the beta to see if these results hold true in a much broader study.

We are very happy with the positive feedback we have received about UAC from beta testers and individual users overall. This helps us validate our “regular people” focus in terms of the trade-offs we continue to consider in this design choice. We will continue to monitor the feedback and our telemetry data to continue to improve our design choices on UAC.

So as you can see there is a lot of depth to the discussion of UAC and the improvements made in Windows 7 in UAC itself and in improving ways to prevent malware from ever reaching a PC. We are working hard to be responsive to the feedback we received from Vista to provide the right usability and security for people of all types. We believe we’ve made good progress and are listening carefully to the feedback on our UAC changes. Again please accept our most sincere thanks for the passion and feedback on Windows 7. While we cannot implement features the way each and every one of you might wish, we are listening and making a sincere effort to properly weigh all points of view. Our goal is to create a useful, useable, and secure Windows for all types of people.

Jon



# Recognizing Improvements in Windows 7 Handwriting

Steven Sinofsky | [2009-02-09T03:00:00+00:00](#)

---

*Microsoft has been working on handwriting recognition for over 15 years going back to the Pen extensions for Windows 3.0. With the increased integration and broad availability of the handwriting components present in Windows Vista we continue to see increased use of handwriting with Windows PCs. We see many customers using handwriting across a wide variety of applications including schools, hospitals, banking, insurance, government, and more. It is exciting to see this natural form of interaction used in new scenarios. Of course one thing we need to continue to do is improve the quality of recognition as well as the availability of recognizers in more languages around the world. In this post, Yvonne, a Program Manager on our User Interface Platform team, provides a perspective on engineering new recognizers and recognition improvements in Windows 7. --Steven*

Hi, my name is Yvonne and I'm a Program Manager on the Tablet PC and Handwriting Recognition team. This post is about the work we've done to improve recognition in handwriting for Windows 7.

Microsoft has invested in pen based computing since the early 1990s and with the release of Windows Vista handwriting recognizers are available for 12 languages, including USA, UK, German, French, Spanish, Italian, Dutch, Brazilian Portuguese, and Chinese (Simplified and Traditional), Japanese and Korean. Customers frequently ask us when we plan to ship more languages and why a specific language is not yet supported. We are planning to ship new and improved languages for Windows 7, including Norwegian, Swedish, Finnish, Danish, Russian, and Polish, and the list continues to grow. Let's explore what it takes to develop new handwriting recognizers.

Windows has true cursive handwriting recognition, you don't need to learn to write in a special way – in fact, we've taught (or "trained" as we say) Windows the handwriting styles of thousands of people and Windows learns more about your style as you use it. Over the last 16 years we've developed powerful engines for recognizing handwriting, we continue to tune these to make them more accurate, faster and to add new capabilities, such as the ability to learn from you in Vista. Supporting a new language is much more than adding new dictionaries – each new language is a major investment. It starts with collecting native handwriting, next we analyze the data and go through iterations of training and tuning, and finally the system gets to you and continues to improve as you use it.

## Data Collection

The development of a new handwriting recognizer starts with a huge data collection effort. We collect millions of words and characters of written text from tens of thousands of writers from all around the world.

Before I describe our collection efforts, I would like to answer a question we are frequently asked: "Why can't you just use an existing recognizer with a new dictionary?" One reason is that some languages have special characters or accents. But the overriding reason is because people in different regions of the world learn to write in different ways, even between countries with the same language like the UK and US. Characters that may look visually very similar to you can actually be quite different to the computer. This is why we need to collect real world data that captures exactly how characters, punctuation marks and other shapes are written.

Setting up a data collection effort is challenging and time consuming because we want to ensure that we collect the "right kind of data". We carefully choose our collection labs in the respective countries for which we develop recognizers.

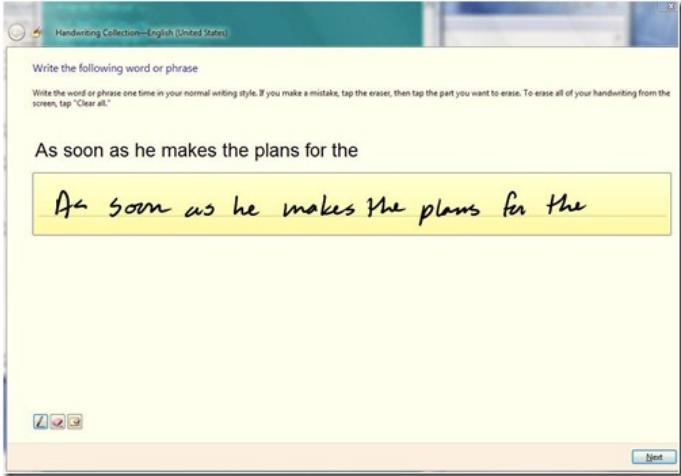
Before we start our data collection in the labs, we configure our collection tools, prepare documentation, and compile language scripts that will guide our volunteers through the collection process. Our scripts are carefully prepared by native speakers in the respective language to ensure that we collect only orthographically correct data, data from different writing styles, and data that covers all characters, numbers, symbols and signs that are relevant to a specific language. All of our scripts are proofread and edited before they are blessed to be used at the collection labs.

Once our tools and scripts are ready, we open our labs and start to recruit volunteers to donate their handwriting samples. Our recruitment efforts ensure that we have balanced demographics such as gender, age, left handedness, and educational background that represent the majority of the

population for that country.

A supervisor at the lab instructs the volunteers to copy the text as it is displayed in the collection tool in their own writing style. What is important to note is that we want to collect writing samples that accurately represent the person’s natural way of writing. We therefore encourage volunteers to treat “pen and tablet” like “pen and paper”. If one of the volunteers tends to writes in big, curvy strokes, then we want to collect his/her big, curvy strokes during the collection session. High quality data in this context refers to data that was naturally written.

Here is a snapshot of what our collection tool looks like:



**Figure 1: Collection Tool**

A collection session lasts between 60-90 minutes at which point our volunteer has donated a significant amount of handwritten data without feeling fatigued. The donated data is then uploaded and stored in our database at Microsoft ready for future use. The written samples contain important information like stroke orders, start- and end points, spacing, and other characteristics that are essential to train our new recognizer.

Let’s take a look at some of our samples in our database to illustrate the great variation among ink samples:

Prompt	Ink
black	black
black	black
black	black

**Figure 2: Ink samples illustrating different stroke orders.**

The screenshot shows how three different volunteers inked the word “black”. The different colors are used to illustrate the exact stroke orders in

which the word was written. Our first two volunteers used five strokes to write the word “black”; our third volunteer used four strokes. Please also note how our third volunteer used one stroke only to ink the letters “ck”, while our first volunteer used three strokes for the same combination of letters. All of this information is used to train our recognizers.

### **Neural Network and Language Model**

Once we have collected a sufficient amount of inked data, we split our data into a training set, used by our development team, and a “blind” set, used by our test team. The training set is then employed to train the Neural Network, which is largely responsible for the magic that is taking place during the recognition process. Good, naturally written data is essential in developing a high quality recognizer; the recognizer can’t be any better than its training set. The more high quality data we feed into our Neural Network, the more equipped we are to handle sloppy cursive handwriting.

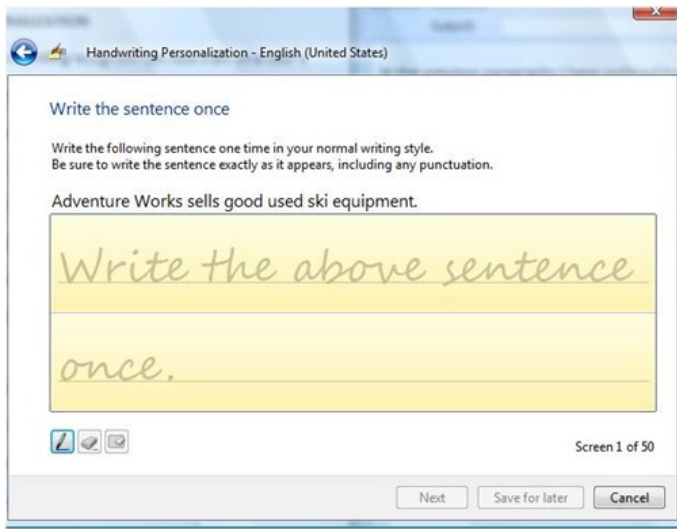
Our Neural Network is a Time-Delay Neural Network (TDNN) that can handle connected letters of cursive scripts. A TDNN takes ink segments of preceding and following stroke segments into consideration when computing the probabilities of letters, digits and characters for each segment of ink. The output of the TDNN is powerful but not good enough when handwriting is sloppy. In order to come within reach of human recognition accuracy, we have to employ information that goes beyond the shape of the letter: we call this the Language Model context. The majority of this Language Model context comes in form of the lexicon, which is a wordlist of valid spellings for a given language. For many languages, this is the same lexicon that the spellchecker uses. The TDNN and the lexicon work closely together to compute word probabilities and output the top suggestions for the given input.

Training the Neural Network is an involved process that takes time. We often experiment with borrowing data from other languages to increase the size of the training data with the ultimate goal to boost recognition accuracy. Borrowing characters from other languages does not always lead to success. As I mentioned above, stroke order, letter shape, writing styles and letter size can differ significantly from country to country and can have a negative impact on the performance of the TDNN. It often takes us several rounds of training, re-training and tuning before we find “the right formula” that will lead to high recognition accuracy.

How do we know if we are headed in the right direction when we build a new recognizer? This is an important question that the test team and native speakers answer for us. The test team is responsible for generating our recognition accuracy metrics that reflect how good our recognizer is. These accuracy metrics are based on our blind test set which is the collected data that development could not use for training. In addition to our accuracy metrics, we work with native speakers in house and at our world-wide subsidiaries to get feedback and further input.

### **Improving the recognizers through personalization**

In the previous paragraphs I have outlined how we develop high quality recognizers that can handle a wide variety of different writing styles. But there is more as each person can also train the recognizer his/her unique writing style. The training that is done to teach the recognizer a personal writing style is the same training that happens before Microsoft ships the product. The only difference is that we are now collecting unique training data from a specific person (and not that of thousands of people). We call this process “Personalization”.

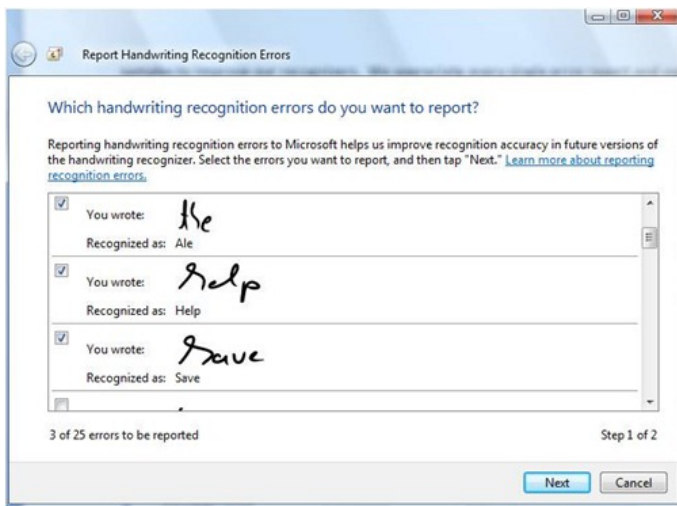


**Figure 3: Personalization Wizard (Sentence module).**

As the screenshots of our Personalization wizard illustrates, a person is asked to write the requested sentence to provide his/her ink samples. The more data a person donates during the personalization process, the better the recognizer will become. In addition to providing writing samples based on specified sentences, a person can target specific recognition errors, shapes, and characters that will all be used for training. Our Personalization feature is complex and offers a variety of different modules that enable a person to optimally tune the recognizer. We are proud to announce that Personalization will be available for all Vista languages and all new Windows 7 languages. We encourage you to use this feature to improve your recognition accuracy.

We continue to work on improving our recognizers which also means that we are incorporating our customers feedback through online telemetry (anonymously, privately, voluntary, and opt-in). In Windows Vista we released a new feature called "Report Handwriting Recognition Errors", which gives people the opportunity to submit those ink samples that the recognizer did not recognize correctly. After the person has corrected a word in the Tablet Input Panel (TIP), we enable a menu that allows a person to send the misrecognized ink together with its corrected version to our team

Here is a screenshot of what our error reporting tool looks like:



**Figure 4: With “Report Handwriting Recognition Errors” people can choose which of the misrecognized ink samples they want to submit.**

We receive approximately 2000 error reports per week. Each error report is stored in our database before we analyze it and use it to improve our next generation of recognizers. As you can imagine, real world data is extremely helpful because it is only this type of data that can reveal shortcomings of our recognizers.

We value and appreciate every single error report. Keep sending us your feedback, so that we can use it to improve the magic of our present and future recognizers.

Thank you,

– Yvonne representing the handwriting recognition efforts

# Advances in typography and text rendering in Windows 7

Steven Sinofsky | [2009-02-13T03:00:00+00:00](#)

---

*Even with the pictures and videos so commonplace on PCs, many of us spend most of our time looking at and interacting with text. Yet few of us stop to think about the depth of technology required to render text well and that this is an area that continues to benefit from improved technology in displays, graphics cards, as well as the APIs available to developers. In Windows 7, The support for text and fonts in GDI continues to provide the foundation for compatibility and application support. Building on the foundation of the modern DirectX graphics infrastructure, Windows 7 enhances the text output available to developers with DirectWrite. This is a new API subsystem and one that over time you will see adopted more broadly by applications from Microsoft, independent software developers, and within Windows itself. This post will also talk about improvements to ClearType and the Fonts, both available as part of the improvements to the GDI-based text APIs. This work was introduced at the PDC (pointers towards the end of the post). This post is by Worachai Chaoweeraprasit, a development lead on our Graphics feature team. --Steven*

One of the high-level goals of Windows 7 is to have even better graphics – graphics with higher fidelity. To that end, my team is looking into how to improve one of the most basic graphic elements in Windows, and that is text – the thing that’s always right in your face, but we hope you’ll never actually see it.

## The need for good text

About 80% of the time people spend with their PC is to either read or write. This should come as no surprise when you realize that text is essentially how the machine *talks* back to you, and until we have a technology that would allow it to interject thought directly into our brains, text would probably continue to be the way we receive information from the computer screen.

Studies have shown that good text leads to better productivity. Essentially we are wired as human to be incredibly good at capturing words and making a smooth, rapid transition between them – the basis of reading. We’re so good at it that we can do it unconsciously with incredible speed *given that the text is optimized for that process*. This might explain why many can *sink in* to a good book for hours, but some quickly become tired after staring at the computer screen for a while. Any visual-related factor that could disrupt the reading process effectively slows us down. Good text, therefore, is text that is tuned to support the human reading process with minimal distraction possible.

The evenness of the *white* surrounding each letter, word, line, and paragraph plays a huge role in keeping the pace of reading while the *black* elements holds our attention together. A line too long, a word too tight, a paragraph too uneven, any of these conditions take us farther and farther away from the message being delivered but closer and closer to the mere medium delivering it. The art of text is essentially to make the actual text itself disappears before your eyes, so that the ideas it delivers reappear in your head. The study of how to prepare proper text is known as *typography*. And, as a typographer would say: good typography is not to be seen; only the bad ones are. As a platform, the role of Windows is to deliver great presentation of text and offering software developers great tools for creating the best presentation possible in the context of the software they develop.

## Improving current techniques

People tend to develop habits and often over time these become the preferred way of getting things done. The more mundane the activity is, the easier we become attach to it, and the harder we’re willing to change. When it comes to text on your screen, the same screen you look at days in and days out. It could quickly become awkward if that completely changes overnight – even for the better. So, how do we go about improving on what we all become so used to? We want to make sure to support what is there and improve it, while supporting existing methods. But, before we get to understand the improvement, let’s first take a closer look into the current implementation really is and what challenges it presents over the years.

The current implementation is the product of text rendering design based on device pixel. The dimension of text at a certain size eventually translates into a fixed number of pixels in horizontal and vertical direction on the device surface. A 10-point text would translate to roughly 80 pixels height on a typical printer device of 600 dpi, while the same text would merely acquire 13 pixels on a 96 dpi monitor. This physical screen

condition was hardly adequate for the quality we're seeking for good text on screen.

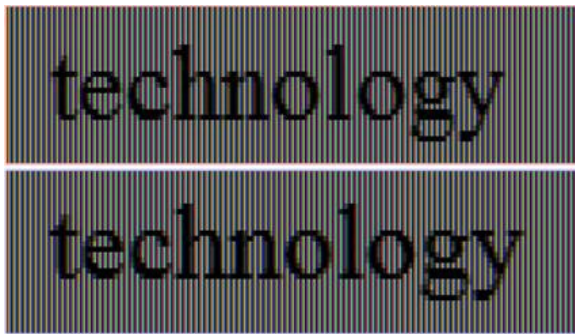
Fortunately, the advent of *ClearType* during the past decade has largely improved the clarity aspect of quality. ClearType leverages the anatomy of the LCD pixel structure and takes advantage of the human visual system to distribute the energy typically emit to a whole display pixel, across the neighboring *sub-pixels* in the LCD's typical 3-color channels making up each individual pixel, to create the visual illusion of higher resolution raster quality on a lower resolution device. As the result, ClearType text looks significantly sharper than the typical text on an LCD display, mitigating a large portion of the quality problem on a display technology that would become hugely popular a few years later.

Another pleasant design of the original ClearType in Windows was that it has improved the clarity of text without breaking application compatibility – that is, it doesn't change the actual size of each individual glyph in either direction, nor did it change the distance between the two adjacent ones. This is the reason one could turn it on or off at will without having to “store” the selected option in the document or application. It is entirely per-user rendering preference. In Windows 7 we also improved the ClearType Text Tuner in keeping with our theme of being in control of your PC experience, by providing even more granular choices when tuning ClearType (and of course you can still turn it off).

But like many other things in the world, the coin comes in two faces. While it is able to preserve backward compatibility, it is limited by its own leverage unable to advance the state of the art. The width and height of the individual glyph and the nominal distances between the adjacent two remain fixed to the rounded number of screen pixels at a given size.

One of the graphics improvements we made in Windows 7, therefore, is to move from the physical pixel model of the past, and instead creating a new design around what we call the “*device independent pixel*” unit (or “*DIP*”), a “virtual pixel” that is one-ninety-sixth an inch in floating-point data type. In this model, a glyph (or any other geometric primitive for that matter) can size to fractional pixels, and be positioned anywhere in between the two pixels. The new ClearType improvement allows sizing and placement of glyph to the screen's sub-pixel nearest to its ideal condition, creating a more natural looking word shape and making text on screen looks a lot closer to print quality.

The following figure shows the side-by-side comparison of the same word between the original or today's ClearType (above) and the Windows 7 improvement – *Natural ClearType* (below), which does require calling the new APIs to render. Notice the width of the letters in the word and the spacing between them, as well as how the more consistent width and spacing improves the overall appearance of the entire word. Note that all the letters are placed with its nominal spacing and there is no kerning adjustment being applied here. A great [article](#) by Kevin Larson – a researcher in the Advanced Reading Technology team, discusses in details the scientific aspect of word recognition.



The ability to be more precise in approximating the screen placement of natural text also lends itself to a very nice side-effect, and that is the fact that text can now be placed on the line with no regards to the actual display device's resolution. It means a UI designer can design an application UI knowing it'll look the same on all other screens as it appears on his or her screen regardless of what type of display device the users might have. This fact is also particularly handy for software localization where the translated text produces the same layout everywhere.

This improvement could also offer a more realistic view of a print document on screen, or make the screen document looks closer to its print counterpart. It could also improve the quality of document zooming. Imagine document zoom that could go in and out in the same manner as what you would see when pulling the actual print page closer and farther away from your sight. It could mean a more joyful experience for online reading.

## Fonts and Font Management

The Font is the heart and soul to typography, much like photo is to photography. A lot more fonts are shipped with Windows these days while even more are developed around the world. Windows Vista shipped with 40% more fonts comparing to Windows XP. Windows 7 is expected to ship with 40+ new fonts, just to underscore this trend. We've also added some additional viewing/categorization capabilities using the Windows 7 Explorer to improve working with a large library of related fonts.

The default common controls' font dialog and the font chunk in Windows 7 Ribbon are also updated to be more intelligently selective of what fonts to be present to the user of the current user's profile. Depending on a number of settings including the current UI language, the user locale, and the current set of keyboard input locales, the font list hides fonts of languages not typically used by the user of different culture and locale. For example, all the international fonts are automatically hidden away from a typical English user to reduce clutter and promote better productivity in common system applications such as NotePad, WordPad and Paint. Third-party application utilizing the Ribbon or the common controls' font dialog could also have the same benefit. The user still retains the option of selecting any desired font back to the view by explicitly marking it in the Windows 7 Control Panel's Font applet.

Operating System	Fonts shipped "in-box"
Windows XP SP2	133
Windows Vista	191
Windows 7	235 (currently planned)

This growth, however, introduces some new opportunities for improvement. We've long treated fonts as system-wide resources. It gets "installed" on the machine and kept in a single flat namespace managed by the core part of the operating system. It may be interesting to some that the font named "Arial Black" isn't really in the same grouping as "Arial Narrow" or "Arial". This is because as far as the operating system is concerned, they are just different fonts with different names. And because font is uniquely identified by its name, you can't have multiple versions of the same font at the same time.

Because font is system resource, non-traditional usage of font such as font embedded within the document, and font used exclusively in an application is done through the mechanism known as *private* installation, which involves making sure the font name is unique before installing it programmatically but doing so by hiding it from others to see. Private font is just like font installed publicly as far as the operating system internal is concerned.

An important improvement in Windows 7's new font system is the notion of "font collection" which allows partitioning of fonts sharing the same usage into a separate namespace. The *system* collection is similar to what exists today and is created and managed by the system whereas *custom* collection can be created and managed, as many as needed, entirely by the application program. This allows document to have its own set of fonts local to it, and third-party application or plug-in to ship with its own font used exclusively within the program. This partitioning not only reduces unnecessary system-wide font update and allows update to happen only locally as needed, it also allows access to multiple versions of the same font in different collections.

The new font system also improves the way fonts are organized within the collection. It supports the notion of *weight-width-slope* variation where



fonts with the same stylistic root but vary in its weight (thin, light, bold, black, etc.), width (wide, narrow, etc.), or slope (italic, oblique) are grouped together in the same font family. For instance, “Arial Narrow” becomes a variation or *face* in the “Arial” family. This grouping model is advocated by the CSS recommendation.

## Font Art

Fonts also represent art and artistic expression. The technology helping create font is therefore the artist’s tool of expression. An important technology called *OpenType* emerged during the past decade. It enables new ways type design can be realized. OpenType allows designer to define how glyphs interact and transform in stages. The designer then exposes this function as an executable unit known as the “font feature” for application programmable access.

OpenType was an offshoot of the *TrueType Open* technology Microsoft developed in 1994-95. The TrueType Open technology added the *GSUB*, *GPOS*, *BASE*, *JSTF*, and *GDEF* tables to the *TrueType* format. The primary usage at the time was to help with the creation of Arabic font due to the inherent complexity of the task. Microsoft chose to rename the technology to *OpenType* in 1996 and Adobe added their CFF glyph outline format to the technology in the same year. Today OpenType is used to improve readability of text as well as to express new and exciting type design in various languages.

However, despite its long-time presence and availability, the usage of OpenType in the Windows world remains largely in specialized programs. The Windows native graphics system has not fully embraced OpenType for its mainstream usage of text. This absence discourages many designers as there is no standard way in Windows to test the feature they produce. Likewise, its limited exposure doesn’t encourage discoverability for mainstream application developers. Improving this and transitioning to this improved rendering technology is a multi-step and multi-release investment done so as to maximize the benefit while minimizing the disruption that might be introduced as incompatibilities. Windows 7 takes another step on this path. We know for many that care deeply about this area there is a strong desire to move faster. We are doing our best to balance the speed of transition with the desire to maintain compatibility.

Windows 7 new text system not only uses available OpenType features internally but also allows access to any feature made available in the font in the high level programming interface, making it easier for application developer to discover and exercise the font feature in mainstream scenario. Windows 7 also ships with a brand new OpenType font “*Gabriola*” developed by a well-respected typographer [John Hudson](#). *Gabriola* makes heavy use of contextual letterforms and offers an unprecedented number of stylistic sets for different usages of the font in different occasions. The figure below enumerates all stylistic sets available in this font; notice the subtleties and not-so-subtle way to distinguish each stylistic set.



The figure below also demonstrates the power of contextual letterforms in the eighth rendition of *Gabriola*’s stylistic set (“*ss07*”) to produce different ways the same word is rendered depending on where it’s at in the line.

Valentines Valentines Valentines  
Valentines

## New APIs

Rendering text is complex and involved, even though it seems like something that should be straight forward. There are probably hundreds of ways to format text in a document and often many paths that ultimately yield the same results. HTML/CSS is a complex standard and is a great example of the richness of how text may be formatted and typeset. Underneath the formatting logic lies the language requirement – the rule of writing for the language. Windows has long been supporting Unicode – another complex standard for global data interchange. Windows supports an increasing number of Unicode script in every single release. The mapping from the input text to the final glyphs in the font requires intricate transformation, which involves parsing of font data and analyzing the language writing pattern. Once the glyph is finalized, it is then rasterized, merged and filtered into the final visual on the display device.

Due to this staging nature, different types of applications require different support from the text system. While a typical application such as the legendary “Hello world” type application may be satisfied with only the ability to get some text out showing to the user. The same level of support is hardly adequate for document preparation system such as Microsoft Word and Adobe InDesign. Some of the more mature application code bases may also have to deal with different graphics systems. This makes it harder in practice for a text system that tie to a particular graphics model to really be widely useful across the wide variety of applications in the Windows ecosystem.

It became obvious to us early on during the planning stage of Windows 7 that text processing is not homogeneous, and different types of applications have different needs and requires different levels of support. The appropriate level of programming access to the text functionality is as important as the functionality itself. The new text system in Windows 7 is assembled into a self-sufficient system called [DirectWrite](#). The API is provided in four layers – the interfaces for font data, rendering support, language processing, and typesetting, each built upon the others with the lower layer makes no requirement to the upper one, and none depends on a specific graphics model. To illustrate the latter point, the figure below shows a sample application that uses the new typesetting interface and language processor while the final rendering happens as an extruded filled 3D geometry from the 2D graphics environment also new to Windows 7 called [Direct2D](#). Both systems were introduced in [PDC 2008](#) as the new graphic foundation in Windows 7.



DirectWrite preserves developer’s investment in existing technologies such as GDI and GDI+ in three important aspects. First, the previously described layering design of DirectWrite allows for the clean separation between the two fundamental processes of placing and rendering of text. It enables applications to use DirectWrite to place text while having it rendered onto traditional graphic surfaces such as GDI and GDI+. The reverse scenario in which the application may use GDI to place text while having it rendered through DirectWrite is also naturally supported. The

second aspect of compatibility comes from the fact that DirectWrite also supports all existing methods for placing and rendering text found in GDI. A DirectWrite application can use DirectWrite to place and render text in the same manner as GDI does without actually using GDI. Text placed and rendered under this compatibility mode is indistinguishable from GDI text from the user's point of view, and as such preserving existing layout of application UI and text document. Lastly, DirectWrite exposes a set of APIs that interoperate with GDI. An application selecting a GDI font object can turn it into a DirectWrite's font object and vice versa. Since the font system is at the low end of the DirectWrite API layer, it provides a natural interoperability point that is fundamental enough to ensure high degree of data preservation and correctness. Once the application is able to acquire a DirectWrite's font object, it can in turn use it in any other DirectWrite API requiring a DirectWrite font from that point onward. The conversion from a DirectWrite's font object back to a GDI font object allows the rest of the GDI-based application to function with no change while still being able to reap the benefit of using DirectWrite's new and improved font model. As in some real world examples, the XPS print rasterizer in Windows 7 is implemented on top of DirectWrite and utilizes DirectWrite's interoperability API to convert back to a GDI font as part of the conversion of an XPS-based print job for a non-XPS printer driver. The Windows 7 XPS Viewer also uses DirectWrite alongside the GDI+ graphic rendering for its onscreen display.

There's a lot more to the details of the API. In the PDC session linked to above, Leonardo Blanco and Kam VedBrat go into the details of DirectWrite and Direct2D and how to develop applications such as this.

The world has changed a lot since the first text APIs of Windows GDI, such as [TextOut](#) or [ExtTextOut](#) in Windows NT 3.1 (or the subsequent [API additions](#)). The evolution of support for text is a critical part of the underpinnings of Windows 7. We continue to improve this most "basic" element of a graphical operating system so that regardless of the language, script, or device used to render text, Windows will offers a great set of tools and APIs for developers and a great experience for end-users.

--Worachai

# Engineering the Windows 7 Boot Animation

Steven Sinofsky | [2009-02-18T03:00:00+00:00](#)

*As we connect through this blog and through all of those talking about Windows 7 it is clear that folks have a lot of passion around many topics. We learned early on about the passion around the boot/startup sequence and how important it was for that to go by quickly! At the same time, we know that it is really dull to watch a HDD light blink when resuming a machine from hibernate or powering up a machine. To improve this first connection with people, we set out to improve the boot sequence—jazz it up if you will. It sounds pretty easy, but as we looked into solving this we found it is a pretty significant engineering challenge. Our goal was to have fun while having no impact on the boot performance of the system. To explain this engineering and describe the boot sequence, Karen Wong, a program manager on our Core User Experience feature team, authored this post. --Steven*

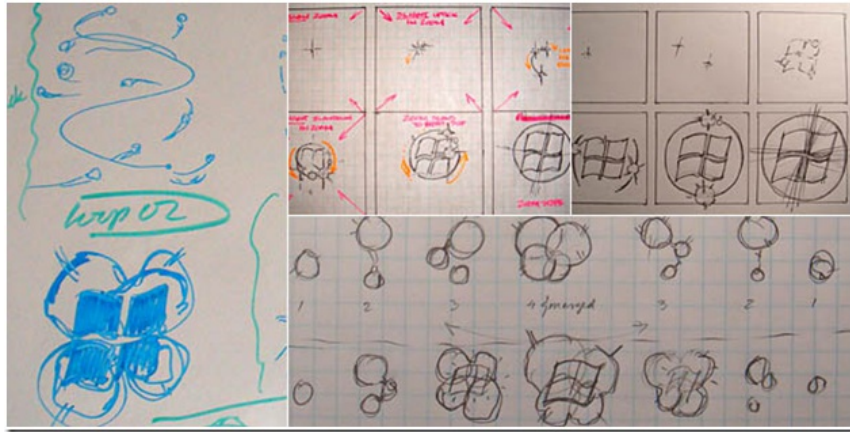
## Design

We use the word “personality” to refer to some of the characteristics of software that connect emotionally with people. ‘Light’ and ‘energy’ are some of the terms we use to describe the personality of Windows 7. As we designed Windows 7 it became clear that in order to showcase these elements of Win7’s personality, we needed go above and beyond what we did with Vista’s boot visuals.



From a design perspective, we know that the visual presentation of a feature plays a key role in the user’s perception of performance and quality. Our objective was to make Windows boot beautiful and was inspired by our Windows 7 personality of light and energy; and the way these forms reveal themselves in nature became our design palette. Words such as “bioluminescence”, “organic”, “humble beauty”, and “atmosphere” came up frequently in our brainstorming sessions. We know that in isolation these might sound a bit *corny*, but this is all part of the overall goals of Windows 7.

Over two dozen boot sequence designs were created, reviewed, and user tested to evaluate them against our goals. Designs varied in the saturation and/or brightness of color, the complexity of motion, and lighting effects. Here are some sketches from our design journey:



The final design in Windows 7 shows energy approaching from four directions, that join to form a light that projects through a window (of course it is no coincidence that the Windows logo resembles a window!). A subtle pulse indicates progress thereafter; another design detail that reinforces the liveliness of Windows 7's personality.



From a design perspective, this new boot sequence met all of our design goals, and we were excited to send it out into the world. However, boot had to be more than just a pretty face. From an engineering perspective, we had some clear challenges to overcome, as we knew that “time to desktop” was still the most important thing to users. Visual delight could not trump getting to the desktop faster and many of you have been critical of features that are dubbed “eye candy” – the boot sequence is not going to be one of those features for sure.

### **No compromise on performance**

If we had kept everything the same from Vista and simply updated the boot animation to the new Win7 look, we would not have achieved new levels performance and quality that we aspire to. In fact, significant code changes were required in order to make the new boot animation even possible in Win7.

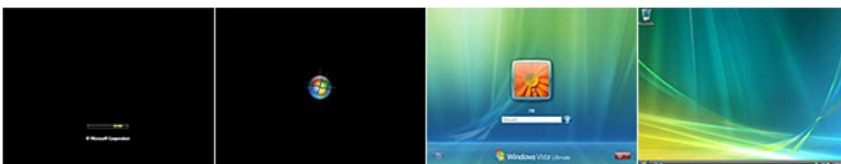
In Vista, the boot loader is using a low resolution 640x480 screen, and file size required for the green animated progress bar is very small. Furthermore, the Vista boot screen had low color depth – 16 bits per pixel (bpp). We increased this in Win7 to 32 bpp, which enabled the color richness you see in the new boot animation. Updates to the Vista boot progress indicator were achieved via the CPU, which was susceptible to I/O time, therefore causing occasional glitches in the animation. With the low resolution screen, limited color depth, and susceptibility to glitches – we knew we had our work cut out for us if we wanted to build something fancier for Win7.

We started with the Win7 boot loader using a different mechanism to display the boot animation. It gets a pointer to the frame buffer from the firmware (either BIOS or UEFI firmware), and displays a higher resolution image (1024 x 768). It animates the image while the kernel and boot critical device drivers are loaded into memory. Since the native graphics driver for the display is not loaded into memory and initialized yet, the animation is run by using the CPU, and by updating the frame buffer for the graphics display. We made an additional optimization - to have the CPU use write-combined caching to accelerate performance.

Michael Fortin's [blog entry](#) on boot performance describes how the early stage of boot is I/O bound, as it is loading the kernel, device driver files, and other system component files. We therefore limited the dimensions of the boot animation to a small region of the screen, to avoid introducing any delay during the early stage of boot. A larger animation area would require loading a larger set of animation images, which adds to the file I/O. The animation images are compressed by incorporating the bitmaps as resources, which are then compressed using WIM image compression. WIM image compression reduces the overall file size, thereby reducing the I/O required to read them in. It also reduces the on-disk footprint. Animating a smaller region of the screen, and using a slightly lower frame rate also keeps the CPU overhead of updating the frame buffer to a low enough level, that there is no added overhead to the boot time.

Another change we made that improved not only the performance of boot, but the quality, was the reduction of transitions in graphics mode. These transitions occur during initialization of the graphics subsystem and Windows shell. In Vista, these cause the boot experience to be less smooth, as the display changes (flashes black) a few times before presenting the user with a logon screen (or the user's desktop if there is only one system user).

After looking deep into our boot architecture for performance and quality improvements to enable the new animation, we were pleasantly surprised that the act of beautifying the boot animation created a new opportunity to further decrease time to desktop. In Vista, when a customer powered on the machine, the boot sequence included an animation of the Windows flag, or 'pearl', before reaching the login screen (or the desktop if the user is set to auto-login). Due to the Vista boot architecture constraints, this pearl animation can only play after boot code has already completed.



Vista Boot Sequence, with Pearl Animation

Now that new boot visuals display a rich animation that reflects the Windows 7 personality, the pearl animation seemed out-of-date and redundant, and was removed. As a result, we saved the time it takes to play this animation after boot is complete.



Windows 7 Boot Sequence, Pearl Animation Removed

You may also be wondering what happened to the startup sound. In Vista, the sound had to be synchronized with the pearl animation to produce the highest quality experience. This has potential performance impact on some hardware, as we require the system's sound stack to be loaded to

complete the pearl sequence. In the cases when we are waiting for the system's sound playback to be ready, a delay can occur in getting to the desktop. As such, we changed the sound to now play asynchronously, anytime after the logon screen loads. On most hardware that we tested, this is right when the logon screen displays. We heard customer feedback in Vista that the sound played and caught your attention, but boot was not yet complete. So in addition to performance benefits, this change also improves the user experience by letting users know when their machine is ready for use.

The sum of the boot code optimizations and removal of the pearl animation from Vista enabled us to add a rich, high-quality animation during boot, with no increase in the time it takes a user to reach the desktop.

## **Designing for a wide range of hardware**

The boot experience varies depending on the user's hardware. We made some design decisions to ensure the best *visual* experience across a wide range of hardware, however the time it takes a system to get to the desktop is mainly hardware-dependent.

For example, you may notice that there is a delay before the animation starts during boot, and this delay time varies depending on system hardware. To optimize for showing immediate feedback, we actually display text on the boot screen before Windows has had a chance to start all the processors on the system. It is only when that is complete that the animation can run asynchronously to the rest of the I/O during boot (which as discussed earlier is necessary for optimal performance and quality).

You may also notice that the Windows flag's dimensions during boot may change slightly on different screen sizes. Due to technical constraints in Win7, boot is always displayed in our recommended minimum resolution – 1024x768, regardless of the system's native resolution. Today, most hardware is set to stretch the boot sequence to fill the screen, as opposed to centering it. Consequently, the boot animation is usually stretched on screens that are of different aspect ratio than 1024x768; however, we did test the sequence on common aspect ratios to ensure that visual quality was preserved.

## **Boot, Reboot and Resume from Hibernate**

With all this hard work to improve the boot experience, we couldn't let it go to waste. As such, users will also have this experience when they resume from hibernate.

## **Personalization**

We know many of you might be asking if you could include your own animation or customize this sequence. This is not something we will support in Windows 7. We've talked about and shown a great many "personalization" elements of Windows 7 already, such as the new themepacks which you can try out in the beta. The reasons for this should be pretty clear, which is that we cannot guarantee the security of the system to allow for arbitrary elements to be loaded into memory at boot time. In the early stages of starting Windows, the system needs to be locked down and execute along a very carefully monitored and known state as tools such as firewalls and anti-virus checking are not yet available to secure the system. And of course, even though we're sure everyone would follow the requirements around image size, content, etc. due to performance we would not want to build in all the code necessary to guarantee that all third parties would be doing so. One of our design goals of Windows 7 was around making sure there are ample opportunities to express yourself and to make sure your PC is really your PC and so we hope that you'll understand why this element is one we need to maintain consistently.

This was a quick behind the scenes look at something that we hope you enjoy. With Windows 7 we set out to make the experience of starting a Windows PC a little more enjoyable, and from the feedback we've seen here and in other forums, we think we're heading in the right direction. In

addition to our efforts to make boot fast, we also have a goal to make the system robust enough, such that most of you will not see this new boot animation that often and when you do it will be both enjoyable and fast!

--Karen



# Feedback and Engineering Windows 7

Steven Sinofsky | [2009-02-25T03:00:00+00:00](#)

---

Just about every email we receive and every comment we get comes with feedback—something to change, something to do more of, something to do less of, and so on. As we've talked about in this blog, acting on each one in an affirmative manner is easier said than done. What we can say for certain, is that we are listening to each and every comment, blog post, news story, MS Connect report, Send Feedback item, and of course all the data and telemetry. This post kicks off the discussion of changes made to the product with an overview of the feedback process. We'll get into specific changes shortly and we'll continue to return to the theme of changes in the Release Candidate (RC) over the next weeks. Yesterday on the [IE Blog](#), you saw that we'll be updating IE 8 on Windows 7, and there we also talked about the feedback process in general.

Feedback about Windows 7 of course starts before we've written any code, and by the time we've got running code thousands of people outside of Microsoft have provided input and influenced the feature set and design of Windows 7. As we've seen, the input from even a small set of customers can often represent a wide variety of choices--often in alignment, but just as often in opposition. As we're developing the features for Windows 7 we work closely with PC makers, enterprise customers, and all types of customers across small business, education, enthusiasts, product reviewers and industry "thought leaders", and so on. We shape the overall "blueprint" of the release based on this wide variety of input. As we have design prototypes or code running, we have much more targeted and specific feedback by using tools such as usability tests, concept tests, benchmark studies, and other techniques to validate the implementation of this blueprint. Our goal with this level of feedback is for it to be *representative* of the broad set of Windows customers, even if we don't have a 1:1 interaction with each and every customer. Hopefully this post will offer some insights into this process overall--the tools and techniques, and the scope of feedback.

In the first few weeks of the Windows 7 beta we had over one million people install and use Windows 7. That's an astounding number for any beta test and while we know it has been fun for many folks, it has been a lot of work for us--but work that helps to raise the quality of Windows 7. When you use the beta you are automatically enrolled in our Customer Experience Improvement Program (anonymous feedback and telemetry, which is voluntary and opt-in in the RTM release). Just by using Windows 7 as a beta tester you are helping to improve the product--you are providing feedback that we are acting on in a systematic manner. Here is a sense of the scale of feedback we are talking about:

- During a peak week in January we were receiving one Send Feedback report *every 15 seconds* for an entire week, and to date we've received well over 500,000 of these reports. That averages to over 500 reports for each and every developer to look through! And we're only through 6 weeks of using the Windows 7 beta, even though for many Windows 7 already seems like an old friend.
- To date, with the wide usage of the Windows 7 Beta we have received a hundreds of Connect (the MSDN/Technet enrolled beta customers) bug reports and *have fixes in the pipeline for the highest percentage of those reported bugs than in any previous Windows development cycle*.
- To date, we have fixes in the pipeline for nearly 2,000 bugs in Windows code (not in third party drivers or applications) that caused crashes or hangs. While many Beta customers have said they are very happy with the quality of Windows 7, we are working to make it even better by making sure we are fixing the issues experienced by such broad and significant usage.
- To date, we have recorded over 10,000,000 device installations and over 75% of these were able to use drivers provided in box (that is no download necessary). The remaining devices were almost all served by downloading drivers from Windows Update and by direct links to the manufacturer's web site. We've recorded the usage of over 2.8M unique plug-and-play device identifiers.
- On a personal note, I've received and answered almost 2,000 email messages from folks all around the world, just since this blog started in August. I really appreciate the discussion we're having and am doing my best to keep up with all the mail.

We have a variety of tools we draw on to help inform the decision making process. A key element that we have focused on quite a bit in Windows 7 is the role of data in making decisions. Everything we do is a judgment call as ultimately product development is about deciding what to get done from an infinite set of possibilities, but the role of data is essential and is something that has become far more routine and critical. It is important to be super clear—data is not a substitute for good judgment or an excuse to make a decision one way or another, but it most definitely informs the decision. This is especially true in an era where the data is not only a survey or focus group, but often includes a “sampling” of millions of people using Windows over the course of an extended time period.

A quick story from years ago working on Office, many years ago before the development of telemetry and the internet deciding what features to put in a release of Office could really be best described as a *battle*. The battle took place in conference rooms where people would basically

debate until one or more parties gave up from fatigue (mental or otherwise)—essentially *adrenaline-based product development*. The last person standing, the one with the most endurance, or the one who pulled an all-nighter to write the code pretty much determined how features ended up or what features ended up in a product. Sort of like turning feature design over to a *Survivor*-like process. I'm sure many of you are familiar with this sort of process. The challenges with this approach are numerous, but inevitably features do not hold together well (in terms of scenarios or architecture), the product lacks coherency, and most importantly unless you happen to have a good match between the “winner” and the target customers, features will often miss the mark.

In the early 1990's we started instrumenting Word and learning about how people actually used the software (this was before the internet so this was a special version of the product we solicited volunteers to run and then we would collect the data via *lots* of floppies). We would compile data and learn about which features people used and how much people used them. We learned things such as how much more people used tables than we thought, but for things very different than tables. We learned that a very significant amount of time the first suggestion in the spelling dictionary was the right correction (hence autocorrect). We learned that no one ever read the tip of the day (“Don't run with scissors”). This data enabled us to make real decisions about what to fix, the impact of changes, and then when looked at the goals (the resulting documents) what direction to take word processing.

Fast forward to the development of Windows 7 and we're focused on using data to help inform decisions we make. This data takes many forms and helps in many ways. I know a lot of folks have questions about the data – is it representative, how does it help fix things people should be using but don't, what about doing new things, and so on. Data is an important element of making decisions, but not a substitute for clear product goals, meaningful customer engagement, and working across the ecosystem to bring Windows 7 to customers.

Let's talk a bit about “bugs”. Up front it is worth making sure we're on the same page when we use the much overloaded term bug. For us a bug is *any time the software does something that someone one wasn't expecting it to do*. A bug can be a cosmetic issue, a consistency issue, a crash, a hang, a failure to succeed, a confusing user experience, a compatibility issue, a missing feature, or any one of dozens of different ways that the software can behave in a way that isn't expected. A bug for us is not an emotional term, but just shorthand for an entry in our database representing feedback on the product. Bugs can be reported by a human or by the various forms of telemetry built into Windows 7. This broad definition allows us to track and catalog everything experienced in the product and do so in a uniform manner.

Briefly, it is worth considering a few types of data that help to inform decisions as some examples.

- **Customer Experience Improvement Program.** The CEIP covers the full set of data collected on your PC that is provided to Microsoft in an anonymous, private, and opt-in manner. During the beta, as we state, this is defaulted on. In the retail product of course this is optional. During the course of the beta we are seeing the data about usage of new features, where people are customizing the product, what commands are being used, and in general how is Windows 7 being used. You've seen us talk about some of this data from Windows Vista that informed the features of Windows 7, such as the display resolution being used or the number of accounts on a machine. There are many data points measured across the product. In fact, an important part of the development cycle is to make sure that new features are well instrumented to inform us of usage during beta and down the road.
- **Telemetry.** While related to CEIP in the programmatic sense, we look at telemetry in a slightly different manner and you've seen this at work in how we talk about system performance or about the diversity of devices such as our discussion of high DPI support. Throughout the course of the beta we are able to see how boot time evolves or which devices are successfully installed or not. Important elements of telemetry that inform which bugs we fix are how frequently we are seeing a crash or a hang. We can identify software causing a higher level of issues and the right team or ISV can know to work on the issue. The telemetry really helps us focus on the benefit of the change—fixing a bug that represents thousands of customers, a widely used device, or broadly used third party software has a much bigger impact than a bug that only a few people, lower volume device, or less used software product might address. With this data we can more precisely evaluate benefit of changes.
- **Scenario based tests.** During the course of developing a feature we can take our designs and prototypes (code, paper, or bitmaps) and create a structured study of how customers would interpret and value a feature/scenario. For example, early in the planning of Windows 7 we created a full working prototype of the taskbar enhancements. With this prototype we can study different types of customers (skill levels, familiarity with different versions of Windows, competitive product customers, IT pro or end-user) and how they react to well-defined series of “tasks”. This allows a much more detailed study of the feature, as one example. As with all tests, these are not a substitute for good judgment in broader context but a key element to inform decisions.
- **Benchmarking studies.** As we transitioned to the pre-beta we started to have real code across the whole product so we began validation of Windows 7 with real code in real world scenarios. We call these studies benchmarking because often we are benchmarking the new product against a baseline of the previous version(s) of Windows. We might do a study where we see how long it takes to share a printer in

the home and then compare that time to complete/success rate with a Windows 7 test using HomeGroup. We might compare setting up a wireless network with and without WPA. We have many of these types of benchmarks and work to make sure that we understand both the progress we've made and where we might need to improve documentation, tutorials, or other forms of assistance.

This type of feedback all represents *structured feedback* in that the data is collected based on a systematic study and usually has a hypothesis associated with it. We also have the *unstructured* feedback which represents the vast array of bug reports, comments, questions, and points of view expressed in blogs, newsgroups, and the Send Feedback button—these are unstructured because these are not collected in a systematic manner, but aggressively collected by any and all means. A special form of this input is the bug reporting done through the Connect program—the technical beta—which represents bug reports, feature suggestions, and comments from this set of participants.

The Windows 7 beta represents a new level of feedback in this regard in terms of the overall volume as we talked about above. If you go back and consider the size of the development team and the time it would take to just read the reports you can imagine just digesting (categorizing, understanding, flagging) issues let alone responding to them is a massive undertaking (about 40 Send Feedback reports per developer during that one week, though as you can imagine they are not evenly distributed across teams).

The challenge of how to incorporate all the feedback at this stage in the cycle is significant. It is emotional for us at Microsoft and the source of both considerable pride and also some consternation. We often say “no matter what happens, someone always said it would.” By that we mean, on any given issue you can be assured that all sides will be represented by passionate and informed views of how to resolve it, often in direct opposition to each other plus every view in the middle. That means for the vast majority of issues there is no right or wrong in an absolute sense, only a good decision within the context of a given situation. We see this quite a bit in the debates about how features should work—multiple solutions proposed and debate takes place in comments on a blog (people even do whole blogs about how things should work). But ultimately on the Windows development team we have to make a call as we're seeing a lot of people are looking forward to us finishing Windows 7, which means we need to stop changing the product and ship it. We might not always make the right call and we'll admit if we don't make the right call, even if we find changing the behavior is not possible.

Making these decisions is the job of program management (PM). PMs don't lock themselves in their offices and issue opinions, but more realistically they gather all the facts, data, points of view, and work to synthesize the best approach for a given situation. Program management's role is making sure all the voices are heard, including beta testers, development, testing, sales, marketing, design, customer support, other teams, ISVs, IHVs, and on and on. Their role is to synthesize and represent these points of view systematically.

There are many factors that go into understanding a given choice:

- **What is it supposed to do?** At the highest level, the first question to ask is about how is something supposed to work. Sometimes things are totally broken. We see this with many many beta issues around crashes and hangs for example. But there's not a lot of debate over these since if it crashes in any meaningful frequency (based on telemetry) it should be fixed. We know if it crashes for you then it is a “must fix” but we are looking across the whole base of customers and understanding the frequency of a crash and also whether the code is in Windows, a driver from a hardware maker, or software from a third party—each of those has a different potential resolution path to consider. When it comes to user interaction there's two elements of “supposed to do”. First, there's the overall scenario goal and then there's the feedback of how different people with different experiences (opinions) of what it should do. As an example, when we talked about HomeGroup and the password/passphrase there was a bunch of feedback over how this should work (an area we will be tweaking based in part on this feedback). We of course have specifications and prototypes, but we also have a fluidity to our development process such that we do not have 100% fidelity before we have the product working (akin to architectural blueprints that leave tons of decisions to be made by the general contractor or decided while construction is taking place). There are also always areas in the beta where the feature is complete but we are already on a path to “polish” the experience.
- **How big is the benefit?** So say we decide something is supposed to behave differently. Will it be twice as good? Will it be 5% better? Will anyone notice? This is always a great discussion point. Of course people who advocate for a change always are convinced that the change will prevent the feature from being “brain dead” or “if you don't change this then the feature is dead”. We see this a lot with areas around “discoverability” for example—people want to put something front and center as a way of fixing something. We also see many suggestions along the lines of “make it configurable”. Both of these have benefits in the near term of course, but both also add complexity down the road in terms of configurations, legacy user interface, and so on. Often it is important to look at the benefit in a broader context such as how frequently something will be executed by a given person or what percentage of customers will ultimately take advantage of the improvement. It is not uncommon internally to see folks extrapolate instantly to “everyone does this”!

- **How big is the change?** Early in the product cycle we are making lots of changes to the code—adding new code, rearchitecting, and moving things around a lot. We don't do so willy nilly of course but the reality is that early in the cycle there is time for us to manage through the process of substantially changed code and the associated regressions that will happen. We write specifications and have clear views of features (scenario plans, prototypes, and so on) because we know that as the project progresses the cost of making big changes of course goes up. The cost increases because there is less time, but also because big change late in the cycle to a large system is not prudent engineering. So as we consider changes we also have to consider how big a change is in order to understand the impact across the system. Sometimes change can be big in terms of lines of code, and lots of code is always risky. But more often the change is not the number of lines, but the number of places the code is connected—so while the change sounds like a simple “if” statement it is often more complex than that. Over the years, many have talked about componentization and other systems engineering ways to reduce the impact of change and of course Windows is very much a layered system. The reality is that even in a well layered system, it is unlikely one can change things at the bottom and expect no assumptions of behavior to carry forth through subsequent upper layers. This “defensiveness” is an attitude we have consistently throughout our development process because of the responsibility we feel to maintain compatibility, stability, performance, and reliability.
- **How costly is the change relative to the benefit?** Change means something is different. So any time we change something it means people need to react. Often we are deliberate in change and we see this in user interface, driver models, and so on. When new are deliberate people can prepare and we can provide tools to help with a transition. We've seen a lot of comments about new features that react to the cost of change. Many times this commentary is independent of the benefit and just focuses on the change itself. This type of dialog makes it clear that change itself is not always good. With many bug reports we hear “this has been in Windows for 3 versions and *must* be fixed in Windows 7”. Over many releases of Windows we have learned that behaviors in the system, particularly in APIs, message order and semantics, or interfaces might not be ideal, but changing them introduces more complexity, incompatibilities, and problems for people than the benefit of the change. Some view these decisions as “holding us back” but more often than not it would be a break from the past one day only to create a new past to break from the next. The existing behavior, whether it is an API or a user interface, defines a contract we have and part of building a release is making sure we have a well understood cost/benefit view, knowing that as with any aspect of the system different people will have different views of this “equation”.
- **In the context of the whole release, how important is this issue?** There is the reality that all decisions need to be made in the context of the broader goals of the release. Each release stands for a set of core scenarios and principles that define the release. By definition it means that each release some things will change more than others and some things might not change at all. Or said another way, some parts of the system will be actively worked on towards a set of goals while we keep other parts of the system more or less “stable” release over release. It means that things you might want to see changed might not change, just because that is an area of the product we're not mucking with during Windows 7. As we've talked about, for Windows 7 we put a lot of work into various elements of system performance. Aside from the obvious scenario planning and measurement, we also took very seriously areas of the system that needed to change to move us forward. Likewise, areas of the system where the performance gain would not be significant enough to warrant change do not change that much. We carry this forward through the whole cycle as we receive data and telemetry.
- **How does the change impact security, reliability, performance, compatibility, localizability, accessibility, programmability, manageability, customizability, and so on?** The list of “abilities” that it takes to deliver windows is rather significant. Members of our development team receive ongoing training and information on delivering on all of these abilities so we do a great job across the product. In addition, for many of these abilities we have members of the team dedicated full time to delivering on them and making sure across the product we do a good job. Balancing any change or input against all of these abilities is itself a significant undertaking and an important part of the research. Often we see input that is very focused on one ability which goes counter to another—it is easy to make a change to provide customization for example, but then this change must also be customizable for administrators, end-users, and PC makers. Such complexity is inherent in the very different scenarios for usage, deployment, and management of PCs. The biggest area folks see us considering this type of impact is when it comes to changing behavior that “has been in the product forever”. Sometimes an arbitrary decision made a while back is best left as is in order to maintain the characteristics of the subsystem. We know that replacing one old choice with a new implementation just resets the clock on things that folks would like to see be different—because needs change, perspectives change, and people change.

These are just a few of the factors that go into considering a product change. As you can see, this is not something that we take lightly and a lot goes into each and every change. We consider all the inputs we have and consider all the data we can gather. In some ways it is easy to freeze thinking about the decisions we must make to release Windows 7—if you think too hard about a decision because you might start to worry about a billion people relying on something and it gets very tricky. So we use data to keep ourselves objective and to keep the decision process informed and repeatable. We are always humbled by the responsibility we have.

While writing this post, I received a “bug report” email with the explicit statement “is Microsoft going to side step this issue despite the magnitude of the problem” along with the inevitable “Microsoft never listens to feedback”. Receiving mail like this is tough—we're in the doghouse before we even start. The sender has decided that this report is symbolic of Microsoft's inability or lack of desire to incorporate *critical* feedback and to fix *must fix* bugs during development. Microsoft is too focused on shipping to do the *right* thing. I feel like I'm stuck because the only answer being looked for is the fix and anything less is a problem or further proof of our failure. And in the back of my mind is the reality that this is just one person with one issue I just happen to be talking to in email. There over a couple of million people using the beta and if each one, or for that matter just one out of 10, have some unique change, bug fix, or must do work item we would have literally years of work just to make our way through

that list. And if you think about the numbers and consider that we might easily get 1,000,000 submitted new “work items” for a product cycle, even if we do 100,000 of them it means we have 900,000 folks who feel we don’t listen compared to the 100,000 folks who feel listened to. Perhaps that puts the challenge in context.

With this post we tried to look at some of the ways we think about the feedback we’re getting and how we evaluate feedback in the course of developing Windows 7. No area is more complex than balancing the needs (and desires) of such a large and diverse population—end-users, developers, IT professionals, hardware makers, PC manufacturers, silicon partners, software vendors, PC enthusiasts, sysadmins, and so on. A key reason we augment our approach with data and studies that deliberately select for representative groups of “users” is that it is important to avoid “tyranny of the majority” or “rule by the crowd”. In a sense, the lesson we learned from *adrenaline-based* development was that being systematic, representative, and as scientific as possible in the use of data.

The work of acting on feedback responsibly and managing the development of Windows through all phases of the process is something we are very sincere about. Internally we’ve talked a lot about being a learning organization and how we’re always learning how to do a better job, improve the work we do, and in the process work to make Windows even better. We take this approach as individuals and how we view building Windows. We know we will continue to have tough choices to make as everyone who builds products understands and what you have is our commitment to continue to use all the tools available to make sure we are building the best Windows 7 we can build.

--Steven

# Some Changes Since Beta for the RC

Steven Sinofsky | [2009-02-26T03:00:00+00:00](#)

---

*We've been quite busy for the past two months or so working through all the feedback we've received on Windows 7. It should be no surprise but the Release Candidate for Windows 7 will have quite a few changes, many under the hood so to speak but also many visible. Some have asked if the featureset is "frozen" then what will we change—we change a lot of things in the beta based on feedback and we try to do so in a systematic manner with the focus on the goals for the release. The goal of having a fully functional Beta was to make sure we received reliable feedback and not a lot of "hey this doesn't work at all" sorts of reports. This has allowed us to really focus on delivering a refined RC where the changes we made are all the reflection of feedback we have received.*

*Building on the previous post that looked at the broad view of feedback, we want to start posting on the feedback and the engineering actions we've taken in responding to the feedback. We won't be able to cover all the changes (as we're still busy making them), but for today we wanted to start with a **sampling** of some of the more visible changes. We're still on the same path working towards the release candidate and of course we know everyone is anxious for the next phase of our path to RTM. In the meantime, our full time machines are still running the Beta build.*

*Today's post is from Chaitanya, who has previously posted on some of the core user interface work. --Steven*

This blog post talks about a few of the improvements that will be in our Release Candidate (RC) based upon customer feedback. There are many under the hood changes (bug fixes, compatibility fixes, performance improvements, and improvements) across the entire dev team that we just don't have room to discuss here, but we thought you'd enjoy a taste of some changes made by three of our feature teams: Core User Experience, Find & Organize and Devices & Media. The comments in this article come from a variety of verbatim sources, with identifying information withheld.

## Desktop Experience

### *1. Windows Flip (ALT + TAB) with Aero Peek*

We've received overwhelmingly positive feedback about Aero Peek and how it helps customers switch windows with increased confidence. Daniel wrote to tell us "I'm wondering why Peek was never implemented for the ALT + TAB window. The thumbnails look/ behave the same way as the taskbar thumbnails when you hover the mouse over them. It seems logical that they would exhibit the peek behavior, too". We decided to make this change since we heard many requests for it. One can still quickly flip between and cycle through running windows using the ALT+TAB keys, but when more window information is needed Aero Peek will appear. This is triggered by a time delay as you pause while keyboarding through running windows.

Fig 1.

*Aero Peek triggered from Windows Flip (ALT+TAB)*



## 2. Windows Logo + <#> keyboard shortcut

Enthusiasts often ask us for more keyboard shortcuts to simplify their common tasks. Efficiency is key. We've answered with a very powerful new keyboard shortcut for the taskbar that may just alienate mice everywhere. Pressing Windows Logo + <#> (where <#> corresponds to an item's order in Quick Launch) in Vista would simply launch the item. As part of our unification of Quick Launch with the taskband in Windows 7, we now beef up the shortcut so it can both launch *and* switch. For example, if IE wasn't running in Fig 1 then Windows Logo + 2 will launch the program (as it did in Vista). If IE is running with a single window, the same shortcut will now switch to the program. The magic really begins when IE is running with several windows or tabs—holding down the Windows Logo and tapping the 2 key repeatedly will actually cycle through the open IE items off the taskbar (with Aero Peek, of course). Letting go simply switches to the corresponding window. Think of this as per-program ALT+TAB shortcut for the first 10 items on the taskbar. If you need a new instance for IE, simply use SHIFT + Windows Logo + <#>. A program's Jump List may also be accessed via ALT+ Windows Logo + <#>. Finally, you can even flip back to the last active window of a program by using CTRL+ Windows Logo + <#> (this also works by holding CTRL with a mouse click on a taskbar button). Keyboard aficionados rejoice!

## 3. Needy State

"Needy window" is the internal term we use for a window that requires your attention. Since the '90s, the taskbar has always provided some type of visualization to alert the customer to this state such as by flashing the button. A careful balance must be struck between providing information and not irritating the customer. With the new taskbar, we received feedback that Outlook reminders or a Messenger chat sometimes went unnoticed because needy windows were too subtle. For example, Mudassar opened a bug to say "The flashing is not obvious enough to get user's attention. Sometime I don't even notice it. It flashes for a little bit and then stops. If I am away the icon flashes and stops before I come back. The icon is not noticeable." We've made three changes that should address the issue. First, we changed the flashing animation curve to make it more noticeable (from a sine to a sawtooth wave). Second, we used a bolder orange color. Finally, we wanted to double the number of flashes which is currently set to three. As a nod to Windows 7, we decided to go with seven flashes instead.

## 4. Taskbar "Open With"

Quick Launch always supported the ability to drop a file onto a pinned program and have it open with that program. The new taskbar on the other hand, always treats a drop as a pin command. Drop a program and the program is pinned. Drop a file and the file will be pinned under its respective program's Jump List and that program automatically gets pinned to the taskbar. It was important for us to keep drag/drop consistent. We believe that for most cases people will open files through the desktop by just double-clicking them or from the Jump List and the default program will open. However, there are some scenarios when a customer wants to open a certain file type with another program. We heard this feedback and decided to revive "Open With" drag/drop on the taskbar with a keyboard modifier. One can hold down SHIFT and drop the file on the desired program.

## 5. Taskbar scaling

We've reclaimed lots of space on the taskbar by unifying launching/switching, by collapsing open windows and by cleaning the notification area. Still, some have asked for even more room to pin the programs they use regularly. We've made a change to squeeze in 24-39% more icons before the taskbar scrolls; depending upon your resolution, icon size and assuming the default notification area. Table 1 illustrates the new button capacity before the taskbar begins to scroll as well as the capacity growth since Beta. We believe customers will find more than enough room to pin their common programs.

Table 1.

*Maximum taskbar button capacity before scrolling*

<b>Resolution</b>	<b>Large Icons</b>	<b>Small Icons</b>	<b>% Increase from Beta (large/small icons)</b>
800x600	10	15	25% / 36%
1024x768	15	22	25% / 38%
1280x1024	20	29	25% / 32%
1600x1200	26	39	24% / 39%



## 6. Anchoring taskbar thumbnails

Hovering or clicking on a taskbar button surfaces all the running windows for that program. Upon seeing a set of open thumbnails, Kozlow asked “How do I know which application has opened the thumbnails group?” In other words, the thumbnails didn’t appear visually connected to the taskbar. We made a visual update that now keeps the color hot-track effect on when the mouse is over a thumbnail. In fig 2 you can see that IE retains its blue Color Hot-track visual even though the mouse is over a thumbnail.

Fig 2.

*Color Hot-track stays active when the mouse hovers over taskbar thumbnails*



## 7. Newly installed programs

“Customer in control” is so strong a mantra for Windows 7 we don’t even allow programs to pin themselves to the taskbar when they are installed. This is a task expressly reserved for the customer. We’ve gotten some requests to make this goal a bit easier so now when a program is installed, it is automatically and temporarily surfaced at the bottom of the Start Menu. The customer can easily discover this new addition, launch it directly and optionally drag it to the taskbar for convenient access in the future.

## 8. Jump List length

Jump Lists are proving to be a valuable tool to quickly jump to commonly access files, folders, links and tasks. Steve filed a bug in which he said “The whole point of the jump list is to make it easier to jump to your favorite locations. However, it doesn’t save me time having to scan through a long list of frequent locations.” In other words, sometimes it’s hard to parse an item when the list gets too long. Our telemetry data informs us that in most cases customers are clicking on the first 10 items. Therefore, we’ve updated Jump Lists so that only a maximum of 10 items may be automatically suggested (this doesn’t apply to tasks or pinned items). Don’t worry—there’s even a setting for enthusiasts to customize the length of the list.

## 9. Increased pinning flexibility with Jump List

For organizational, scaling and identification purposes, the taskbar is designed to hold files, folders and links in a program's Jump List. Items can only be pinned to the Jump List of programs registered to handle that file type. Based on feedback we've received we now allow one to pin items to a Jump List belonging to a program that isn't registered to handle that file type. Better yet, pinning the item in most cases will create a new registration so that launching it from the Jump List will always open the file with that specific program. For example, one can pin an .HTML file to Notepad's Jump List and when clicked on from the menu, the file will always open in Notepad even though IE by default handles the file type.

#### *10. Desktop icon and gadget view options*

Windows 7 makes gadgets far easier to manage, view and access by building them directly into the desktop. David's feedback matches what others were telling us: "In Vista, I was able to hide desktop icons while my gadgets were still visible and available. I liked this feature in Vista, especially with all the icons that are constantly dropped on the desktop by app installers. I don't want to see the icons, but I still want to see my gadgets." In Beta it was impossible to separate desktop icons from gadgets under the View setting available by right-clicking on the desktop. We made a change to afford independent control to each so that one can opt to hide just her gadgets or just her desktop icons.

### **Touch**

#### *11. Aero Peek for touch*

We're excited about Peek and we further refined its functionality. Our touch customers enjoy the benefits of direct manipulation, but inform us they feel left out of some of new functionality that's available for the mouse and keyboard. We've made two improvements that spreads the love. First, the taskbar's thumbnails now support a touch gesture so one can drag her finger across the UI and trigger Aero Peek. Also, the Show Desktop button is improved so a press-and-hold will allow the customer to peek at the desktop. A regular tap in both these scenarios still commits the switch.

#### *12. Multi-touch touch keyboard*

A funny thing happens when one uses touch to interact with a software keyboard for the first time. The natural instinct is to press multiple buttons simultaneously like they do with a real keyboard. It's quite reasonable to try to use SHIFT + <letter> to capitalize, for example. RC ushers in multi-touch support for the Touch Keyboard so that customers enjoy a more realistic experience.

#### *13. Multi-touch right-click*

People who rely on touch give us mixed feelings towards tap and hold to bring up a context menu. This approach works, but it also involves a slight delay. We now have a fast new multi-touch gesture for right-click. Simply touch an item with one finger and use another finger to tap and summon a context menu.

#### *14. Drag/Drop and selection*

In Beta there was no discoverable way to select text in a website that scrolled both horizontally and vertically. Customers are now able to drag/drop and select items with touch, even inside scrolling pages. The new behavior is optimized for the two most common actions by touch customers—scrolling up and down and dragging left to right.

### **Networking**

### *15. Internet access feedback*

The new network experience from the taskbar's notification area makes it much easier to find and connect to networks. People seem to also really like the wireless signal strength that is available at a glance. In our effort to simplify the experience we removed indications for some advanced scenarios. Based upon feedback, we've decided to introduce a new overlay icon which now reveals when there is a local connection without internet access.

## **Control Panel**

### *16. User Account Control*

If you've been following this blog, then you already know about a recent design change we've made that will prompt for any modification made to the UAC Control Panel. For more information, please refer to the earlier post on [UAC Feedback and Follow-Up](#).

### *17. Locking a machine without a screensaver*

It isn't uncommon for IT administrators to want their corporate machines to auto-lock after a certain amount of time. In Beta, enabling this functionality required a screensaver to be set. We've since made a change to allow this functionality even when no screensaver is specified.

### *18. Faster access to High Performance power plan*

Clicking on the battery float from the taskbar notification area offers two different power plans: Balanced and Power saver. Windows 7 laptops are configured by default to use the Balanced plan since this setting best balances a good experience while promoting more environmentally friendly power use. However, some customers tell us they want to be able to quickly toggle between Balanced and High Performance (yet another power plan). We've taken a change to now show the latter in the flyout menu when it is enabled under the Power Options Control Panel.

### *19. Custom theme improvements*

We've always known customers love personalizing their Windows experience. At the center of this expression of individuality are ingredients such as the desktop background, glass color, sounds and screensavers. In Windows 7 we've introduced themes that make it easy to enable a whole package of default combinations or for customers to save their own creations. However, during Beta we heard feedback along the lines of "I just changed my background or color and I see the change, but I thought it was saved when it really wasn't". We added text under each theme to not only aid in identification, but also to provide feedback on the state of a theme. The new "Unsaved Theme" text also ties better to the nearby "Save theme" command. These tweaks should make personalization a more predictable and enjoyable experience.

## **Windows Media Player**

### *20. Improved Internet Radio playback*

Internet radio playback continues to gain in popularity. We received feedback that sometimes playback of radio streams may be inconsistent depending on network conditions. It's worth noting that our understanding of this issue was greatly helped by the broad scale of usage across so

many customers and network topologies and our telemetry in the Beta. Windows Media Player has made changes to make streaming playback more reliable and resilient.

#### *21. Improved playback support for video content from digital camcorders and cameras*

Customers loved the increased range of formats natively supported by the Windows 7 Beta, but noticed areas where they wanted broader support. For example, one was unable to seek to a specific spot in the video in Windows Media Player or Windows Media Center for AVCHD content that was imported from a digital camcorder. We've addressed this. Also, while the support for video from some digital cameras worked great, we also got feedback about supporting a broader set of devices out of the box. We've since added support for Windows Media Player to natively support the .MOV files used to capture video for many common digital cameras.

#### *22. Cleaner Now Playing view*

Customers are sharing positive reviews of Media Player's new light-weight Now Playing view. Still some have asked to make the experience even cleaner. We've responded with a visual update that is more lightweight and compact.

#### *23. Filtering content that cannot be played*

Media Player's library view is designed to surface and showcase one's content. However, in some cases items were displayed that couldn't be played. For example, Apple's lossless .M4A or .H263 MPEG-4 content would be shown in a library even though Media Player could not play them. In RC, this content will no longer appear in the library view so that there is better expectation of what is supported by the player.

#### *24. Resume from sleep*

Customers are used to resuming a CD or DVD after an interruption. With customers choosing new low-cost, smaller form-factor, machines without optical drives, an increasingly popular scenario is to have content played directly from the hard drive. In Beta, it was not possible to resume playback on such content after a laptop goes to sleep. Customers assume the experience should match that of physical media so we fixed the experience to meet this expectation.

#### *25. Quieting Windows Media Player sync relationships*

When Media Player is open and a portable media player or a USB drive is inserted, we trigger a dialog to determine whether a sync relationship should be created with the new device. Our original goal was to be proactive and help customers make a decision in context, but we received comments that this experience is jarring. As a result, we will no longer interrupt when the player is running. This is consistent with our "customer in control" goal of Windows 7 and we trust people can manually configure this should they wish to.

#### *26. Easier access to advanced settings*

What enthusiast doesn't want to tweak her player settings? This was echoed by several comments so we've made it easier to access and adjust settings. The equalizer, play speed, SRS WOW and other options are now surfaced via the Now Playing context menu under Enhancements.

#### *27. Jump List improvement*

Media Player's Jump List provides quick access to the content customers consume. The list becomes even more powerful and complete in the RC now that we also include items launched from Explorer.

## **Device Stage**

### *28. Enriching the Device Stage ecosystem*

Customers have been so positive about the new Device Stage experience, one of the biggest pieces of feedback we got was "Why aren't even more of my devices supported?" We've taken that feedback to heart and then took the feedback to our IHV and OEM partners to get their support for more devices. Our hardware partners in turn asked us to make it easier to integrate with the Device Stage and we worked with them on improvements. Although Windows already supports tens of thousands of devices, customer feedback on the Beta introduces even more device support in RC via the new Device Stage experience.

## **Sound UX**

### *29. Improving the headphone experience*

Customers informed us that sometimes their audio streams did not properly move from the default speakers to their headphones. The fix required an update to the algorithm we use to detect new devices. In RC the transition works more reliably.

### *30. Increased audio reliability*

In some cases people reported not having any audio device after installing Beta. The problem is that some audio hardware does not work out of the box with our inbox audio class driver. Amazingly there are over 26,000 custom audio drivers and while many are on Windows Update, many are still not. The Release Candidate tightens the Windows Logo test to better ensure clean install delivers baseline functionality for speakers and microphones. Furthermore, we will continue to populate Windows Update with frequently needed drivers.

## **Windows Explorer and Libraries**

### *31. Improved header*

It is great to see customers realize the convenience and power of libraries. Having files aggregated into one convenient view, without worrying where they are all physically located, simplifies many scenarios. The library header in Beta showed only a static string that reflected how many locations were represented as part of the library. We heard feedback that this wasn't very clear and more importantly, customers preferred to have more information so that they could be better orientated themselves. The RC will introduce a new header that updates to reveal the subfolder as one browses a library. Furthermore, the "Arrange by" views are better expose in the upper right, in proximity of the other view and search controls.

### *32. Reduced confusion with drag/drop*

The Release Candidate will remove the ability to drag/drop a folder into the Libraries node in the Explorer navigation pane. We know some liked this functionality to create a new library, but it also presented some serious design issues. For example, some were surprised to find a new library

was created when their intent was to simply copy the folder. More seriously though, there were circumstances where people then deleted the original folder thinking it was already copied. Data loss is a grave concern of ours and we don't want customers to suffer from such a mistake. Don't worry though—one can still easily create a new library using the "New Library" or "Include in Library" commands in the Explorer command bar.

### *33. Reviving familiar entry points*

Mando writes, "In Win7 the Win+E shortcut opens an explorer window but the path is "Libraries" instead (which isn't where I want to go most of the time). Is there a way to configure the target folder of "Win+E" or is there an alternate shortcut that will get me to the "Computer" path like it did in Vista?" RC reverts the behavior and now the shortcut will launch the "Computer" Explorer. Also, we changed the link in Start Menu -> Ustream to match the Vista behavior.

### *34. FAT32 support*

Local FAT32 hard disk drives were not support in libraries for Beta. RC libraries will now support non-removable FAT32 and NTFS hard disk drives thanks to the feedback we received.

### *35. Arrangement view enhancements*

It's been great to see people's reaction to the arrangement views in libraries. Being able to browse using metadata certainly makes quick work of finding files. We've received many requests to further enhance the arrangement views in a variety of ways and we've made a number of changes in response to them. For starters, RC makes it easier to switch arrangement views—one can now do so directly from the view context menu, which is the familiar home of switching the view mode, sorting, and grouping. Second, the specific arrangement views themselves have been enhanced for RC. The "Month" and "Day" views in the Pictures library now group together both the pictures and videos taken on the same date, whereas previously the videos were split out into a separate group. The "Artist" and "Genre" views in the Music library now show the thumbnails for up to three unique albums per artist or genre instead of typically just one in Beta. The Videos library now features a Length view that lets customers split out the shorter clips from longer movies in their video collection. Finally, we've made it so that changing the grouping of the Folder view in a library is now remembered just like other arrangement view customization. People who prefer to see their files grouped a particular way no longer have to reset the grouping each time.

## **Performance**

### *36. Improving performance through data*

Feedback comes to us in many different forms. Typically it consists of comments customers share. However, some of the most valuable information actually comes to us automatically when people just use Windows. PerfTrack, for example, is a telemetry system that provides us with invaluable real-world performance data on over 500 different Windows scenarios. The exciting aspect of PerfTrack is that it represents what people are really experiencing "out in the wild". Performance is a very important to both the engineering team as well as to our customers and we strive to continuously improve this area. The topic has been discussed in [several posts](#) on this blog.

Let's look at just one example of a Windows scenario that was improved with the help of PerfTrack. The two graphs below show the performance of opening the Start Menu for both Beta and for a more recent version of Windows 7. Some caveats first—the sample sizes are different (after all Beta did go to a far wider audience) and these numbers shouldn't be taken too literally since they really do just represent a snapshot. The different colors denote performance against the "interaction class"—the acceptable experience range defined by each feature team. In this case we want the Start Menu to appear within 50ms to 100ms. A trace capturing tool running on each machine lets us investigate and fix

what may be impacting performance. The charts shows in Beta 85% of interactions were within the acceptable range (i.e. green or yellow, but not red). After examining the traces and making some optimizations, we find 92% of interactions are this range for a more recent build.

Fig 3.

*Start Menu Open Times for Windows 7 Build 7000 (Beta)*

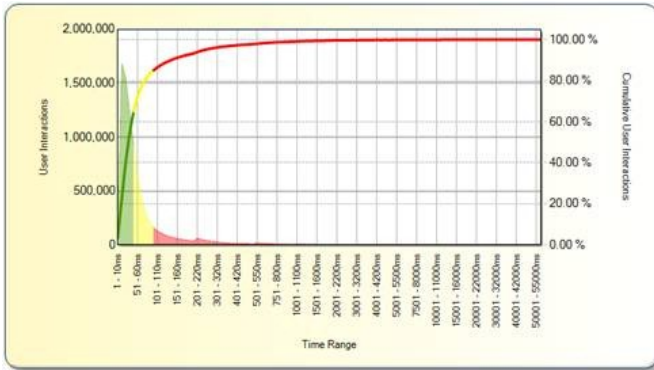
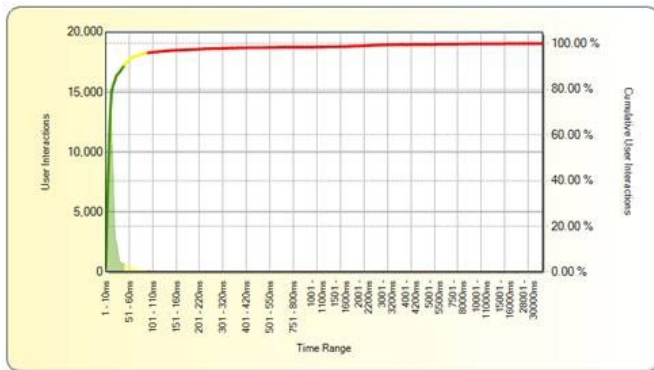


Fig 4.

*Start Menu Open Times for Windows 7 Build 7033*



As is evident from this sample of changes, we've been very busy improving Windows 7 based upon what our customers are telling us in many forums.

- Chaitanya Sareen

# Application Compatibility Testing — Overview

Steven Sinofsky | [2009-03-03T03:00:00+00:00](#)

---

*Delivering a new release of Windows includes a major effort to insure that applications continue to function as well on the new release as they have on the previous release. At the PDC we talked about some of the new areas of Windows Vista that reduced this level of compatibility, such as changes we made around the OS security model. With Windows 7 we renewed our engineering efforts to maintain compatibility. As with device testing, compatibility testing is an effort that spans the entire engineering organization, though we also have a group that is dedicated to this effort. This post was authored by a set of folks and coordinated by Grant George, the corporate vice president for testing in the Windows Experience team. --Steven*

We have taken a very proactive approach to Application Compatibility and our process starts from the moment we first plan our product schedule and design and check in code for Windows and runs through all of our engineering processes and disciplines leading up to our final release (and beyond).

## Application Compatibility Goals for Windows 7

Our main Application Compatibility goal for Windows 7 is to make sure that most all applications which work on Windows Vista will continue to work seamlessly on Windows 7. We do have to be careful about making this claim to be universal because there is a class of applications that are always updated in tandem with a new Windows release. These applications are primarily system utilities, diagnostics, and security software—the common thread is that they make assumptions about the underlying implementation of Windows internals and thus require updates. We carefully coordinate with a large set of ISVs who provide these applications. This was talked about earlier this month as we announced our [Ecosystem Readiness Program](#), which we will discuss more below.

At the start of our product cycle we review our new features and changed designs to ensure that every element of Windows 7 has Application Compatibility in mind. Our engineering process includes automated quality checks to assure public APIs don't change, and our test engineers have the right tools, engineering time and information that is used to find application issues as early as possible in our development cycle. Telemetry information is collected to assess and prioritize the breadth of applications our users depend on, paired with market data and install base information, across a wide variety of software categories to make sure they work as expected in our new OS version.

Below we expand on how we deliver on this goal.

## Engineering Application Compatibility

Rich telemetry from external usage and the broad software marketplace helps us to provide a list of the most popular and critical applications, updated frequently throughout our development cycle. Our engineers then acquire these applications and build automated tests that verify each one works as expected on Windows 7.

Changes that could impact application compatibility are followed up closely, for example **Legacy Code Removal** which involves removing code that existed in the previous release of the product is strictly guarded and tightly managed and should not happen without proper documentation and engagement with the impacted parties. For example if we need to deprecate an API call, the documentation for this API will be updated and we will wait until the impact of the removal is minimal as indicated by telemetry data unless it is required sooner to fix a security issue.

Throughout the development process we are running tests in the background creating an ongoing validation of new code relative to application compatibility. As code is getting ready to be checked into the main build, if a compatibility failure is detected in an automated regression test the checkin is halted. At that point the code is scanned for known compatibility issues and if an issue is detected the developer is asked to fix the problem. Of course we also develop new tests throughout the course of developing Windows 7 in order to broaden our coverage of third party software.



## Engineering & Testing Strategy

We have several teams dedicated to application compatibility as part of the Windows development effort. These teams provides guidance on how to build in application compatibility, provides data on application dependencies, and information on what applications may be impacted for any particular change we make in the Windows platform. These teams also reviews new feature designs, as well as other planned changes, to ensure that the engineering team has fully taken application compatibility into account so that we achieve the tenet of "keeping applications working on Windows 7".

In addition to working with the internal Windows engineering teams, we also reach out to 3rd party developers writing Windows applications to ensure these partners have all the information they need to make their solutions fully compatible with Windows Vista and Windows 7. Furthermore if we do uncover any issues that may need to be resolved by the 3rd party developers, we collect all the information, resources and guidance and engage in a conversation with those external developers to help them understand and fix these issues.

We recently announced the [Windows 7 Ecosystem Readiness Program](#). This program provides partners with access to Windows 7 builds and tools they need to test solutions for Windows 7, Windows Vista, Windows Server 2008 and Windows Server 2008 R2. The Ecosystem Readiness Program also facilitates testing multiple components of the ecosystem together to improve the overall user experience. Rather than just focusing on getting a specific OEM product, software application, or hardware device certified, we will be bringing multiple components together to verify a rich user experience that delivers quality, reliability, and performance as well as innovation through new feature adoption.

As mentioned earlier we use telemetry data and market share information to choose the applications we directly test on as part of our compatibility efforts, below you can find the a sample of the Consumer Scenarios we focus our testing on:

- Communications
- Gaming
- Fundamentals like Setup, Security and Performance
- Memories
- Music
- Productivity
- TV/Movies
- Data Backup/Security
- Mobility
- Financial Management

*To view a full list of Application Categories covered please take a look at the Appendix section at the end of this post. While we would love to provide the detailed list of products, several of the sources of this information are based on proprietary research from third parties.*

Another very important set of technologies we test on are the middle tier technologies like Java, the .Net Framework, etc. to make sure the applications that use these technologies continue working as expected

In addition to 3<sup>rd</sup> party stand alone applications we test a subset of OEM pre-installed software and their inbox applications for compatibility. The software tested come from the engagements we have with our OEM partners and their submitted installation images. These images are tested on clean installations of Windows 7 and upgrades from Windows Vista on OEM standard hardware. This level of coverage allows us to best replicate the initial experience with Windows 7 for many of our customers. Because many of these applications are closely aligned with the OS, hardware and drivers, it is not unusual for an OEM to provide updates to this software with a new OS release.

In addition to the above mentioned testing approaches, Microsoft IT maintains a software portfolio of approximately 1,500 applications. These applications must be tested prior to software deployments inside Microsoft.

Microsoft IT developed an application-tracking method that simplified the process of selecting applications for sample-based testing. By identifying groups of applications that have similar data processing, controls, underlying technology, and methods, Microsoft IT is able to test approximately 4 to 6 percent of the total applications and gain a reasonable assurance of compatibility for all.

For more information you can read the [LOB Application Compatibility Technical White Paper](#).

## Validation Strategy

Part of our testing process includes the creation of scenarios that we validate on 3<sup>rd</sup> party applications. We approach this by verifying the intended functionality of the application while focusing some of our attention to changes in the OS, new functionality and risky integration areas. Manual and automated test passes are scheduled to cover the identified scenarios and to verify the user experience. We cover the applications on different sets of hardware and make sure that that we cover a lot of different configurations, x86, x64, Intel, AMD, touch and multi-touch machines, etc.

We use specific categories when measuring the compatibility of the applications we test on:

- **Excellent** - No compatibility problems
- **Poor** - Minor functionality is not working
- **Failure (regression)** - It used to work in Windows Vista now it is not working properly
- **Failures (non-regression)** - The application does not work on Windows 7 but it did not work on Windows Vista either
- **Accepted Regression** - We know of changes in the OS that may affect the applications and we have/are working with ISVs on mitigations for these issues. A simple example is version number. An application that is hard coded to only work on specific versions and it doesn't know what our new operating system version is would fail.

We will cover these categories in more detail in subsequent blog posts about how we manage application compatibility.

As part of our testing we do find issues that may need to be resolved by the 3<sup>rd</sup> party developers at the companies that develop and sell these applications. We collect all of the relevant technical information, resources and guidance and engage in a conversation with those external partners to help them fix the issues. Of course we also engage them in our technical beta programs so they can test Windows 7 compatibility with their products at the same time we do.

## Final Word

Application compatibility is very important to the Windows team. We are constantly working to improve your experience with applications as you move from one release of Windows to the next. We encourage you to try our Windows 7 beta release to experience the improvements in the application compatibility space and we want to hear your feedback.

It is worth mentioning the work we have done from an end-user perspective to assist in application compatibility. Many failures of application compatibility happen at install time and to assist with this, in Windows 7 we have improved the detection of failed installations and provided a step by step wizard which will help to get an application's "compatibility mode" correctly set. We also provide real-time problem reports and solutions that can help you locate an updated version or patch. Many of you might have experienced this when trying to run Skype after an upgrade or install the current version, and in both cases you were automatically referred to the Beta download site.

## Appendix

### Microsoft Covered Consumer Scenarios & Application Categories

Consumer Scenarios	Application Categories
<b>Communications</b>	Communication, Internet, Network, Security, Security Suite, Server Suite
<b>Gaming</b>	Action, Adventure, Arcade, Children's Entertainment, Family Entertainment, Fighting, Flight, Lottery, Other Games/Compilations, Racing, Role-Playing, Shooter, Sport Games, Sports, Strategy
<b>Fundamentals</b>	Application Utilities, CD/DVD Utilities, Disk Back-up, Disk Utilities, Disk/Tape/File Translation, File Utilities, General System Utilities, Printer Utilities, Security, Security Suite, System Utilities Bundle, System Utilities Suite, Virus Detection, Y2K Solutions
<b>Memories</b>	Camera Utilities, CD/DVD Utilities, Home Graphic/Photo Editing, Home Graphics, Pro Graphics Suite, Pro Video Editing/Compositing Suite, Professional Graphic/Photo Editing, Video Editing
<b>Music</b>	CD/DVD Utilities, Music, Music Education
<b>Productivity</b>	Astrology, Atlas, Auto/Transportation, Bible/Religion, Buying Guide, Career Development, Checkwriting, Children's Entertainment, ClipArt, Collecting, Compression, Cooking, Creativity, Critical Thinking, Culture, Desktop Publishing, Dictionary/Thesaurus, Document Generation, Educational Bundle, Electronic Book, Encyclopedia, Family Entertainment, Fonts, Foreign Language, Forms, Gardening, Genealogy, General Business, Geographic Info System, Geography, Graphic Images, Health/Nutrition, History, History/News, Hobbies, Home Bundle, Home Design, Home Graphic/Photo Editing, Home Graphics, Home Repair, Labels, Language Arts, Legal, Literature, Lottery, Math, Medical, Memory Manager, Multi-Subject, OCR, Office Productivity Suite, Other Education, Other Occult, Parenting, Personal Finance, Personal Improvement, Pets, Presentation Software, Programming, Project Management, Reading, Science, Screen Saver, Server Suite, SOHO, Spelling, Spreadsheet, Study Skills, Tax, Text/Image Management, Training, Travel, Typing, Voice Recognition, Web Development/Publishing, Wedding, Word Processor, Word Processor Add-On, Writing
<b>TV/Movies</b>	CD/DVD Utilities, Movie/TV, Pro Video Editing/Compositing Suite, Professional Graphic/Photo Editing, Video Editing
Small Business Scenarios	NPD Categories
<b>Data Security and Backup</b>	Application Utilities, CD/DVD Utilities, Compression, Diagnostic, Disk Back-up, Disk Utilities, Disk/Tape/File Translation, File Utilities, General System Utilities, Memory Manager, Network, Security, Security Suite, Server Suite, System Utilities Bundle, System Utilities Suite, Virus Detection, Y2K Solutions
<b>Mobility</b>	Communication, Internet, Network
<b>Financial Management</b>	Accounting, Checkwriting, Forms, General Business, Office Productivity Suite, Payroll/TimeSlips, POS, SOHO, Spreadsheet, Tax
<b>Sales &amp; Marketing</b>	ClipArt, CRM/Sales Tools, Desktop Publishing, Document Generation, Fonts, Forms, General Business, Labels, OCR, Office Productivity Suite, POS, Presentation Software, Printer Utilities, SOHO, Text/Image Management, Web Development/Publishing, Word Processor, Word Processor Add-Ons

*Our list includes the top 50% best selling applications in the past 24 months, some of this data is collected and aggregated from several well-known third party research information providers.*

# Beta to RC Changes – Turning Windows Features On or Off

Steven Sinofsky | [2009-03-06T03:00:00+00:00](#)

---

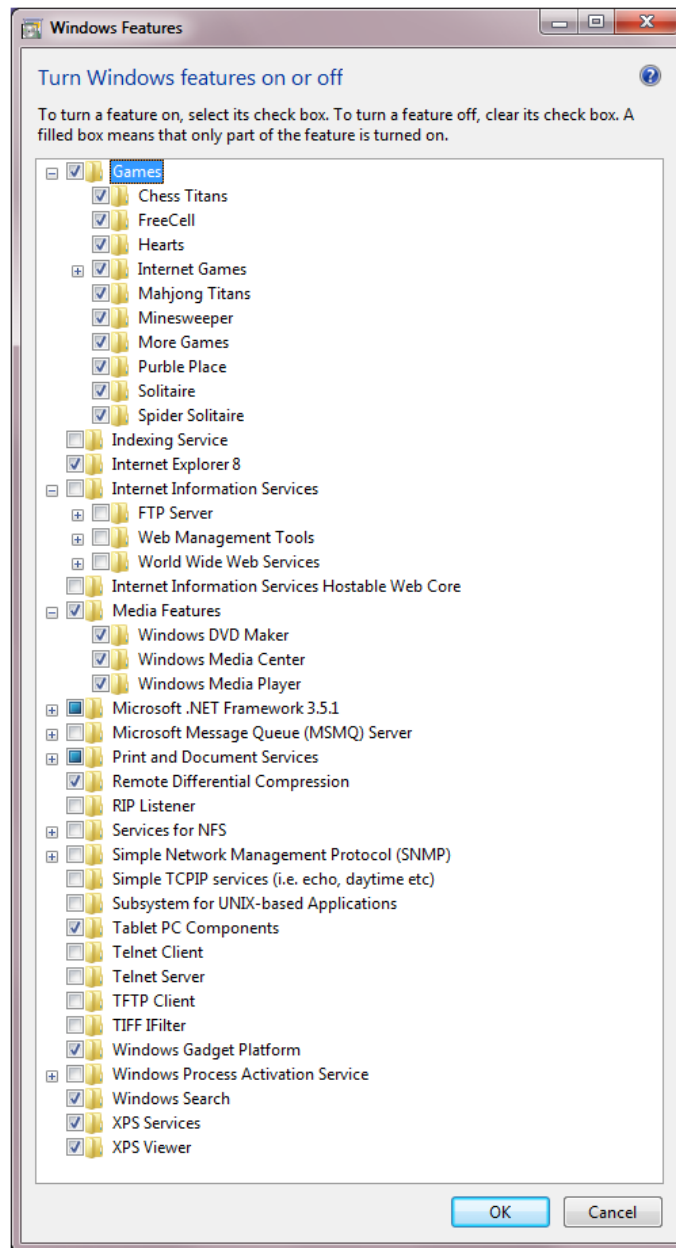
*The theme of “choice and control” has been applied in many aspects of how we have designed Windows 7. We’ve certainly received lots of positive feedback about the theme and about the choices we’ve made in the design, and we’ve also received a few suggestions for how we might continue to implement this theme in the future. We’ve received feedback for features that should be even more customizable (such as Explorer or the logon screen) or features that should be added to Windows (such as a PDF format reader, security tools, or disk utilities). And we’ve received feedback that some users might prefer to run Windows without certain features. This post is about a point of choice and control in the Windows 7 control panel called “Windows Features” which is where you can choose to turn various features of Windows on or off. This continues our discussion of changes we have made based on feedback from the Beta as we progress to the Release Candidate. This post is by Jack Mayo who is the group program manager for our Documents and Printing team and also worked on Internet Explorer 8. –Steven*

“Turning Windows Features On or Off” has a long history in Windows, going back to the earliest days of the 32-bit code base. We’ve received a lot of suggestions about features that you would like to turn on or off using your own criteria for choice. For Windows 7 we’ve engineered a more significant list of features and worked to balance that list in light of the needs of the broad Windows platform as well. We want to provide choice while also making sure we do not compromise on compatibility by removing APIs provided for developers. We also want to strike the right balance for consumers in providing choice and balancing compatibility with applications and providing a consistent Windows experience.

We know many have specific ideas of what constitutes a “feature” or a “program” in Windows and what constitutes an identifiable “part” of the operating system, and yet we also know different people can have different points of view, often strongly held. Some might take an end-user approach and identify a feature based on a window or start menu shortcut. Some might take an approach based on one perspective of architectural subsystems, such as storage or security. Some might take an approach based on what to some are alternate choices to some similar functionality. All of these are valid in some context, but would not result in consistently identifying “features” considering these varied points of view. As engineers we know that no software system can be decomposed into an arbitrary set of layers or parts and any decomposition is likely to change over time.

We don’t want the discussion about this feature or these choices to digress into a philosophical discussion about the [definition of an operating system](#), which is ultimately a challenging exercise (judging by the revision history on the community page), but we do want to improve a feature centered on helping to meet the feedback [expressed by some](#) over the summer when this blog started.

In the Release Candidate for Windows 7 we have extended the control panel called “Windows Features” which is available from the standard “Programs and Features” control panel (we often call this ARP, for the original name of Add/Remove Programs). This location is unchanged from Vista and XP, though the wording has been clarified. In Windows 7 if you bring up the Windows Features control panel by clicking on “Turn Windows Features on or off” (or just typing “Windows features” in the start menu) you will see the following in the Release Candidate (by default the hierarchy is not fully expanded, but in this screen shot I’ve expanded some elements for additional information):



For those familiar with the Vista version or the Beta version of this dialog you will notice the list has grown. Let's talk about what we've added and briefly how it works.

If a feature is deselected, it is not available for use. This means the files (binaries and data) are not loaded by the operating system (for security-conscious customers) and not available to users on the computer. These same files are staged so that the features can easily be added back to the running OS without additional media. This staging is important feedback we have received from customers who definitely do not like to dig up the installation DVD.

For any of the features listed you can change the state to enable it or disable it. The Vista and Windows 7 beta control panel lists a wide range of features. Some are targeted towards Developers working on a client workstation (IIS, MSMQ, etc.), others are utilities for network administrators and enthusiasts (RSM, SNMP, Telnet, etc.), and some are features customers have asked us to make optional (Games, Fax and Scan, Tablet PC

components).

In Windows 7 we are expanding the number of features you have control over in this regard, giving customers more control, flexibility and choice in managing the features available in this version of Windows. In addition to the features that were already available to turn on or off in Windows Vista, we've added the following features to the list in Windows 7:

- Windows Media Player
- Windows Media Center
- Windows DVD Maker
- Internet Explorer 8
- Windows Search
- Handwriting Recognition (through the Tablet PC Components option)
- Windows Gadget Platform
- Fax and Scan
- XPS Viewer and Services (including the Virtual Print Driver)

It is worth describing the details of "remove" since this too is a place where there are engineering and customer decisions to be made. We've already seen one decision which is to make sure we keep the features staged for future use so that a DVD is not required. A second decision is that we also continue to support the APIs available for features where these APIs are necessary to the functionality of Windows or where there are APIs that are used by developers that can be viewed as independent of the component. As many of you know these are often referred to as "dependencies" and with Windows the dependencies can run both internal to Windows and external for ISVs.

It should be no surprise, but when we develop new features in Windows we tend to use the underlying infrastructure and associated APIs rather than duplicate code which would create extra working set, slow performance, and increase the surface area that needs to be secured, etc. We all know code reuse is a good engineering practice. As a platform, Windows tends to emphasize the creation of APIs for many systems, even when those subsystems are viewed as part of a larger system. When we have APIs that are used, we faced the choice of breaking software that just expected those APIs to be there or to continue to support the API. When we continued to support the API our approach was to remove a feature by making sure that an end-user could not invoke the feature via traditional end-user mechanisms. These are often difficult decisions as we work to balance the expectations of developers, the shared desire to deliver a robust release of Windows 7, and to maintain the goals set out by the feature "Turn Windows Features On or Off". Because there are so many combinations of dependencies just represented in this list, selecting some options might provide you with some explanation as to the challenges in selecting a combination (for example Windows Media Player and Windows Media Center share a lot of code so turning one off might introduce a pretty complex situation for the average end-user).

Finally, we know some have suggested that this set of choices be a "setup option". Some operating systems do provide this type of setup experience. As we balanced feedback, the vast majority of feedback we have received was to streamline setup and to reduce the amount of potential complexity in getting a PC running. We chose to focus this feature on the post-setup experience for Windows 7.

--Jack

# Application Compatibility Testing — International

Steven Sinofsky | [2009-03-09T03:00:00+00:00](#)

---

*This post continues the discussion of Compatibility testing from our test team. --Steven*

In the previous blog post "*Application Compatibility Testing for Windows 7*" we talked about the importance of Application Compatibility and work we are doing to engineer this in Windows 7. In this post we will examine the challenge that emerges as we consider the world wide audience that Windows serves.

This blog post will cover the following areas:

- Overall International App Compatibility Strategy
- Approach to International App Compatibility
- Application Acquisition
- Testing Applications
- Measuring our Success
- What it means to “Rescue An Application”

For Windows 7 we have made significant investment in application compatibility, ensuring applications that worked on Vista, continue to work on Windows 7 and we’ve also rescued some applications that were broken in Vista to work on Windows 7 (more on that later). As we’ve talked about, there are some applications that are OS version specific by design (utilities, firewalls, security, etc.) and those are not included in this discussion.

## Approach

One of the biggest challenges in International Application Compatibility is what applications we test, the scale of testing, and what it means for us to say that an application “works”. For Windows 7 we are testing over 1200 applications across 25 specific markets. We have improved our coverage over Vista by adding over 300 more international applications.

We look at applications in 3 buckets.

1. **Global ISV (GISV) Applications** – Localized software sold by major ISVs in several international markets as well as the United States.
2. **Microsoft (MS) Localized Applications** – Microsoft software that has been localized for use in other markets other than the United States.
3. **3<sup>rd</sup> Party Local Applications** – Software where the user interface language is not-English and the application is sold in non-English speaking markets (for example, IchiTaro – Japanese Word Processor, Илпыс 8 – Russian ERP system,)

Categories 1 & 2 are pretty straightforward. There are a known set of key applications and scenarios used around the world and we must ensure these applications function in Windows 7. Category #3 is where there is some complexity.

The applications list we build for 3<sup>rd</sup> Party Local Applications is built using a number of methods. First, we build on the list of applications we have used in previous versions Windows (XP/Vista, etc). If it worked on Vista, it must work on Windows 7.

Next we work with our teams in markets around the world to rank top applications in particular markets. It is amazing to see the diversity in application use around the world. The application testing list is based on a combination of market data where it is available, individual knowledge of markets, culture, revenue, usage and even sometimes just “word on the street”. The cultural knowledge in these markets is probably most critical to our success. For example, casual gaming in Korea is hugely popular and we need to ensure our Windows 7 testing accounts for this.

Our goal in selecting applications is to test as many applications as we can that will expose the most issues across different scenarios and markets.

These scenarios include:

- Productivity
- Memories (photo editing and sharing apps, etc.)
- Graphics
- Productivity
- Music
- Fundamentals (security, data backup, etc.)
- TV/Movies

## **Application Acquisition**

Once we build the list of applications we need to test the next process is acquiring them. We acquire applications in a variety of ways but many times we have to buy an application from a retail store just as any end user would. Other methods we use to acquire applications include downloading full featured trial versions, purchasing software, and working with ISVs to acquire their applications to ensure compatibility.

## **Testing Applications**

Testing applications means more than just installing them and making sure they launch. Every application gets a unique test plan written for it to cover as much functionality as we can. We write test cases to cover primary and secondary application functions – for our word processing example this would include opening a file, typing a letter, adjusting formatting, save, and print, emailing a copy to someone, etc. These applications go through 6 or more test passes during the product cycle.

Now, we can't test every piece of every application and we do run into some interesting challenges when we focus on a worldwide audience. Many applications depend on location specific information (meaning if you aren't testing the application in that location – you aren't likely to have the information needed). Examples include Brazilian citizen's CPF ID, or Brazilian personal number of identification which would be required to test something like tax preparation software. We run into similar problems with SMS applications requiring active local mobile phone accounts.

## **What it means to Rescue an Application**



Along with the core tenet of ensuring that any application that worked on Windows Vista also work on Windows 7 we have a stretch goal to “raise the bar” and make applications work on Windows 7 that never worked on Windows Vista. For Windows 7, we have some good news early in the development cycle. So far we have made over 30 applications that were “broken” on Vista work on Windows 7. This means that Windows 7 will have higher application compatibility than Windows Vista. We are continuing to push this number up. Below is a table of the # of applications by language that we have made to work on Windows 7 but didn’t work on Vista.

Language	Number of Apps Fixed	Example Applications
Arabic	1	Khalifa Cartoon Characters Creator
Chinese (Simplified)	1	Arcsoft WebCam Companion
Chinese (Traditional)	3	<p>Asure Purchase/Sale/Stock Master 2008</p> <p>Cyberlink DVD Suite v6</p> <p>Asure Accounting Master 2008</p>
Czech	1	J.K.R. BYZNYS
Danish	1	Bogskabet 3.2
German	2	<p>QuickTime 7.1.6</p> <p>Haufe Personal Office Professional - Haufe Formular-Manager</p>
Hebrew	3	<p>Compedia Timmy in English World</p> <p>Compedia Moomins: The Search for the Ruby</p> <p>Compedia The Puzzling Time Quest</p>
Hungarian	1	Infocentrum Road Register

Italian	5	Finson Costo del Lavoro Italian v2 Finson Falco 6 Finson Progetto Condominio Finson Contintasca 7 Finson ContinBanca
Japanese	5	PostPet v3 Kenchako Adventure 9.0 WZ Editor 5.0 QuickTime 7.1.6 Overland LOKI: with Japanese Manual
Norwegian	1	Visma Avendo Fakturering
Polish	2	WF-Fakturka dla Windows Nahlik eTeacher 5
Portugese	1	Mr. Escola Win Port
Spanish	3	Mexico Federal Taxes Simplified SAT: Individual Taxes Monografias Spanglish IKEA Home Kitchen Planner
Turkish	1	MYTR Filter 2.6

Along with ensuring these applications work on Windows 7 we have taken an extra step for our existing Vista customers. Of the applications outlined in the above table, 27 of the fixes we made have been back ported to Windows Vista for possible inclusion in future updates. We really wanted to raise the bar for application compatibility and go beyond just looking at Vista as the baseline.

## **Takeaway**

There is a lot of information here and hopefully gives you some insight into what it means for us to make the application experience (application compatibility) on Windows 7 as high as possible for users around the world. We started out with a goal of making sure if an application worked on Windows Vista it should work on Windows 7. We have taken that further by bringing applications that never worked on Vista to work on Windows 7 and even future updates to Vista.

# A few more changes from Beta to RC...

Steven Sinofsky | [2009-03-13T03:00:00+00:00](#)

*Hey folks, just wanted to provide another update (building on the recent post on [some changes since Beta](#)) on some of the changes you will see in the Release Candidate. Again, there are many and this is not an exhaustive list. Of course we continue to gather telemetry from the large number of people running the Beta full time. Just a reminder, the Beta is the only official build from Microsoft. Chaitanya compiled this list from a broad set of feature teams focused on visible changes based on feedback that go beyond “bug fixes”, though we included some of the more widely reported bugs on this list as well. –Steven*

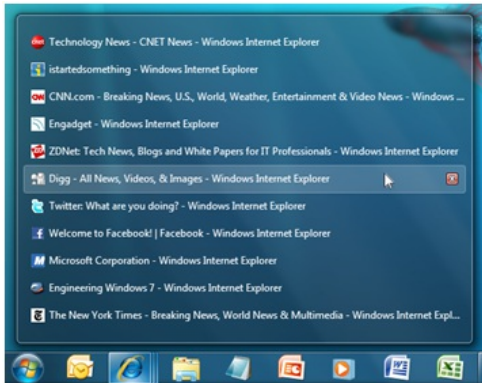
## Desktop Experience

### 1. Improved taskbar thumbnail overflow

Our customers are enjoying how windows are grouped and revealed on the enhanced taskbar. Some enthusiasts who have a significant number of open windows for a program encounter our scaling mechanism; the thumbnail view turns into a list view. Although this UI is virtually identical to experience in XP and Vista, customers still want to enjoy new functionality of the thumbnail view. Bentronic wrote, “It’s nice that there’s a little close button on the thumbnail previews--why not have a similar button for when it’s showing as a list? Being able to run down the list clicking the close button instead of right-clicking would be great.” For RC we’ve made the list view architecturally the same as the thumbnail view, just sans thumbnails. Customers will now enjoy close buttons and the menus open on hover (in Beta one had to click to open them).

Fig 1.

*List View of running windows appears on hover and supports close*

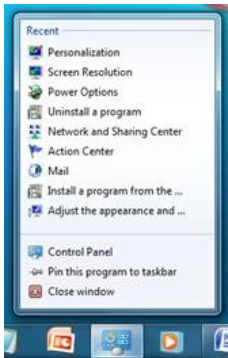


### 2. Control Panel Jump List

Right-clicking on the Control Panel icon on the taskbar in Beta revealed a noticeably sparse Jump List. A few people such as Britney told us “Should most recently used items be displayed in the Jump List of the CPL when pinned to the taskbar? Something should be shown and nothing is there right now”. In RC the Control Panel Jump List offers quick access to recently used items.

Fig 2.

*The Control Panel Jump List now surfaces recently used items*



## *2. PowerShell Jump List*

By default PowerShell in Beta launched a streamlined console. Customers could load optional modules via distinct shortcuts in the Start Menu. We heard from you that this was a confusing experience. Additionally, PowerShell did not surface a way to launch related tasks such as the Integrated Scripting Environment (ISE) from within their console experience. PowerShell now has a robust Jump List that affords a method to load modules, launch the ISE and open documentation.

## *3. Remote Desktop Jump List*

Rajeev made us smile with his comment, “Being able to add my Remote Desktop shortcut to the taskbar—good. Saving settings and showing them in the Recent items section—awesome. Being able to pin the connections in the Jump List, so they always appear—priceless!” Well, Rajeev and others who shared this request, you will be enjoy this functionality in RC.

## *4. Applying taskbar settings*

Have you ever customized the taskbar, only to find your changes were not saved across sessions? Has the taskbar ever inexplicably moved on you after you log in? For a variety of reasons, previous versions of Windows saved taskbar settings only after Explorer exited at the end of a session. However, if the OS is not shutdown properly these settings did not persist. Based on the bugs we saw from Beta, we decided to change our architecture and write these settings within 30 seconds (providing enough time to batch a group of changes) during the session. This means settings will now be more reliable.

## **Touch**

### *5. Multi-touch zoom*

One of the pieces of feedback we heard from the Beta was that customers enjoy the new multi-touch zoom feature, but wish it was supported in Windows Explorer. In response to this feedback we have added support for the zoom gesture in Windows Explorer. Using the zoom gesture you

can switch between view modes in Explorer such as zooming from Small Icons to Extra Large icons.

## Windows Explorer and Libraries

### 6. Invert Selection

In an effort to make improvements to performance, network bandwidth and memory footprint for various scenarios (e.g. libraries, search and search federation), we rearchitected the implementation of the view code in Windows Explorer. As part of this we did not to port “Invert Selection” since this rarely used feature is pretty complex to implement in the context of virtualized lists. Despite the small percentage of usage we’ve recorded, those who missed it have been pretty vocal?? On one of the blog posts, GGreig summarized what we heard from several of you —“Invert Selection; that’s a useful - sometimes absolutely invaluable - little piece of functionality, and I definitely don’t want to see it go...Please reinstate Invert Selection.” Given the feedback from enthusiasts, we added back the functionality for RC.

### 7. Going up?

We’ve heard feedback, especially from those on this blog, that in Windows 7 moving up in the folder hierarchy often requires multiple clicks since longer folder names in the address bar often bump the parent folder into the overflow dropdown.

For RC, we’ve improved the overflow algorithm so that the parent folder’s button will appear in the address bar at all times and therefore going ‘up’ will always be a single click away in a predictable location. When there isn’t enough room to display the parent folder’s full name, it will appear truncated instead of going into the overflow. If space is especially tight, then the current folder’s name may appear truncated too, but in all cases the parent folder’s button will remain as a click target in the address bar.

In addition to making the address bar an even better tool for navigating ‘up’ in Explorer, this change also makes it easier to tell where you are as you navigate around since you can now see at least part of the parent folder’s name. It also avoids introducing any more redundant buttons to the Explorer frame and hence taking away any more screen space from being able to see your address. Also, it goes without saying that if you navigate into a folder, you can still use the back button to go back up. And the keyboard shortcut is also available.

Fig 3.

*In Beta, a parent folder would collapse into an overflow dropdown*

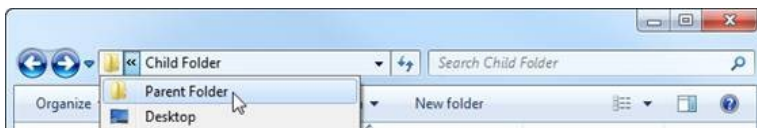


Fig 4.

*In RC, parent folders always remain within single click access*



#### *8. Finding music by artist*

We covered several of the improvements to arrangement views in the last post, but one we did not mention is that the “Artist” view in the Music Library now accounts for album artists and compilation albums. ShadowChaser summarized some feedback we heard from a number of customers in a comment: “The only concern I still have is with the ‘Artist’ view... it groups by ‘Contributing Artist’, not ‘Album Artist.’” Grouping only by contributing artist results in too many artists showing up and tracks from the same album getting split up in cases where customers didn’t expect. In RC, the “Artist” view in the Music Library groups together multiple tracks from an album by the common “Album Artist” property when it is available, groups tracks from compilation albums together into a “Various Artists” group and finally resorts to grouping by “Contributing Artist”. This reduces clutter when browsing music collection by artist, in addition to improving consistency with artist views in other applications and devices.

#### *9. New folder is always available*

We’ve gotten a lot of positive feedback during Beta about adding a top level “New folder” button in Explorer, freeing customers from digging into submenus. A common complaint we received, however, was that the button only appeared when nothing is selected. For RC, we’ve changed this so the “New folder” button will always appear, regardless of selection.

#### *10. Right-click in Windows Explorer*

For RC we’ve changed the behavior when right-clicking items in the view to address concerns customers were reporting with the Beta. We heard feedback that it was too hard to find space and get to the view’s background context menu for items such as New and Paste. Previously if one right-clicked over any portion of an item she would get the item’s context menu. We now show the view’s context menu when one clicks on any large white space, including the space between a file’s name and its properties.

#### *11. Content view for search results*

For RC we’ve adjusted the behavior when right-clicking items in the view to address concerns customers were reporting with the Beta. We heard feedback that it was too hard to find space

Content view is the new view mode we’ve added to Windows Explorer for Windows 7. It’s especially useful for search results where it surfaces the most relevant properties for each kind of file (e.g. documents, email, pictures and music) as well as a contextual “snippets” of the file content where the search term match occurred. There are a few changes here in the RC build. One thing we heard feedback on is that customers want to know exactly which properties were being shown for each item, so all properties now appear with labels. The text layout and colors have been updated in response to feedback to make each item even easier to parse and to avoid confusion with the colors used for encrypted or compressed files. We heard loud and clear that many found snippets very useful and wanted to see more of them, so in the RC we’ve allowed longer snippets and we’re using them in more places. In response to feedback we heard from customers when resizing their Explorer window or toggling the preview pane, we’ve made the transitions smoother as additional columns of information about each item are revealed when you make the view larger.

#### *12. Intelligent re-indexing after application installation*



In RC the Windows Search service now keeps the index up-to-date whenever support for new file types are introduced to the system. We know that in the past customers have sometimes had difficulties searching for files on their computer after new file handlers are installed. (File handlers govern how content and metadata is made searchable and are typically installed with applications such as Microsoft Office or updates such as the Microsoft Filter Pack).

In Win7 Beta (and previous versions of Windows), customers were required to rebuild their index whenever a new file handler was installed to ensure that any existing files were indexed with the newest functionality. Few customers knew to do this and it was an unnecessarily time consuming operation. Windows Search is more efficient in RC by automatically re-indexing the specific files affected by new file handlers. Rest assured that when one installs support for a certain kind of file, she can search for those files without doing any additional work.

## **Performance**

### *13. Trimming sound schemes to help performance*

We know our customers care about performance. We discovered that by just trimming the shutdown and logoff WAV files, we could save up to 400 ms. Every little bit counts.

## **Device Stage**

### *14. Baseline Device Stage experience*

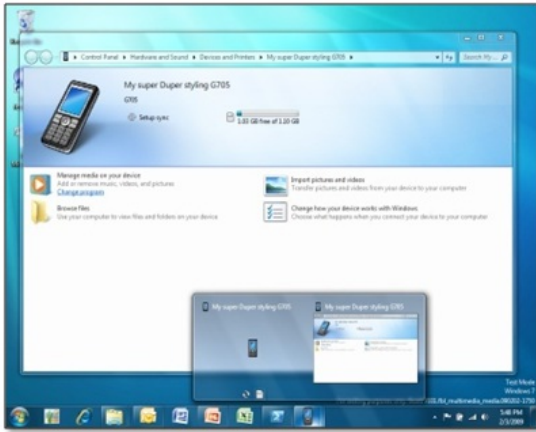
Device Stage continues to enjoy positive reviews. For example, we saw this post on a blog: "I have to be honest this works very well, it worked with my MP3 player in showing how much charge it had and other details as well is able to display the manual and offer me everything I needed to do with it effortlessly, including having the correct icon and image of the product." However, we occasionally hear "too bad , my N70 aint supported either ?? ...hopefully they are gonna support a ton more device by the time windows 7 get released".

We took feedback like this to the devices makers and they too would like more integration given the interest from our customers. Several manufacturers are implementing custom experiences, but a large number have also opted to support their older devices in what we call the "baseline" Device Stage experience.

This UX works exactly like full Device Stage; the device image appears on the taskbar whenever it is connected and tasks are exposed in the Jump List. On first connect, the shell Window containing all of the built-in tasks appears automatically and is always just one click away from the desktop icon or device image in the Devices and Printers folder. When the device maker implements a custom Device Stage experience for a device, it gets posted on the Web and the baseline experience gets upgraded when the device is later reconnected. The core functionality is the same, but all of the branding, imaging and vendor-specific tasks are now available automatically in the same convenient UI.

Fig 5.

*Baseline Device Stage experience for a mobile phone*



### 15. Devices and Printers enhancement

PC and laptop makers such as Lenovo, were very interested in doing more than just showing the machine's icon in Devices and Printers. They told us they wanted to leverage Device Stage to help them better customize the experience for our mutual customers. In RC double-clicking on the PC icon now offers a Device Stage UX. Like the other Device Stage devices, Device Stage for PC will be enabled when the PC maker has chosen to participate with their system.

Fig 6.

### Device Stage experience for a PC



## Devices and Printers

### 16. Unified experience for removing devices

One of the tasks customers perform in Devices & Printers is removing devices that are no longer in use. We received feedback that the remove action varied across different device classes. For example, removing a printer only removed the print queue and for Bluetooth devices it only removed the pairing of the device to the PC. We have changed this action to always completely uninstall the device across all device classes – which is the action that most customers expect.

### 17. *Hardware properties*

We know enthusiasts use the Device Manager's property page to check the status of a device. We heard feedback that this wasn't convenient and so we now also surface the property page directly from the Devices and Printers experience. Simply right-click on the device and one has one less reason to visit Device Manager.

### 18. *Improved eject experience*

The Safely Remove hardware functionality enables customers to make sure that their device is ready for removal. During the Windows 7 Beta, customers still had the Safely Remove Hardware functionality available on the taskbar as well as an Eject option on the context menus of applicable devices in Devices and Printers. Based on feedback, we have integrated these two separate pieces of functionality in RC and have changed its name from "Safely Remove" to "Eject". The tool Notification Area icon still appears, but its context menu now has the option to open Devices and Printers. Also, we have simplified the options by eliminating the drop-down submenu and made the semantics for eject functionality more consistent across the different kinds of media. For example, ejecting an optical drive now ejects the media instead of the drive and ejecting a USB flash drive ejects the entire device instead of an individual volume.

### 19. *USB device reliability on resume*

We got feedback from a number of customers that their USB devices (e.g. keyboards, mice and drives) stopped working after a suspend/resume cycle. We worked with a number of customers to get traces and isolated the causes to address them post-beta builds. The work around in Beta was to unplug and replug the device to get it functional again—easy for external devices but not possible for internal devices. This workaround will not be needed on RC builds.

### 20. *FireWire camera support*

Some customers informed us they were unable to connect their 1394 HDV camera and stream its contents to their Beta machine. With the help of customers, we were able to identify a fault with our core 1394 stack and we've validated the scenario works in RC. This is another good example of the combination of telemetry and more "manual" follow up on the part of our test team.

## **Device Installation**

### 21. *Add Legacy Hardware functionality restored*

The Add Legacy Hardware action was provided in Device Manager on past Windows releases to install non-Plug and Play devices. We removed this functionality for Windows 7 with the belief that this was rarely used. Aaron blogged, "You might have noticed that the old 'Add Legacy Hardware' option seems to be missing. I tend to use this quite a bit whenever I need to add in a Loopback adapter or some piece of hardware that is not quite installing correctly." As a result, this functionality has been restored to Device Manager for RC to help add non-Plug and Play devices.

### 22. *Increased responsiveness of Add Printer Wizard*

There are some situations with legacy network printers in which Plug and Play cannot automatically identify the appropriate driver even when it's available on Windows Update. For these situations, the Add Printer wizard allows customers to download a list of all the printer drivers available on Windows Update so they can manually select the driver for the specific printer being installed. The process of retrieving the list can take a few

minutes and we received beta feedback that many people felt their machine was hung since there was nothing in the UI to let them know that it could take a few minutes. We have made some UI changes to indicate that process of retrieval can take some time. Additionally, we have also improved the overall performance of retrieving the list from Windows Update.

## **System**

### *23. Partition size reduction*

In Windows Vista, configuring features such as Windows Recovery Environment and Bit Locker required significant customer interaction. Also, a significant amount of drive space was reserved. The Windows 7 System partition enables features to be configured to work “out of the box” so very little customer interaction is needed to configure and utilize them. Based on feedback and telemetry data received through the beta, it became clear that we could cut the drive size in half (from 200M to 100M).

### *24. Reserved System Partition naming*

The system partition is created automatically by Setup when installing on a machine with no existing partitions. During the Beta the existence of this partition on default installs confused many people and feedback indicated that a label telling them that this is space reserved for the system would be helpful when browsing disk configurations, and further help prevent it’s accidental deletion by enthusiasts. We will now label is “System Reserved”.

### *25. Dual Boot partition drive letter assignment*

For a dual boot configuration for the Beta, the other Windows OS wouldn’t get a drive letter and therefore wouldn’t show up in explorer. We heard overwhelmingly from Beta customers that the lack of a drive letter was confusing and even caused some to believe that their secondary OS was lost. Assigning the drive letter makes it visible in explorer and aids in navigation across OS installations.

### *26. Pagefile reduction*

Through extensive use of Beta telemetry data, we have determined we can slim down the Windows disk footprint further by reducing the default page file size to be 100% of the available main memory. It used to be “Memory + 300MB” so on a 1GB RAM system there was an extra third allocated that is no longer required. The pagefile on some occasions will increase in size if required, but we just pre-allocate less.

## **Network**

### *27. Improved driver support*

Based on telemetry data received from the beta, we identified networking drivers that were not available inbox. We worked with ecosystem partners to achieve increased inbox driver coverage across wireless and wired with significant coverage for some of the new ATOM-based laptops.

We hope you enjoyed yet another sneak peek into what's coming in RC.

# Designing Aero Snap

Steven Sinofsky | [2009-03-17T03:00:00+00:00](#)

---

*Here's a behind the scenes look at the design of the Aero Snap feature in Windows 7. We thought it would be fun to take a look at the overall design process of the feature and the tools and techniques used. This feature poses a unique design challenge in that you just use the feature without any user-interface specifically to invoke it. As with all features this is a collaboration across all of our engineering disciplines. For this post, Stephan Hoefnagels, a Senior UX designer, presents the design perspective. – Steven (P.S., keep an eye out on the Microsoft MIX conference this week!)*

In [Managing Windows windows](#) and [Follow-up: Managing Windows windows](#) we talked about, and you shared, some interesting window management scenarios that we might address in Windows 7. We also touched on some data around typical configurations, as well as goals that guide our thinking in this area.

In this post we'd like to have a closer look at the Aero Snap feature that many of you have already been able to experience in our PDC builds, and of course the Beta. We'll briefly describe the feature itself, but mostly we'd like to invite you to take a behind-the-scenes peek at our design process so far, and share our iterations, challenges and considerations.

## Goals and scenarios

As we explained more in depth in our [previous post](#), our top goal for the Aero Snap feature is to provide you with an effortless way to position your windows the way you want them. We want to reduce the number of clicks and precise movements needed to perform common activities. In a general sense, we want you to be able to *manage your windows with confidence* and create a feeling of power and control. This is something we touched on in our post on the [Windows 7 taskbar](#) as well, and really a theme that weaves through much of our new desktop experience.

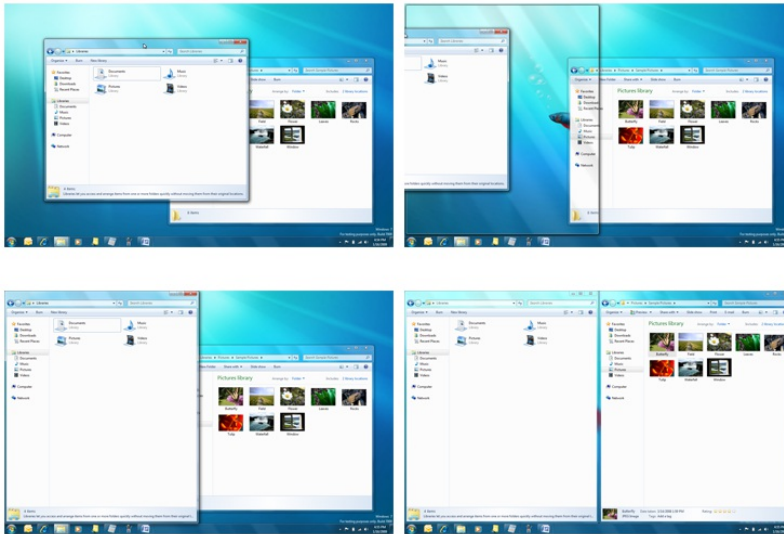
Before we look at how we address our goals in the design, a quick note. In the scenarios below you'll notice sequences of interactions that are fully written out. Sequences like: select the window, click the caption button, and then resize the window. Looking at interactions at this level of detail makes for somewhat awkward reading. These individual steps are so small, and so frequent, that for most of us they normally barely register. Why not simply gloss over some of those details? Well, we spell out sequences of events this way to force us to be consciously aware of the amount of "overhead" that is sometimes involved to get to the task at hand. It forces us to realize what we normally might not. Also, besides providing insight in the problems, it provides us with the right level of detail to consider our solutions.

Now, let's look at the design!

## Side-by-side windows

As many of you mentioned, doing a drag drop operation from one window to another is sometimes a pain. Windows tend to overlap and to get them positioned right can require a lot of fine mouse movement. Oftentimes the steps are as follows: select a window, resize it appropriately, and position it on the screen so that enough of it shows for it to be a meaningful drag or drop target. Then repeat the same actions with the other window. Similarly, comparing content in two windows requires a lot of mouse clicks, resize actions, careful arrangement, window switches, and a fair amount of mouse mileage.

With Aero Snap you can grab a window and move your mouse to the edge of the screen and the window will resize to fill half the screen. Repeat with the other window. Now with two easy motions you have a setup that makes both of these scenarios much easier to accomplish!



*Put windows side-by-side with Aero Snap by moving the cursor to the edge of the screen (left to right, top to bottom)*

### Vertical maximized windows

We know our users love the maximized window state. Many love it so much that they maximize all of their windows and never even run in any other state! However, with screens increasing in resolution and widescreen layout becoming more prevalent, the maximized window state can lose some of its appeal in certain cases. E-mail is an example. Reading long lines of text across the screen is not ideal. Your eye simply cannot track a line all the way across. Web browsing is another example. Content will sometimes not fill the entire width of the screen, leaving a lot of unused white space on the side.

Now, with Aero Snap you can maximize a window in the vertical direction only. When you resize a window to the top of the screen, it will also resize all the way to the bottom. Great for reading long blog posts!



*Vertically maximize a window with Aero Snap by resizing the window to the edge of the screen*

### Moving maximized windows from screen to screen

We realize there are a few multi-mon users out there, especially amongst the readers of this blog :-). Ever wanted an easier way to move a maximized window to another screen? A way that's quicker than clicking the restore caption button, moving the mouse to the title bar, dragging the window over, and finally clicking the maximized caption button again? With Aero Snap you can simply drag a maximized window down, move it over and snap it to the top, all in one gesture. Finally!

### Arranging windows on laptops

Arranging windows on a desktop PC can sometimes involve excessive fiddling, fine mouse movements and lots of mouse mileage, as touched on above. On laptops this situation is further exacerbated by the lack of a mouse, making some of these movements even more cumbersome. For these scenarios, and our power users, we've introduced some convenient key combinations. Hold down the Windows key and one of your arrow keys to give it a try. You may want to try holding down Shift as well, especially if you're in a multi-mon setup.

## Design process

OK - So those are the scenarios, and the way we chose to address them. Seems straightforward enough, right? Especially for those of you that have been using Aero Snap in our Beta build. It all behaves as you'd expect. But how did we get here? Well, in this next bit we'd like to something that we don't do that often and really give you a peek into our [design process](#) so far and some of the snags we ran into along the way. Keep in mind that this is a brief overview and by no means exhaustive. While we allude to usability testing for instance, this post is by no means meant to give insight into the scope of those efforts. There will certainly be opportunities to talk about some of those topics in the future.

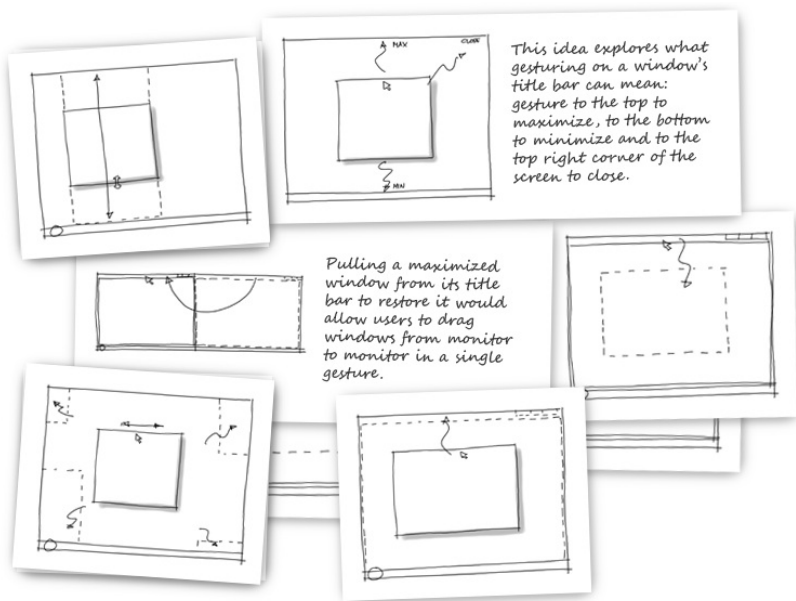
Let's have a look!

## Sketching

We're back in early 2007, and after we established window management as a potential area of interest, and identified several appealing scenarios (again, see [our previous post](#) for more detail on that stage of the process), we started with brainstorm to generate ideas. "How can we make window arranging more efficient?" "More direct?" "More fun?" are some of the questions we asked ourselves. This was a multidisciplinary process, with disciplines like design, user research, program management and development involved. All in all there were around a handful of us, certainly less than a dozen, thinking about this space at the time.

We imposed a significant constraint on ourselves: we wanted to achieve our goals *without* introducing any extra widgets in our UI. Imagine an extra caption button on the window title bar for instance: this may not seem too bad for one window, but when we're talking about 10 windows or more, we've all of a sudden introduced a significant amount of clutter on your screen. And that's something that we simply didn't feel good about. After all, as we shared in [our UX Principles talk at the PDC](#), we believe in "Solving Distractions, not Discoverability".

We captured our, many, ideas in very quick sketches that we shared via an internal website. Transferred from the whiteboards in our offices and hallways, these took less than 5 minutes to sketch each. The sketches below are some actual examples in which you can start some of the Aero Snap ideas forming.

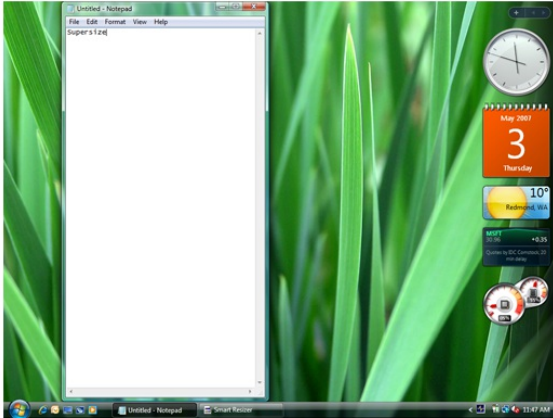


*Early ideas are captured in quick and disposable sketches*

## Interactive prototyping in early code

With so many ideas on paper we were eager to try out the best ones. Now was the time to prototype. Note that we are still very early in the process. We wanted our prototypes to be interactive, and we wanted to be able to live with them in our day-to-day work. So we chose to implement the ideas using early code that we could run on our work machines. For example, the image below shows a "smart resizer" prototype running on Windows Vista. Of course these prototypes are not "done" features that we could actually ship; they merely get the basic ideas working, and they definitely have more than a few "quirks" (bugs :-)). What's important however, is that they allow us to experience *just enough* of the interactions ourselves, as well as get feedback in usability studies.





Early "smart resizer" prototype running on Windows Vista, note the taskbar button created by the prototype (and the date in the calendar to get a sense of where we are in the process)

We use this firsthand experience and early usability feedback to iterate on the ideas as we hone in on the final design. We approach this part of the process in a very open minded way and allow ourselves to be surprised. Sometimes ideas that don't look like much on paper prove to be startlingly powerful. On the other hand, we found out that others look much better on paper than they were in practice! Since we're not invested in any one idea or implementation yet, we freely refine, and drop ideas.

Finally, the prototypes ease us into thinking about design details. And we might stumble on some insights too (of course we tell ourselves the real feat is to recognize the insight and hold on to it). Here's an example from an e-mail at the time from one of the team members about a new version of the "smart resizer" prototype:

*I noticed something changed. In the original version if I resized it to the max it "super-sized" the window. Then if I resized the window smaller, it jumped back to the normal restored state. It was as if the super-size state was different than restored state. With this version when I super-size the window and then resize it again later, it doesn't jump back to the previous restored size. There was something kind of nice about the super-sized state being different than the restored state. We should think about it more and consider making that a part of the design.*

After many a prototype we settled on the concept of side-by-side windows and vertical maximized windows. We're getting clearer on what we want to build.

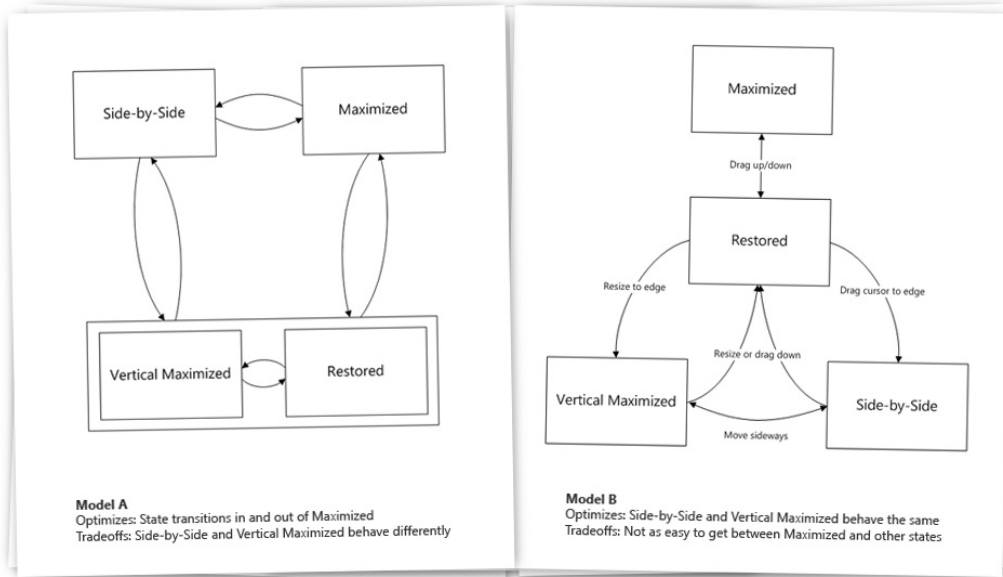
### Detailed design: state transitions

OK - We've whittled down our ideas to a good, but not fully specced, set of interactions and behaviors. Time to start filling in the blanks by asking detailed questions. "What does it mean to have a side-by-side window in a world where there used to be only minimized, restored and maximized windows?" "How exactly would you get to and from this new window state?" The e-mail snippet above already pointed to some of these questions.



Currently the common window states are (left to right) minimized, restored, and maximized, how would side-by-side and vertical maximized windows fit in?

Let's look at the state problem in detail. Below is an example of two proposals made during this time that show how you can move from one window state to another, for all the different states. Which model is better?



Two proposals detailing the various ways we can transition between states

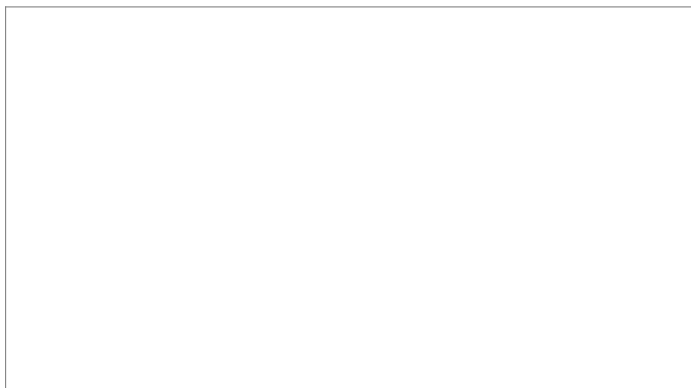
To answer that question we considered more specific questions like “What states do we want to link directly and how do we move between them?” “Is it compelling to go from a vertical maximized state directly to a maximized state?” “Should the vertical maximized and side-by-side states behave similar, as they look similar?” Our answers of course guided us to the model that you are now familiar with, which is model B.

But that’s not the end of it. More details need to be worked through, and more questions come up. “What if I want to move a vertical maximized window sideways?” “Resize its width?” “Then pull it down?”

Soon you’ll find yourself in elaborate sequences of events, many possible actions, and even more possible outcomes. Which would be the most expected when actually using the entire system?

To help guide our decision making process we established some guiding statements. Assumptions that we hold to be true. Examples are: “The intuitive way to undo an effect triggered by a mouse movement is to make the opposite mouse movement.” And: “It should always be effortless to go back to the previous “restored” state so as to avoid excessive work to get the window back to a reasonable size.” Or: “If the user specifies a width for a window in a given state, that size should be preserved across state changes when it makes sense.”

Using these statements we were able to answer questions like the ones above in a predictable way and as a result craft a predictable experience. And while the underlying state transitions and rules are fairly complex when added all together, the resulting behavior is, we hope, intuitively understood. That’s definitely something we’re aiming for.



## Conflicting rules

Here's an example of a sequence problem for the really attentive reader. When working through our many new state transitions, sometimes the rules that determine what should happen, conflict. Consider the following scenario. Two rules in our new window model are:

1. *Dragging a window to the top of the screen maximizes it*
2. *The intuitive way to undo/cancel an effect triggered by a mouse movement is to make the opposite mouse movement*

OK – Let's try the following scenario in your Windows 7 build. Start with a restored window. Vertically maximize it by resizing it to the top of the screen. Release the mouse. Drag (don't resize) down the window and drag it to the top again, all in one motion. Release the mouse. What happened?

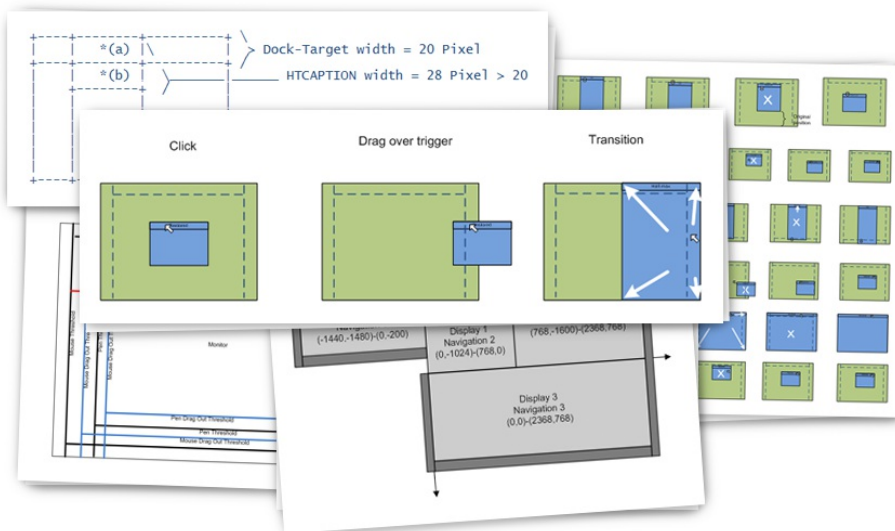
Your window should be maximized. Which means in this case we chose to follow rule 1. We could have also followed rule 2, in which case your window would be vertically maximized. We figured rule 1 would more accurately reflect the user's mental model for this scenario.

This is just one example of the decisions we had to make for each transition to and from a window state.

Throughout this process we made sure to preserve the subtleties in the current model. One small example that some of you may be familiar with that we *did* tweak is the scenario of dragging a window title bar off screen. In previous Windows releases we would snap the window back halfway in an attempt to provide you with just enough space to still move the window around, while optimizing vertical space as much as possible. We chose to be a little more deliberate and straightforward here by snapping back so that the entire title bar is visible. This way you can start to rely on the fact that all of your windows are always very easy to grab, also with touch, and move around. If we *really* felt half of a title bar is enough, why don't we always half it, right? For our vertical maximized window states we of course chose to keep the entire title bar visible as well, leading to a solid story all around.

## Storyboards and other visualizations

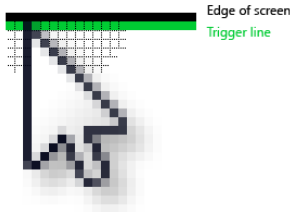
State diagrams are of course only one way to look at the world. We used various ways to communicate different aspects of the feature, picking our medium based on familiarity, availability and intent. We didn't even shy away from the occasional ASCII art as you can see below! We'll simply use whatever tool gets the point across. Most of all, interaction storyboards were a very valuable technique to help us understand the user flows, and even though this is only a small sample, you can see we did quite a few of those.



Feature details are communicated throughout using appropriate means

## Accidental triggers

Besides figuring out the right state transitions, one of our biggest discussions was around *when* a state transition should exactly occur, or in other words: when the feature is triggered. We talked a lot about “accidental triggers”, i.e. running into the feature without deliberately setting out to do so.



*Aero Snap* is triggered by touching the edge of the screen with the mouse

From the very beginning we always made sure that our feature did not get in the way of current scenarios such as tucking a window off-screen to the side. After all, we don't want to have a detrimental effect on your current, expected, way of managing your windows. That's why you have to literally touch the edge of the screen with your mouse, not the window edge, to trigger the transitions.

However, at this time, the feature as you know it now was different in one very important, fundamental way. In our early builds, *Aero Snap* followed an “instant commit” model. When you moved your mouse to the top of the screen, your window would literally be maximized *while you're still dragging*. That is, before you even release the mouse. Moving back in one motion would literally restore the window. We liked this approach as the “preview” was very accurate, i.e. the preview *was* the window, and there is a certain directness in not having a commit model.

Because our UI is invisible by design, we expected some accidental triggering. In fact in some sense we were relying on accidental triggers to help with the discoverability of the feature. However, after living with the builds for a while we got a little worried because accidental triggering seemed higher than we expected. From our telemetry data we saw people running into the feature, and then cancelling it nearly twice as often as committing. In our usability studies we observed confusion as to what exactly triggered the behavior. Was it the window touching the edge that did this? A gesture? Or something else?

We're now in early 2008. What should we do? Cut the entire feature? We actually did consider this as an option. Again, we really respect our current window management behaviors, and the last thing we wanted to do is degrade the experience. More constructively, we took on the challenge to come up with a design tweak that would address these issues. We explored several solutions. Some conservatively centered on smoothing out the resize transitions, so an accidental trigger would at least be smooth. A conservative approach for sure, but probably not satisfying enough as a real solution. Others focused on trying to detect user intent more accurately, based on the angle of motion into the screen edge, or the speed of the motion. This proved much too complex to be predictable. Maybe we could trigger the transition only when double-bumping the edge? We were worried that this would degrade the experience of fluidity and flow of the current model.

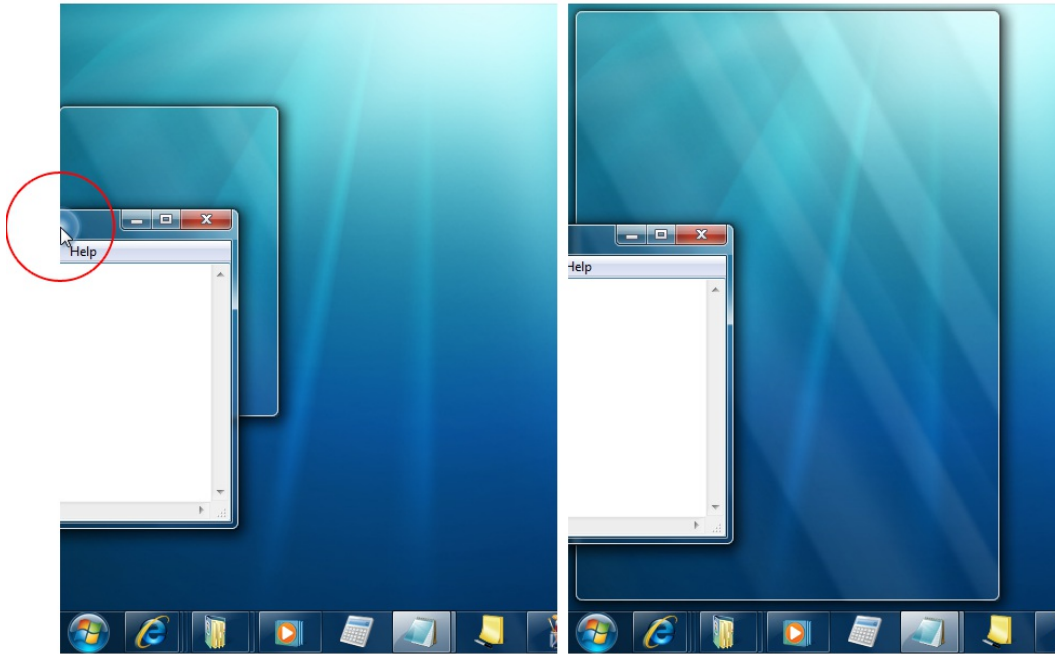
In the end we settled on the implementation you're familiar with. We don't trigger until you *release* the mouse, making it easy to back out of the effect before anything happens to your window. In addition we provide you with a smooth preview animation, and a cursor effect to help you understand what just happened. This way you can be more deliberate in the future, and use the feature to your full advantage.

Did we solve the issues? Feel free to let us know ??

## Look and feel

It's interesting to think back and realize that up to this point we had essentially designed a feature without any visible UI whatsoever. Now all of a sudden we have window previews, and a cursor effect. What should those look like, and how should they behave?

Well, luckily we had some things to go on. At this point we had already established a clear picture of our taskbar look and feel (we call it “personality” and will talk about it more in a later post). We had settled on using glass sheets for *Aero Peek*. We saw an opportunity to use the same visual representation for our preview windows. But how should the glass sheets appear? After experimentation, we settled on a scale animation that originates from the cursor. This gives a subtle hint as to where this preview window came from. We also made sure to animate our transitions. Try this in your build for instance: move a window to the top, and then to the side, in one motion without releasing the mouse. Notice the smooth morph of the preview? Why did we spend time on this? We believe “[Small Things Matter, Good and Bad](#)” of course.



*Light effects are used to indicate the snap trigger, and glass sheets for snap previews*

Finally, we tied into some of our ideas around “light” for the trigger indication. We tuned this to be noticeable, but not too loud and made sure to synch with other light effects in the system such as our touch feedback.

We hope this post has given you some insight in the Aero Snap feature, including our design process. We would love to hear your thoughts!

*Stephan Hoefnagels*

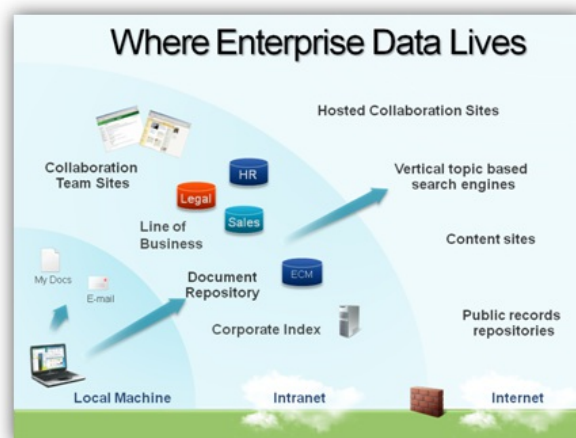
# Federating Windows Search with Enterprise Data Sources

Steven Sinofsky | [2009-03-23T03:00:00+00:00](#)

*The Windows Explorer has evolved by enabling you to find all sorts of content by searching for it. Many of you have used the search features in Windows Vista (based on our instrumented data) from the start menu or from the search box in Explorer. It has been a long time since most of us could remember where everything is by carefully managing our folder hierarchy and finding things based on file name alone. We often rely on domain specific search (in music players, mail clients, photo clients) but with Windows Vista and Windows 7 we make it possible to search within a namespace and across namespaces. This post is about a new feature based on Search that allows searching across PCs and even servers in an Enterprise setting. Alwin and Scott, program managers, and Brandon, a developer, on the "Find and Organize" feature team authored this post. --Steven*

## Finding your stuff

Whether you're searching or browsing, Windows Explorer is really about *finding your stuff*, and once you've found it, doing something with it (such as copying, opening, deleting, etc). For data that lives on your PC or home network, Windows 7 has invested in HomeGroup and Libraries (subjects for a future posting from our team) to provide an easier and richer experience than ever before. However, we didn't stop there. Over the last few years, we've seen enterprise customers' important content migrate towards (or aggregated in) centralized content stores, such as SharePoint. These products typically provide great features for team collaboration, document versioning and workflow management, archiving, retention policy enforcement, and other centrally-managed functionality that IT managers appreciate.



*Important enterprise data is found on local machines, in a variety of centralized content stores and also beyond the firewall*

Unfortunately, this has placed an extra burden on customers to learn each new content store's user interface, often asking them to give up familiar desktop features like drag-and-drop. Given their collaborative focus, these sites grow organically and it can become hard to remember where a particular document was stored and then wade through long lists of them every time you want to get back to it. Enterprise customers have asked us for a solution that simplifies finding important content in these various data stores but without leaving their normal Windows work flows.

As we looked at this trend and the lack of integration with content management and content indexing web services, we used these guiding principles in developing a solution:

- **Natural for people to use.** Customers want a more consistent experience for finding and working with data in these disparate content

stores, and would like us to bridge the local and remote content experiences by helping them “roll over” from one to the other.

- **Easy for IT admins to deploy.** IT admins don’t like to deploy code, and want low-maintenance solutions that are easy to manage. Meanwhile customers want to connect up these sources without going through long and tedious installation processes or having to get help every time they want to set up a new search location.
- **Easy for developers to adopt.** Developers want to enable this functionality in their offerings quickly and easily. There are a lot of data sources which need to be supported because IT folks don’t want to be locked in to a specific server technology.

## Choosing to build Federated Search

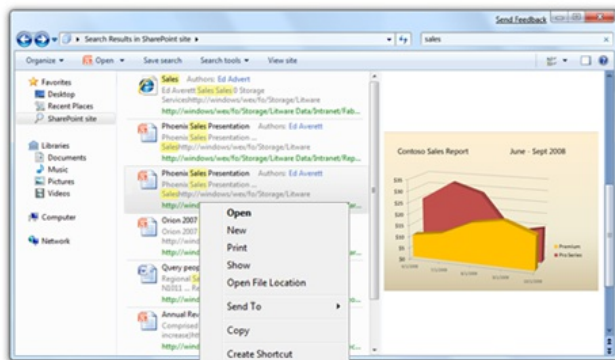
Federated Search wasn’t the only way to address these challenges. The brute force approach would have been to take our existing [Windows Search](#) indexing technology and just use it on these content stores—that index the remote content on a local PC. This isn’t a very realistic solution since it’s inefficient to have all content indexed over the network by each person’s machine, especially when the content is changing at a rapid pace and represents a large corpus. Corporate retention policies may also prevent keeping even a local index of certain sensitive data.

Fortunately, there’s a better option – Federated Search. Federated Search enables you to search a remote web service from Windows explorer and get results back that you can act on like any normal file. The largest barrier to doing Federated Search has already been taken care of. That is, most of these content stores are already indexed *on the server*, or at least on *some* server. There are several great offerings that will accomplish this, such as Microsoft Search Server. Not only do these servers index this content, but many of them already expose search results via a standard web protocol. This is largely thanks to the prevalence of OpenSearch and RSS enabled clients (including Internet Explorer and Microsoft Search Server, among many others).

For Windows 7, we’ve added support for Federated Search using [OpenSearch v1.1](#) and worked to make the experience a seamless one. We found this solution strikes a good balance by leveraging the strengths of content services and the strengths of local file interactions within Windows.

## Natural to use

Using Windows Explorer, people are familiar with several important user interface and interaction elements. They know how to use the navigation pane to change what they’re looking at. They know how to scroll around, how to select an item (or several), and they know how to double-click to open them. Most people know how to right-click for context-sensitive options related to their selection, or how to find those options presented in the command bar. They know they can drag and drop items to move them around. They know how to change view modes. We hope that they know how to search their current location using the search box, and in Windows 7 we think we’ve made it much easier to discover and use the Preview Pane to make sure they’ve got the right result.



*Searching a SharePoint site using the new Federated Search support in Windows Explorer*

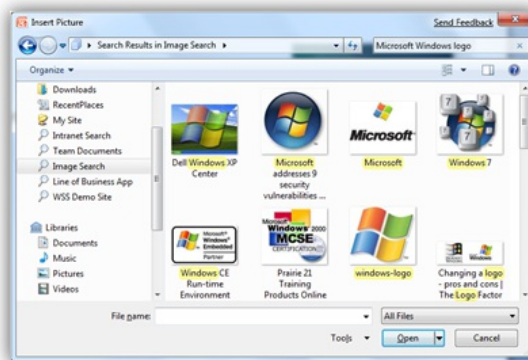
Much of the usefulness of building Federated Search into Explorer is our ability to take advantage of this knowledge and familiarity. This may seem obvious once you see it in action, but behind the scenes there's quite a lot going on to make all of this happen. For example, some applications such as Microsoft Word already know how to work with web URLs. So opening a Word document from a web server is fairly straightforward. But the majority of applications you'll encounter really only understand how to open files on the local machine or via standard network file sharing protocols. This includes everything from the built-in software like Notepad and Paint, to third-party software like Photoshop or iTunes.

To handle this case, we implemented a "just in time" download solution, which will download the file to the internet cache before opening an application or taking actions (like using the SendTo menu) which require local files. This lets us offer searches that are very "lightweight" from a server load perspective, where we display metadata and icons or thumbnails without ever requesting the actual file. Then if you take an action like previewing or opening an item, we will do some behind-the-scenes work to make a local copy of the file *only if necessary*.

That enables us to work with the existing application ecosystem without asking anything of developers. However, applications can also take steps to offer even better functionality in many cases. For example, Windows Photo Viewer has added support for non-file items. So if you open a picture result in the built-in photo viewer, it's the photo viewer that downloads the item, not Explorer. This may not seem like a big deal, but it lets the photo viewer enable the forward and back buttons to jump to the next or previous result – and it will download that image on-demand. Starting at the PDC we began reaching out to third-party ISVs to encourage them to implement similar enhancements for Federated Search scenarios, and we will continue to offer guidance on how to best integrate with all of the newest Explorer features.

Finally, we support all the standard clipboard and drag-and-drop operations. So if you drag a Word document from a Federated Search query onto your desktop, it will be copied there. You'll even see the familiar Windows Explorer copy dialog, with progress indication, cancel ability, conflict resolution, and so on.

But wait, there's more! Windows Explorer is a great tool that many customers know and love. But some people use it without even knowing it. Countless Windows applications make use of what we call the Common File Dialog. This is a special Explorer window that lets you find and choose items to be opened or inserted into your current application, without ever leaving it. If you've ever clicked File and then Open or Save in an application menu, you've probably seen some version of this dialog. PowerPoint, for example, uses the common file dialog to insert pictures. That means from inside PowerPoint you can click Insert Picture, select the Federated Search link for your image repository, search for the picture you want, and then insert it directly into PowerPoint. This works for any existing application that supports the Common File Dialog, and there are a whole lot of them!



*Inserting a picture into PowerPoint's using Federated Search*

Our Federated Search solution is all about simple lightweight access with a common, familiar user interface. This has a lot of benefits as we described above, but there are also cases where a server's web interface will offer its own benefits. This might involve advanced query building, browsing, or server work-flow tasks, for example. So Windows 7 builds a bridge to these content repositories. After doing a search against a supported location, you will see a "Search on Website" button in the command bar which allows you to seamlessly send the query up to the



service's web interface in the default web browser. You'll also see the "Open File Location" menu item when you right-click on a search result. Selecting that option will launch the web browser to the specific location in the document repository where the file is stored.

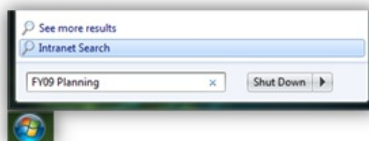
This seamless integration of Federated Search within Windows allows customers to greatly simplify their workflow for getting at remote files while still being able to easily take advantages of the advanced functionality of content repositories.

## Simple to deploy

Our next challenge was to make it easy for customers to get these new connections onto their machines. It wouldn't be practical to ship Windows with a connection to every solution in the world, so we shifted to a way that would make it very easy for any web service to deploy a connection to their specific service.

The model we came up with is similar to the way you add favorites from the web today. A web service can place a link to an .osdx file somewhere on their web page (see [Channel 9's search page](#) for an example). The .osdx file is a simple XML file that uses the [OpenSearch description document](#) format to describe how to connect to the web service, and gives the web service some control of how the data is presented in Windows Explorer. When a person clicks on the link, Windows performs an ultra-lightweight install process that adds a search connector to that web service and places a link to that in the Windows Explorer favorites.

If you are an administrator in an enterprise environment, you will likely want to provide some pre-installed search connectors for your users to search the company intranet or a popular internal SharePoint site for example. You can do this by deploying the search connector (.searchconnector-ms) files to your users' machines via typical deployment techniques such as imaging, group policy preferences or startup scripts. The beauty is that it's just a simple XML configuration file and there's no code that needs to get installed on their machines. It's also possible to pin one of these as a link from the Start menu through group policy. In the group policy editor look for the policy in this area: User Configuration > Administrative Templates > Windows Components > Windows Explorer. The policy name is "Pin Libraries or Search connectors to Search again links and start menu".



*Launching a Federated Search of an enterprise Intranet from the Start Menu*

## Easy to adopt

Of course this technology depends on having services that support it. Although there are only a few services that provide a .osdx for you today, there are many existing services that already support the basic OpenSearch requirements.

We're already seeing positive initial reactions from enthusiasts and ISVs alike echoing that it is indeed easy to enable your service to work with our Federated Search platform. If you're a developer and want to enable an existing web based service to support Windows 7 Federated Search, you'll need to provide a web service that accepts an http **GET** request with the search terms embedded somewhere in the URL and be able to return the results as an RSS or Atom feed. These requirements are typically very easy to meet for most applications that already provide search services via a web browser.

Your web service results should include the basic RSS tags like <link>, <title>, <description>, <pubDate> to get started but there's much more that you can include in the RSS output and customization you can do within the .osdx file to enhance the experience for the end user.

For more information, we've published the [Windows 7 Federated Search implementer's guide](#) with detailed information on how to enable your data source to work with Windows Federated Search. There's also a [recorded PDC session](#) that demonstrates how to build a Windows Federated Search compatible web service for an existing SQL database.

- Brandon Paddock, Scott Dart & Alwin Vyhmeister, Find and Organize

# Touching Windows 7

Steven Sinofsky | [2009-03-25T03:00:00+00:00](#)

---

*We've come a long way in engineering Windows 7 since we first provided an engineering preview of Windows 7 and the work we are doing to support the touch interface paradigm back at the [D: All Things Digital conference](#). We chose to kick-off the discussion about engineering Windows 7 with touch scenarios because we know this is a long-lead effort that requires work across the full ecosystem to fully realize the benefit. For Windows 7, touch support is engineered by building on our advances in input technology we began with the TabletPC work on Windows XP. Touch in Windows 7 requires improvements in hardware, driver software, core Windows user experience, and of course application support. By having this support in an open platform, consumers and developers will benefit from a wide variety of choices in hardware, software, and different PC form factors. Quite a few folks have been a little skeptical of touch, often commenting about having fingerprints on their monitor or something along those lines. We think touch will become broadly available as the hardware evolves and while it might be the primary input for some form factors (such as a wall mounted display in a hospital, kiosk, or point of sale) it will also prove to richly augment many scenarios such as reading on a convertible laptop or a "kitchen PC". One of my favorite experiences recently was watching folks at a computer retailer experience one of the currently available all-in-one touch desktops and then moving to another all-in-one and continuing to interact with the screen—except the PC was not interacting back. The notion that you can touch a screen seems to be becoming second nature rather quickly. This post is our first dedicated blog on the subject. This is a joint effort by several people from the touch team, mostly Reed Townsend, Dave Matthews, and Ian LeGrow. -Steven*

Windows Touch is designed to enhance how you interact with a PC. For those of us that have been living and breathing touch for the last two years we're excited to be able to deliver the capability to people using Windows 7. In this blog we're going to talk about what we've done to make Windows touchable. We approached this from a number of different directions: key improvements to the core Windows UI, optimizing for touch in key experiences, working with hardware partners to provide robust and reliable touch PCs, and providing a multitouch platform for applications.

## Making Windows Touchable

With Windows 7 we have enriched the Windows experience with touch, making touch a first-class way to interact with your PC alongside the mouse and keyboard. We focused on common activities and refined them thoughtfully with touch in mind. You will have the freedom of direct interaction, like being able to reach out and slowly scroll a web page then flick quickly to move through it. With new touch optimized applications from creative software developers you will be able to immerse yourself as you explore your photos, browse the globe, or go after bad guys in your favorite games.

While providing this touchable experience, we made sure you are getting the full Windows 7 experience and not a sub-set just for touch. We've been asked if we are creating a new Touch UI, or "Touch Shell" for Windows – something like Media Center that completely replaces the UI of Windows with a version that is optimized for touch. As you can see from the beta, we are focused on bringing touch through the Windows experience and delivering optimized touch interface where appropriate. A touch shell for launching only touch-specific applications would not meet customers' needs – there would be too much switching between "touch" mode and Windows applications. Instead, we focused our efforts on augmenting the overall experience so that Windows works great with touch.

We took a variety of approaches – some broad, and some very targeted to support this goal:

- **Touch gestures:** Windows 7 has a simple set of touch gestures that work in many existing applications. These include the basics of tap and drag, as well as scroll, right-click, back, forward, zoom, and rotate. More details on how gestures work is described later.
- **Improved high DPI support:** Windows 7 has improved high dpi support (see [High DPI blog post](#)). The broad benefit to touch is that UI elements are rendered closer to their intended size – usually larger – which makes small buttons, links, and other targets easier to access with touch.
- **Improved window management:** The updated taskbar and windows arranging features go a long way towards making Windows easier to use with touch. There have been several subtle but critical touch optimizations:

- The taskbar buttons and thumbnails are ideally sized for pressing with touch, and specific behaviors are tuned for touch input. For example, the Jump Lists can be accessed with a simple drag up from the taskbar, and when opened with touch, the shortcuts in the Jump Lists are drawn with extra vertical spacing to make them easier to select.
- Aero Peek has been tuned to work with touch – the show desktop button is twice as wide (the only visual sign you are on a Windows Touch PC) and instead of hovering (which you can't do with touch), a press-and-hold on the button activates Aero Peek.
- Sizing and positioning windows is easy with Aero Snap – just drag a window to a screen edge. Furthermore, this was tuned with special touch thresholds so that you don't have to drag to the absolute edge of the screen – a better balance for touch usage.
- **Refinements to key experiences:** The top browsing and media activities were refined to provide an optimized touch experience. IE8 includes support for the core touch gestures (scrolling, back, forward, zoom) as well as an optimized address bar that opens by dragging down, and extra spacing in favorites and history lists when opened with touch for easy selection. In Windows Media Player, the transport controls (play, pause, etc) have larger clickable areas even though they still look the same size – so that they are easier to touch.
- **Touch keyboard:** The on-screen keyboard has been optimized for touch with glow key feedback that's visible when your finger is covering the letter and multitouch support for natural typing behavior and key combinations. It's designed for quick usage, like entering a URL.

Overall, the Windows Touch features are designed to work together to deliver a great end-to-end touch experience. For example, the goal with IE8 was to deliver a seamless touch browsing experience, this includes the panning, zooming, URL entry, and several interface enhancements. For this reason, all the new touch features require the presence of a multi-touch digitizer – more on that further down.

## Gestures

The Windows Touch gestures are the basic actions you use to interact with Windows or an application using touch. As we noted above, because the gestures are built into the core of Windows, they are designed to work with all applications, even ones that were never designed with touch in mind.

Our mantra with gestures has been “**Predictable + Reliable = Habits**”. To be *predictable* the action should relate to the result – if you drag content down, the content should move down. To be *reliable*, the gesture should do roughly the same action everywhere, and the gesture needs to be responsive and robust to reasonable variations. If these conditions are met then people are far more likely to develop *habits* and use gestures without consciously thinking about it.

We've intentionally focused on this small set of system-wide gestures in Win7. By keeping the set small we reduce misrecognition errors – making them more reliable. We reduce latencies since we need less data to identify gestures. It's also easier for all of us to remember a small set! The core gestures are:

- **Tap and Double-tap** – Touch and release to click. This is the most basic touch action. Can also double-tap to open files and folders. Tolerances are tuned to be larger than with a mouse. This works everywhere.
- **Drag** – Touch and slide your finger on screen. Like a dragging with a mouse, this moves icons around the desktop, moves windows, selects text (by dragging left or right), etc. This works everywhere.
- **Scroll** – Drag up or down on the content (not the scrollbar!) of scrollable window to scroll. This may sound basic, but it is the most used (and most useful – it's a lot easier than targeting the scrollbar!) gesture in the beta according to our telemetry. You'll notice details that make this a more natural interaction: the inertia if you toss the page and the little bounce when the end of the page is reached. Scrolling is one of the most common activities on the web and in email, and the ability to drag and toss the page is a perfect match for the strengths of touch (simple quick drags on screen). Scrolling is available with one or more fingers. This works in most applications that use standard scrollbars.
- **Zoom** – Pinch two fingers together or apart to zoom in or out on a document. This comes in handy when looking at photos or reading documents on a small laptop. This works in applications that support mouse wheel zooming.
- **Two-Finger Tap** – tapping with two fingers simultaneously zooms in about the center of the gesture or restores to the default zoom – great for zooming in on hyperlinks. Applications need to add code to support this.

- **Rotate** – Touch two spots on a digital photo and twist to rotate it just like a real photo. Applications need to add code to support this.
- **Flicks** – Flick left or right to navigate back and forward in a browser and other apps. This works in most applications that support back and forward.
- **Press-and-hold** – Hold your finger on screen for a moment and release after the animation to get a right-click. This works everywhere.
- Or, **press-and-tap with a second finger** – to get right-click, just like you would click the right button on a mouse or trackpad. This works everywhere.

For touch gestures, seeing them in action is important so here is a brief video showing the gestures in action:



Windows 7 Touch Demonstration

In order to make the gestures reliable, we tuned the gesture detection engine with sample gesture input provided by real people using touch in pre-release builds; these tuned gestures are what you will see in the RC build. We have a rigorous process for tuning. Similar to our handwriting recognition data collection, we have tools to record the raw touch data from volunteers while they perform a set of scripted tasks. We collected thousands of samples from hundreds of people. These data were then mined looking for problems and optimization opportunities. The beauty of the system is that we can replay the test data after making any changes to the gesture engine, verifying improvements and guarding against regression in other areas.

This has led to several important optimizations. For example, we found that zooms and rotates were sometimes confused. Detecting zoom gestures only in applications that don't use rotation has resulted in a 15% improvement in zoom detection.

Gesture	Total replayed	Reco*	Success Rate	Wrong gesture
Zoom	3466	2679	74.12	Rotate = 251
Rotate	3583	2681	73.12	Zoom = 300
Zoom Only	3466	3088	90.50	

Further analysis showed that many short gestures were going unrecognized. The gesture recognition heuristics needed to see 100ms or 5mm worth

of data before making a decision about what gesture the user was performing. The concern that originally led to these limits was that making a decision about which gesture was being performed too early would lead to misrecognition. In fact, when we looked at the collected user data, we found we could remove those limits entirely – the gesture recognition heuristics performed very well in ambiguous situations. After applying the change and replaying the collected gesture sample data, we found zoom and rotate detection improved by about 6% each, and short scrolling improved by almost 20%!

### Supporting Gestures

Gestures are built into the system in such a way that many applications that have no awareness of touch respond appropriately, we have done this by creating default handlers that simulate the mouse or mouse wheel. Generally this gives a very good experience, but there are applications where some gestures don't work smoothly or at all. In these cases the application needs to respond to the gesture message directly.

In Windows, several experiences have been gesture enabled. We've spent a considerable amount of effort on IE8 – ensuring scrolling and zooming are smooth and that back and forward are at your fingertips. Media Center, which is a completely custom interface ideally suited to touch, added smooth touch scrolling in galleries and the home screen. The XPS Viewer has gesture support that will could become a model for many document viewing apps. Scrolling and zoom work as you would expect. When zooming out beyond a single page, pages start to tile so you can view many at a time. When zoomed out in that fashion, double tapping on any page jumps back to the default view of that page. A two-finger tap restores the view to 100% magnification. These predictable behaviors become habit forming quickly.

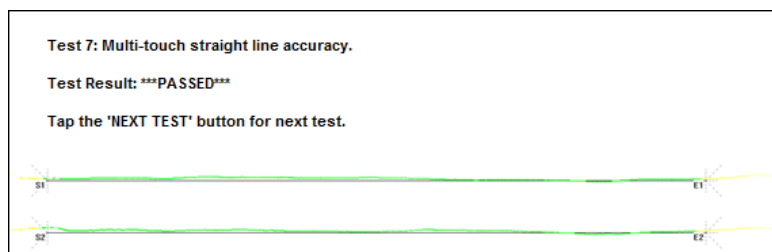
### Working with the Hardware Ecosystem

A major benefit of the Windows ecosystem is diversity – PCs come in all shapes and sizes. To help ensure that there is a great Windows Touch experience across the many different types of PCs we have defined a set of measurements and tests for Windows Touch that are part of the Windows Logo. We've been working with touch hardware partners since the beginning of Windows 7 to define the requirements and ensure they are ready for launch.

Our approach has been to provide an abstraction of the underlying hardware technology. We've specified a requirements for the quantitative aspects of the device, such as accuracy, sample rate, and resolution, based on the requirements to successfully enable touch features. For example, we have determined the necessary accuracy values for a device so people can successfully target common UI elements like close boxes, or what sample rate and resolution are required to ensure quality gesture recognition.

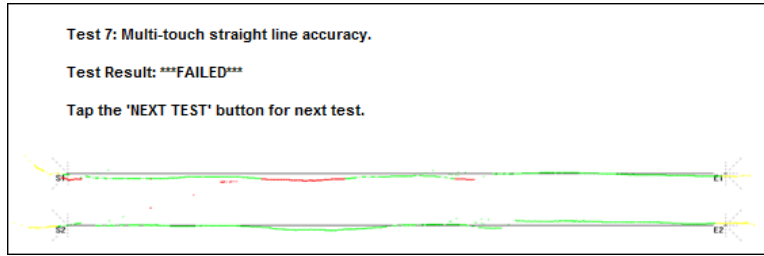
The requirements form the basis for the **Windows Touch logo** program. For consumers, the logo tells you that the PC and all of its components are optimized for Windows. Component level logo, which is what we grant to Touch digitizers helps the OEMs choose a device that will deliver a great touch experience.

Based on the quantitative requirements, we built an interactive test suite that includes 43 separate tests, all validating the core requirements under different conditions. There are single point accuracy tests at various locations on the screen, including the corners which are often harder for accuracy but critical to Windows. There are also several dynamic tests where accuracy is measured while drawing lines on the screen – see the screenshot below of Test 7. In this test, two lines are simultaneously drawn using touch along the black line from the start to the end. The touch tracings must remain within 2.5 mm of the black line between the start and end points. The first image below shows a passing test where the entire tracing is green (apologies for the fuzziness – these are foot long tracings from a large screen that have been scaled down).



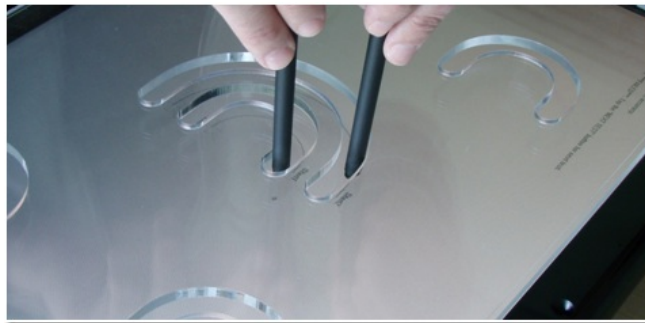
*Figure 1: A passing line accuracy test from the Windows 7 Touch logo test tool*

Not all devices pass the tests. Below is a screenshot of a device that is failing. This one has some noise – notice the deviation from the line in red. These errors need to be resolved before it would receive the logo. Errors like this can result in misrecognized gestures.



*Figure 2: A failing line accuracy test from the Windows 7 Touch logo test tool*

To ensure repeatability of the tests, we've built a set of plastic jigs with tracing cut-outs, see photo below. This particular jig is used for 5 of the tests and measures accuracy while tracing an arc.



*Figure 3. Plastic jigs with tracing cut-outs for testing.*

The testing tool is available to our partners now, we're working closely with several of them to help tune the performance of their devices to meet the requirements and deliver a great touch experience. We have set-up an in-house testing facility that will be testing every device submitted for Logo.

With the Release Candidate, OEMs and IHVs will be able to finalize the logo process for systems designed for Windows 7. Today we already have several hardware partners that have provided us with devices and drivers for testing.

### **Windows Touch for Software Developers**

We also want to talk a little about the touch platform for software developers. Windows 7 provides a rich touch platform for applications. We've already mentioned gestures, there's also a lower level platform that gives developers complete control over the touch experience. We think about it in a Good-Better-Best software stack.

## Good

The “good” bucket is what touch-unaware applications get for free from Windows 7. Windows provides default behaviors for many gestures, and will trigger those behaviors in your application in response to user input. For example, if someone tries touch scrolling over a window that is touch-unaware, we can detect the presence of various types of scrollbars and will scroll them. Similarly, when the user zooms, we inject messages that provide an approximation of the zoom gesture in many apps. As a developer you can ensure that the default gestures work just by using standard scrollbars and responding to ctrl-mouse wheel messages.

## Better

The “better” bucket is focused on adding direct gesture support and other small behavior and UI changes to make apps more touch-friendly. For instance, there is a new Win32 window message, [WM\\_GESTURE \(preliminary MSDN docs\)](#), that informs the application a gesture was performed over its window. Each message contains information about the gesture, such as how far the user is scrolling or zooming and where the center of the gesture is.

Applications that respond to gestures directly have full control over how they behave. For example, the default touch scrolling is designed to work in text centric windows that scroll primarily vertically (like web pages or documents), dragging horizontally does selection rather than scrolling. In most applications this works well, but if an app has primarily horizontal scrolling then the defaults would have to be overridden. Also, for some applications the default scroll can appear chunky. This is fine with a mouse wheel, but it feels unnatural with touch. Apps may also want to tune scrolling to end on boundaries, such as cells in a spreadsheet, or photos in a list. IE8 has a custom behavior where it opens a link in a new tab if you drag over it rather than click it.

In addition to gestures, there are subtle optimizations applications can make for touch if they check to see if touch is in use. Many of the subtle touch behavior optimizations in Windows were enabled in this manner. Larger Jump List item spacing for touch, larger hot spots for triggering window arranging, and the press and hold behavior on the desktop Aero Peek button with touch are all features written with the mouse in mind, but when activated via touch use slightly different parameters.

## Best

Applications or features that fall into the “best” bucket are designed from the ground up to be great touch experiences. Apps in this bucket would build on top of [WM\\_TOUCH](#) – the window message that provides raw touch data to the application. Developers can use this to go beyond the core system gestures and build custom gesture support for their applications. They can also provide visualizations of the touch input (e.g. a raster editing application), build custom controls, and other things we haven’t thought of yet!

We also provide a COM version of the [Manipulations](#) and [Inertia](#) APIs from Surface. The Manipulations API simplifies interactions where an arbitrary number of fingers are on an object into simple 2D affine transforms and also allows for multiple interactions to be occurring simultaneously. For instance, if you were writing a photo editing application, you could grab two photos at the same time using however many fingers you wanted and rotate, resize, and translate the photos within the app. Inertia provides a very basic physics model for applications and, in the example above, would allow you to “toss” the photos and have them decelerate and come to a stop naturally.

## Showcasing Touch

We’ve previously demonstrated, Microsoft Surface Globe, an interactive globe done in partnership with the [Surface](#) effort. Spinning the globe works as you would expect from a real-world globe, but with a touchable globe you can grab and stretch the view to zoom in, rotate, and move the view around. Interacting with the globe and exploring the world is the majority of the UI, and it is exceedingly easy to use with touch. Other features like search and adding markers to the map have also been designed with touch in mind.



Here's another video to get an idea of what we're talking about:



Windows 7 Touch Demonstration of Surface Globe

We're eagerly looking forward to seeing new touch-optimized user interfaces and interactions. If you're thinking about writing touch applications or adding touch support to your existing app, you should start with the MSDN documentation and samples.

### What Next?

We've noted several touch updates in the RC. If you have the Windows 7 Beta you can experiment with touch using a PC that supports multiple touch points. Please note that the multitouch PCs available today were developed while the Windows 7 requirements were also defined, so while we believe they can support Windows 7's requirements, only the maker of the PC can provide the logoed drivers for Windows 7 and support the PC on Windows 7. Keeping that caveat in mind, today there are a few multitouch PCs on the market:

- HP TouchSmart All-in-One PCs (IQ500 series & IQ800 series)
- HP TouchSmart tx2 Tablet PC
- Dell Latitude XT or XT2 Tablet PC

To enable multitouch capabilities on these PCs running the Windows 7 Beta you will need to make sure you have the latest multitouch beta drivers. Remember these are pre-release drivers and are not supported by Microsoft, Dell or HP. And again, they still need to pass through the Windows Logo process we described above before they are final.

- **For HP TouchSmart All-in-One PCs:** The pre-release driver is available from Windows Update. After you have installed the Windows 7 Beta, open **Windows Update** from the Start menu. You might have to click the "Check for Updates" link on your left so it will find the

driver, it is *Optional* right now so you'll have to select it before it will install. Alternatively you can download it from the [NextWindow website](#).

- **For the Dell Latitude XT, XT2 and HP TouchSmart tx2 Tablet PCs:** the drivers are available on [N-Trig's website](#). N-Trig is the company that makes the digitizer in these PCs (you should read the release notes, there are some limitations, like no pen support – that will be fixed by RC – you should be aware of and how to switch between Windows Vista and Windows 7).

We often get asked about single-touch PCs. Will they work with Windows 7? There are many types of hardware available for touch and many screens and PCs can provide single touch (usually based on [resistive touch](#) technology). A single-touch PC will have the same functionality on Windows 7 as it does on Vista, but this functionality will not be extended to the Windows 7 capabilities. As we noted earlier, Windows Touch in Windows 7 is comprised of a collection of touch enhancements, several of which require multitouch, that work together to deliver a great end-to-end touch experience.

As form factors change and the demands of our user interfaces change, input methods change and grow as well. We're excited about the unique benefits touch offers the user, and the new places and new ways it enables PCs to be used. We expect PCs of all form factors and price points to provide touch support and so it makes sense that these PCs will be able to take advantage of the full range of Windows 7 capabilities.

Windows 7 is designed to provide efficient ways to use multitouch for the most common and important scenarios, while being a natural and intuitive complement to the mouse and keyboard people use today.

Keep in Touch!

- Windows Touch Team

# Delivering a quality upgrade experience

Steven Sinofsky | [2009-04-07T03:00:00+00:00](#)

---

This is a little bit of a tricky post to write because we're going to be asking everyone using our Windows 7 Beta to help us out, but doing so is going to take a little time and require a bit of a commitment to helping test the next milestone. This has been a remarkably valuable and beneficial testing cycle for Windows as we have had a tremendous amount of very rigorous testing and usage. We've had millions of people install and use the Beta since January and as we've talked about, the feedback and telemetry have been of tremendous value as we finalize the product. The effort of Beta testers has contributed immensely to our ability to deliver a high-quality product to hundreds of millions of customers. We continue to follow the plan we have previously outlined and this post is no announcement of any news or change in plans. Since we know many people are running the Beta we want to provide a heads up regarding the behavior of the Release Candidate (RC) as it pertains to upgrades. Of course we are working hard on the RC and following the schedule we have set out for ourselves.

A big part of the beta process is making sure we get as much "real world" coverage of scenarios and experiences as possible and monitor the telemetry of those experience overall. One of the most challenging areas to engineer is the process of upgrading one release of Windows to another. When you think about it, it is the one place where at one time we need to run a ton of code to basically "know" everything about a system before performing the upgrade. During the development of Windows 7 we routinely test hundreds of original OEM images from Windows Vista and upgrade them and then run automated tests validating the upgrade's success. We also test thousands of applications and many thousands of devices as they too move through the upgrade process.

Many of you installed the Windows 7 beta on a PC running Vista. We received that telemetry and acted on it accordingly. We believe we've continued to improve the upgrade experience throughout the release. Similarly, based on our telemetry most of you did clean installations onto new drive partitions. Through this telemetry we learned about the device ecosystem and what drivers were available or missing. We also learned about PC-specific functions that required installing a driver / application (from XP or Vista) to enable support for buttons, connectors, or other hardware components. Together we get great coverage of the setup experience.

We've also learned that many of you (millions) are running Windows 7 Beta full time. You're anxious for a refresh. You've installed all your applications. You've configured and customized the system. You would love to get the RC and quickly upgrade to it from Beta. The RC, however, is about getting breadth coverage to validate the product in real-world scenarios. As a result, we want to encourage you to revert to a Vista image and upgrade or to do a clean install, rather than upgrade the existing Beta. We know that means reinstalling, recustomizing, reconfiguring, and so on. That is a real pain. The reality is that upgrading from one pre-release build to another is not a scenario we want to focus on because it is not something real-world customers will experience. During development we introduce changes in the product (under the hood) that aren't always compatible with what we call "build-to-build" upgrade. The supported upgrade scenario is from Windows Vista to Windows 7. Before you go jump to the comment section, we want to say we are going to provide a mechanism for you to use if you absolutely require this upgrade. **As an extended member of the development team and a participant in the Beta program that has helped us so much, we want to ask that you experience real-world setup and provide us real-world telemetry.**

If you do follow the steps below, you might run across some oddities after upgrade. We experience these internally at Microsoft occasionally but we don't always track them down and fix them because they take time away from bugs that would not only manifest themselves during this one-time pre-release operation. From time to time we've noticed on a few blogs that people are using builds that we have not officially released and complained of "instabilities" after upgrade. Nearly all of these have been these build-to-build issues. We've seen people talk about how a messenger client stopped working, a printer or device "disappears", or start menu shortcuts are duplicated. These are often harmless and worst case often involves reinstalling the software or device.

We're just trying to be deterministic and engineer the product for the real world. Speaking of the real world, many have asked about upgrading from Windows XP. There's no change here to the plan as has been discussed on many forums. We realized at the start of this project that the "upgrade" from XP would not be an experience we think would yield the best results. There are simply too many changes in how PCs have been configured (applets, hardware support, driver model, etc.) that having all of that support carry forth to Windows 7 would not be nearly as high quality as a clean install. This is something many of you know and already practice. We do provide support for moving files and settings and will prompt at setup time, but applications will need to be reinstalled. We know that for a set of customers this tradeoff seems less than perfect, but we think the upfront time is well worth it.

So when you try to upgrade a pre-RC build you will find that you're not able to and setup will tell you and you can then exit gracefully. You can install as a clean installation and use the [Windows Easy Transfer](#) feature as well (run this from your current installation of course) if you wish to

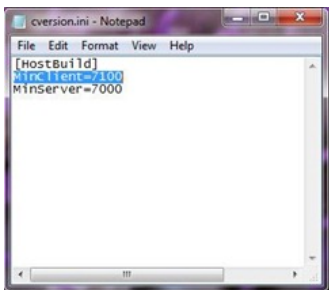
move your accounts, settings, files, and more. To bypass the version check, the instructions below will use a mechanism that is available for enterprise customers (so we are also testing this as well). It is not a simple command line switch. We didn't make it multi-step on purpose but wanted to stick to using proven, documented and tested mechanisms.

These instructions will be brief. Since everyone reading is a well-versed and experienced beta tester you know **ALWAYS BACK UP YOUR MACHINE** before running any OS installation and **NEVER TEST AN OS ON YOUR ONLY COPY OF ANY DATA**. Testing a pre-release product means just that—it is testing and it is pre-release. Even though this is a Release Candidate, we are still testing the product. We have very high confidence but even if an error happens once in 1,000,000 we want to make sure everyone is taking the precautions normal for a pre-release product.

One other related caution is **INSTALL ONLY OFFICIALLY RELEASED BUILDS FROM MICROSOFT**. It will always be tempting to get the build with the "mod" already done but you really never know what else has been done to the build. There's a thrill in getting the latest, we know, but that also comes with risks that can't even be quantified. For the RC we will work to release a hash or some other way to validate the build, but the best way is to always download directly from Microsoft.

Here's what you can do to bypass the check for pre-release upgrade **IF YOU REALLY REALLY NEED TO**:

1. Download the ISO as you did previously and burn the ISO to a DVD.
2. Copy the whole image to a storage location you wish to run the upgrade from (a bootable flash drive or a directory on any partition on the machine running the pre-release build).
3. Browse to the **sources** directory.
4. Open the file **cversion.ini** in a text editor like Notepad.
5. Modify the **MinClient** build number to a value **lower than** the down-level build. For example, change 7100 to 7000 (pictured below).
6. Save the file in place with the same name.
7. Run setup like you would normally from this modified copy of the image and the version check will be bypassed.



These same steps will be required as we transition from the RC milestone to the RTM milestone.

Again, we know many people (including tens of thousands at Microsoft) are relying on the pre-release builds of Windows 7 for mission critical and daily work, making this step less than convenient. We're working hard to provide the highest quality release we can and so we'd like to make sure for this final phase of testing we're supporting the most real world scenarios possible, which incremental build to build upgrades are not. At the same time everyone on the beta has been so great we wanted to make sure we at least offered an opportunity to make your own expert and informed choice about how to handle the upgrade.

We're always humbled by the excitement around the releases and by the support and enthusiasm from those that choose to run our pre-releases. We're incredibly appreciative of the time and effort you put into doing so. In return we hope we are providing you with a great release to work with at each stage of the evolution of the product. Our next stop is the RC...see you there!

THANK YOU!

--Windows 7 Team

PS: At Step 1 above many of you are probably thinking, "hey why don't you just let me mount the ISO and skip the plastic disc". We've heard this feedback and we deserve the feedback. We don't have this feature in Windows 7 and we should have. So please don't fill the comments with this request. There are several third party tools for mounting and if you've got a Vista image there's a good chance your PC came with those tools on it.

# Ink Input and Tablet PC

Steven Sinofsky | [2009-04-23T03:00:00+00:00](#)

*There's a strong community of developers who take advantage of the ink input/TabletPC functionality to develop unique solutions for specific markets (medicine, education, line of business) and create software in Windows that builds on this experience to streamline how these end-users interact with information on their PC (usually with unique form-factors such as slates or wall mounted PCs). Earlier this week I received a great email asking "what's new for us" from the head of development for one such ISV (medical software) and so we put together an overview of the new functionality. Several Program Managers on the team authored this post.*

*Also, as you have noticed, the site has had some uptime troubles over the past 10 days or so and I think we're all back to normal. That's ok since we've also been pretty busy in the Windows 7 hallway ?? --Steven*

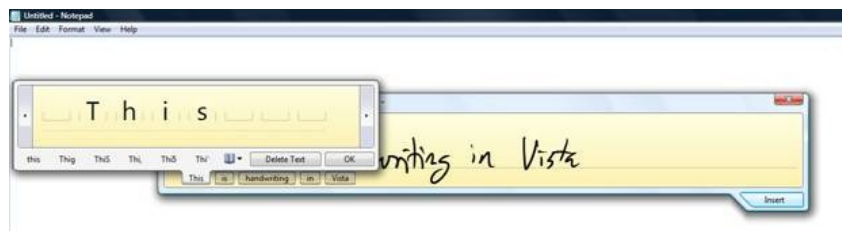
## Tablet PC Input Panel

Hi, my name is Jan-Kristian and I'm a Program Manager on the Core User Experience team for Windows 7. One of my focus areas is pen and touch text input, and I'd like to share some of the exciting things we have been working on.

The Tablet PC Input Panel, what we often called the TIP for short, is the tool to insert text using handwriting into any Windows application. It also has a soft-keyboard you can use for text entry. The Input Panel has been around since the first version of Windows XP Tablet PC Edition, and we've made steady improvements to the user experience in each version.

### The new Writing Pad

Our goal with the TIP is to make it as light-weight as possible so you can think about *what* you are writing and not *how* you are doing it. We received a lot of positive feedback on the improvements we did to the Input Panel in Windows Vista, but there were still areas that caused confusion or took more steps than necessary.



**Windows Vista Input Panel** – The handwriting recognition results are shown as small text bubbles under the writing surface. To verify recognition you need to look down at the bubbles, if you see an error you then tap on a bubble to bring up a secondary window for correction.

Based on analysis of our telemetry from Vista and usability tests we focused on two significant areas of improvement for Windows 7:

- **Simplify the experience** – Handwriting should be an easy, natural, flowing experience. What we found though, was that using the TIP

caused a “high cognitive load”, which means you have to think too much about what you are doing. Your eyes needed to dart back and forth between what you’ve written and the little bubbles down below and corrections meant entering another mode and even then often meant rewriting the whole word. Our goal was to simplify this and make it less taxing.

- **Add flexibility** – We’re all accustomed to the flexibility of using a mouse and keyboard for input. Handwriting with the Windows Vista Input Panel had minimal flexibility because the ink-based model made it hard to edit a sentence once it was written – there was no way to insert more text between words, or to easily replace words. Our goal here was to bring the editing experience more in line with what you are used to with the mouse and keyboard while exploiting the power of the pen.

#### Creating a new model

To achieve these goals we needed to make fundamental changes to the writing pad. As we explored different ideas we decided on a model where ink was converted in-place to text as the user was writing. Although this sounds like a straightforward UI model, there were a lot of open questions on what the right behavior should be: when do we convert, how big should the text be, what font should we use... The only way to make sure we created a natural and efficient handwriting experience was to get real user feedback. We utilized the [RITE \(Rapid Iterative Testing and Evaluation\) method](#). RITE testing is a cycle-based usability method that was developed at Microsoft as part of usability testing of the Age of Empires II game. For each cycle you try to make small improvement to the user experience and then you re-test to see how well it worked. We went through roughly 20 cycles before we had a design that we felt was ready to be documented.

One of the most important things we adjusted during RITE testing was the timing for the automatic ink to text conversion. Converting too early or too late would break the user experience; to get this right we had to do a lot of behind the scenes work. Our final solution is a combination of a distance trigger (automatically adapting to the user’s average word spacing), recognizer-result-based trigger, and a time-based trigger. Another factor was the text size, in the end we use dynamic sizing to closely match the size of the handwriting.

The new text-based UI in the writing surface allows you to get to the text they wanted faster. Having a single representation of the text makes the experience less complex and reduces the height of the Input Panel. Using text instead of ink makes the writing surface much more flexible as we can move the text around as much as we want – inserting a word between two words is now as simple as just starting to write in the space and we will auto-grow the space as much as needed.

With the ink to text conversion working we needed a correspondingly natural way of editing recognized text. Gestures seemed to be the perfect solution for this - we were creating a pen-based UI, so we should use the pen. We limited ourselves to a small number of gestures: delete, split (add space), and join. We collected samples of how people would perform these three actions on paper. Based on our collected data we then created our gestures. To make the gestures discoverable, we added the “gesture panel”, which is an interactive “cheat sheet” in the title bar of the Input Panel.

Let’s take a look at how this all comes together in the new Windows 7 writing surface [Ed. Note, used YouTube with Windows Live Photo Gallery on Windows 7]:



Windows 7 - Handwriting overview

**Writing Pad:** *The new writing pad in action, animation is used to provide meaningful transitions so that the user can easily see the result of their actions.*

#### Smart Corrections

Our telemetry showed us that corrections were one of the more painful parts of using the TIP in Vista, to correct a word you often had to rewrite all of the characters. In Windows 7, we leveraged work from Microsoft Research to design the Smart Correction feature to make word corrections much faster. Now you just start correcting a word left-to-right and Windows performs a new recognition every time you enter a character. This constrained recognition will almost always give you the desired result within a few character corrections.



Windows 7 - Handwriting Smart Correction

**Smart Corrections:** *“worked” is auto-corrected to “wonderful” with just a single character change. All you have to do is start correcting the word from the left and it will keep updating until you get the word you want.*

#### Entering URLs



One extra writing pad feature worth mentioning: our instrumentation data showed that the most used applications with the Input Panel are web browsers, and when you are browsing one of the main scenarios is to enter URLs.



Windows 7 Handwriting - URL correction

*Entering URLs: The flexibility of the new writing pad makes entering URLs easy by pre-populating parts of the URL*

Notice how the different URL segments are separated and all have alternates that make sense. The alternates are based on what you use most frequently, so if you choose “.net” a lot then that will become the top alternate and set by default in the URL template. The “Insert” button also changed to “Go to” to let the user insert the URL and navigate to it with a single click.

## Touch Keyboard

The Input Panel also has a soft-keyboard available which is great for the Pen or Touch. Some of the updates we made might seem like only visual changes, but they were very deliberate and have a big impact on the usability of the touch keyboard. For example, touch screen PCs are often used in mobile situations, we had to be very careful with the color and contrast of the key labels to make sure they are visible on a dimmed screen or in less than optimal light conditions.



## *The Windows 7 Touch Keyboard*

One of the challenges with using a touch based keyboard is the lack of tactile feedback. Coupled with this is the fact that user's fingers cover keys

as they are being tapped. How does the user know that they hit the right key (or even hit a key at all) when they are covering the key with their finger? If a user has to switch focus between the text field and the touch keyboard for every key press they will quickly tire of typing using touch. We wanted to give the user a simple little nudge; ‘we heard you’, and ‘yes, you just hit this key’. Our solution was to let the released key have a short glow fade-out effect. This glow feedback gives the user a subtle confirmation that they hit the key they wanted (or not).

The keyboard now supports multitouch so you can press “ctrl+c” or “shift+a” etc. We also enabled key rollover, meaning you can press another key before you finger has lifted off the previous one – this enables a much more natural typing experience. Don’t worry, if you prefer the sticky modifier-key model, where you press “Ctrl” then press “c”, it is still supported.

## Predictive Text and Handwriting Personalization

Hi, my name is Jen and I’m a Program Manager on the Tablet PC Handwriting Recognition team. In our [previous post](#) you heard from my co-worker Yvonne about new handwriting recognizers. I’m going to talk about some of the new features that leverage or augment our recognizers including Predictive Text and Personalization as well as our new East Asian Recognizers.

### Text Prediction

One of our goals on the Tablet PC team is to make it as efficient as possible to enter text when using your keyboard is inconvenient or impractical. To achieve this we’ve made investments in the TIP, we’ve improved overall handwriting recognition accuracy, and we’re leveraging some of these same technologies to deliver Text Prediction on the soft keyboard. Text Prediction offers possible completions for the word being entered, and may offer suggestions for next words or phrases.



**Text Prediction** – Here I am trying to input the word “Microsoft” using the US English soft keyboard. After entering the first two letters “Mi”, the word “Microsoft” is proposed as the first option. I can then select this option and have the word “Microsoft” inserted into a document.

In Windows 7, we support Text Prediction for English, French, Italian, German, and Spanish using the soft keyboard as well as for Traditional Chinese and Simplified Chinese using handwriting in character by character mode. This section will focus on the Latin based languages; examples for Chinese can be found in the following section.

As we developed Text Prediction, our primary goal was to speed up user input. To do so, we had to make sure predictions were relevant. Our recognizers use a *lexicon* to improve recognition accuracy. The system lexicon ships with the recognizer and is a fancy name for a word list of the most commonly used words in a given language. Using this lexicon, the out-of-the-box predictions are good, but by including additional user-specific words (your words), we can improve accuracy significantly. This is where Text Harvesting comes in.

In Windows Vista, we shipped Text Harvesting (or Automatic Learning) for US English and UK English to improve handwriting recognition. In Windows 7, this feature will be available for all languages. It allows us to automatically extend the system lexicon based on the words you type

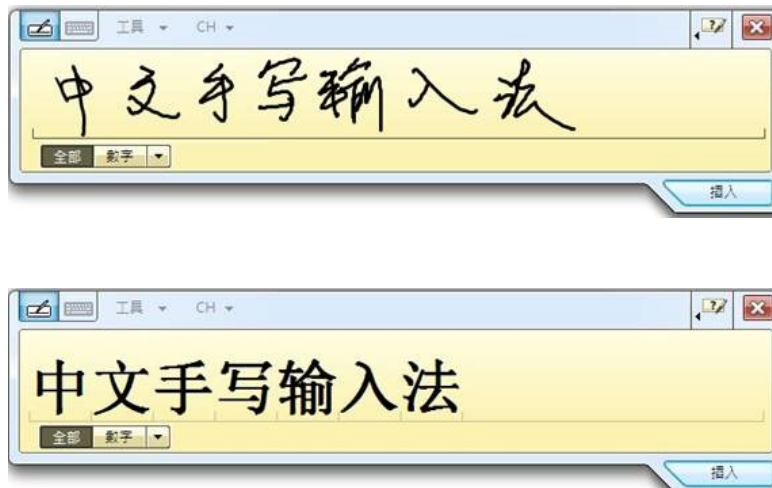
when writing email. Text Harvesting is done on a per user basis, so your data is specific to you. From the results of Text Harvesting, we build a new lexicon containing your specific vocabulary and also increase the probability of words already you commonly use, this is use for both handwriting recognition and text prediction. The results are impressive, after augmenting the lexicon with your words and usage patterns, prediction can seem almost magical in its ability to predict which words you are going to enter next!

Windows 7 also includes support for Custom Dictionaries, these are specialized word lists that can be added to the system. Companies can develop domain specific dictionaries, such as for medicine, chemistry etc. and add them to the system – predicting *acetaminophen* is a lot faster than typing it!

### Improvements in East Asian Handwriting Recognition

Significant improvements were made to handwriting recognition on the four East Asian languages we support: Traditional Chinese, Simplified Chinese, Korean and Japanese. For many people, handwriting is an efficient input method for these languages due to the large character set.

There are two input modes for East Asian handwriting: character by character mode (or box mode) and freestyle mode (or line mode). In box mode, you input a single character at a time into a fixed width box. In freestyle mode, you write the characters cursively on a line and do not have to concentrate on spacing. Which mode you chose depends on your preference; box mode is slightly more constrained but has text prediction and is more accurate, whereas line mode is closer to natural handwriting.



**Traditional Chinese in Line Mode** – The top pane contains the user’s writing and the bottom pane contains the recognized text.

In addition to core accuracy across all these languages improvements, we also use personalization to improve the user experience. One method of personalization is to adapt to how you write using the Shape Collector. The Shape Collector is a Wizard that allows you to train handwriting recognition on your individual handwriting style. For the four East Asian Languages, you can use the Shape Collector in a “troubleshooting” mode to improve recognition of a specific character or word, or to add a character or word that is unsupported.

We also learn as you write and correct mistakes. If you write a character and it is initially misrecognized, you can view the alternates list and select the character you intended. We will learn based on this action, and be more likely to provide that as the first choice the next time you write the character.

In Windows 7, Simplified Chinese and Traditional Chinese also support Text Prediction in box mode. For these languages, Text Prediction is

especially valuable as writing individual characters can be time consuming. The user writes in character by character mode and is provided with options to complete their word or phrase without having to write the whole thing. In the case below, the user wants to input ?????? and only has to input the first two characters (??) to get the desired text as a prediction. The gray words represent what has been entered and the black shows the predictions.



Notice in this example that Text Prediction is working on both characters together (??) as well as just the second character (?). As with the other languages, Text Prediction also works with user-specific vocabulary. If a user inputs the same words multiple times, the recognizer will learn this behavior.

We have made significant improvements to Traditional Chinese, Simplified Chinese, Japanese, and Korean handwriting recognition since Windows Vista. These were based on improving our overall customer experience by improving accuracy and throughput. Our goal is to give customers a more natural way to input in these languages and a viable alternative to keyboard use.

## Math Handwriting Recognition

Have you ever written a math paper in Word or performed calculations in Mathematica, and spent hours creating equations using a multitude of buttons or a complex linear format, thinking: "Oh, what I wouldn't give for an easy-to-use input method?" Well, your wishes have just come true, in addition to improving handwriting in Windows 7, we have also invested in recognizing ink drawn math equations.

Hi, my name is Marko and I am the Program Manager for math recognition in Windows. Let me show you the Math Input Panel that provides you with the most natural and efficient way of math input: handwriting recognition. We have taken a completely new approach to this problem and raised math handwriting recognition to a whole new level in terms of functionality, performance and area coverage.

The Math Input Panel (or MIP) is designed to be used with a tablet pen on a Tablet PC, but you can use it with any input device such as a touchscreen, external digitizer or even a mouse. MIP outputs the recognition result via the clipboard in MathML format, a standardized mathematical markup language. Any equation you write and recognize in MIP reaches your destination application in a completely editable form—you can insert and edit the output as you would edit any text.

We spent a lot of time researching and identifying as many areas of math as possible and endless different math notations. The final result is a great coverage of high school and college level math, and of even more advanced areas.



# Engineering Windows 7 Graphics Performance

Steven Sinofsky | [2009-04-25T03:00:00+00:00](#)

*One of the areas of any release of Windows that receives a significant amount of testing and scrutiny is the performance of graphics—desktop graphics all the way to the most extreme CAD and game graphics. The amazing breadth of hardware supported for Windows and the broad spectrum of usage scenarios contributes to a vibrant ecosystem with many different goals—from just the basics to the highest frame rates on multiple monitors possible. In engineering Windows 7 we set out to improve the “real world” performance of graphics as well as continue to improve the most extreme elements of graphics. This is work we do in Windows 7 and work our partners do as they work to improve the underlying hardware/software combination through drivers (note: Windows Vista drivers continue to work as they did in Windows Vista, but we’ve also been working with partners on updated drivers for Windows 7 which many of you have been testing through Windows Update downloads). This post looks at this spectrum of engineering as well as the different ways performance is measured. Ultimately we want to inform you about what we have done in engineering Windows 7, while we leave room for the many forums that will compare and contrast Windows 7 on different hardware and in different scenarios. This post is written by Ameet Chitre, a program manager on our Desktop Graphics feature team. --Steven*

If you have gone online to check out or purchase a new PC, you must have noticed that “faster graphics” and “great performance” are always some of the key selling points. People have come to expect faster systems which enable them to edit photos, do email, watch high-definition videos and play the latest 3D games all with greater ease, often shuffling between these tasks seamlessly. Quite a few of these users refer to the enthusiast community blogs and various review sites which run graphics benchmarks and report results evaluating how fast the graphics of new hardware or software performs. Traditionally graphics performance has been measured and analyzed through 3D games but it also impacts what we call “desktop scenarios” - such as when you are using the Windows UI, moving or maximizing windows, or scrolling within Word or IE etc. The performance requirements for these desktop scenarios are quite different from 3D games. In fact, this is the reason in Windows Vista Experience Index (WinEI) we give you rating for these two scenarios separately, highlighted in the image below:

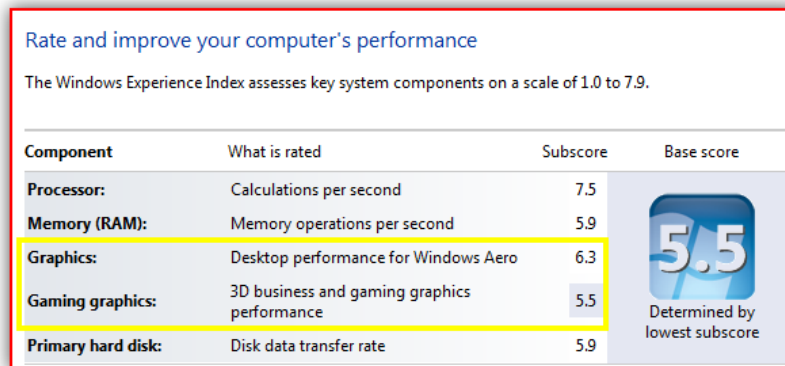


Figure 1. WEI sample with Graphics capabilities highlighted.

Graphics performance is usually assessed through many benchmarks. These can be classified into 2 broad categories:

- **Scenario benchmarks:** These report performance of particular scenarios, e.g. the frame rate when flying over a terrain in a flight simulator. Many of the popular games have in-built benchmark modes that demonstrate how fast the graphics perform in scenes that are typical for that game.
- **Synthetic benchmarks:** These highlight the performance of a particular aspect of graphics such as the ability to draw a number of lines as fast as possible.

However, there are plenty of things that we all do on our PCs that don't have benchmarks tracking them that are still quite critical to make fast. In these cases we use the instrumentation within Windows to obtain timing information and then analyze the performance.

This blog entry discusses various aspects of graphics performance - both gaming and desktop graphics performance. It calls out the changes we made in Windows 7 to address user feedback as well as to take advantage of modern hardware to improve graphics performance.

## Desktop Responsiveness

Many have experienced scenarios where an application, or Windows itself, stops responding momentarily. This is type of a performance issue that can be impacted significantly by the performance of graphics in the PC. We categorize these as desktop responsiveness issues. Improving responsiveness, both in real terms and by avoiding non-responsive moments, is one of the key ways that performance is improved in the system. It is also hard to measure.

Measuring desktop responsiveness is a hard problem since a number of issues which affect responsiveness aren't easily reproducible and there is a great variety of them. They are rarely caught by either kind of benchmark as these issues are dependent on real-world combinations of factors. For Windows 7 we spent a great deal of time looking at these performance glitches using a mechanism in test versions of Windows 7 which has the ability to record key OS events and when they occurred. During real-world testing when we encounter a responsiveness problem, the tester can hit a record key and enter a small description of the issue encountered. The event history with diagnostic information called a "performance trace" is written out to a file and uploaded to a server where a team of performance analysts parse the data to figure out the cause of the responsiveness issue. This process has been successful to the extent that today most responsiveness issues can be quickly tracked down and root-caused.

Using this methodology, we analyzed thousands of desktop responsiveness traces where the tester experienced a frozen desktop anywhere from 100msec to several seconds. The type of issues ranged from an antivirus blocking disk access for all applications while updating itself on the vendor's website to an application doing network access from a UI thread. In a significant portion of these traces, we found that a GDI application is waiting on another GDI application which is experiencing slowdowns due to excessive paging activity. This was the single-most frequently occurring cause of all desktop responsiveness issues, which without this data we probably would not have assumed. Based on these investigations, we worked to improve the architecture in these two key areas:

1. **GDI Concurrency:** Improve responsiveness when multiple applications are running. This required a re-architecture of the code around GDI (Graphics Device Interface) synchronization objects or "locks".
2. **Reduction of overall memory footprint of Windows:** Reduce the memory overhead of composition in the Desktop Window Manager (DWM), which is the component responsible for rendering the desktop. This helps reduce overall paging activity and thus is especially important for low-memory PCs, especially using [shared graphics memory](#).

These are described in more detail below.

### GDI Concurrency

A number of performance traces we investigated in the context of desktop responsiveness pointed us to the design of a key synchronization mechanism in GDI. The performance challenge happens because the design of GDI in Windows Vista allows only a single application to hold a system-wide exclusive global lock. While this seems obvious in hindsight, when this decision was originally made the performance characteristics of different parts of the system made this optimistic implementation perfectly reasonable.

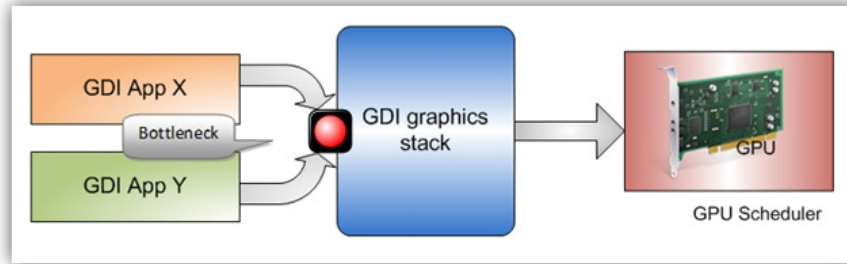


Figure 2. Existing architecture of GDI concurrency.

GDI applications running simultaneously vie for this global lock in order to render on the screen. The application that accesses the global lock prevents other applications from rendering till it releases the global lock. The situation often gets exacerbated when the application that is holding the lock needs to page in a large amount of memory from the disk since moving the memory from the disk to RAM takes a relatively long time. The above picture shows two GDI applications running simultaneously, contending for the global lock. If App X gets hold of the lock, it can render to the screen while App Y is unable to do so and waits for App X to finish.

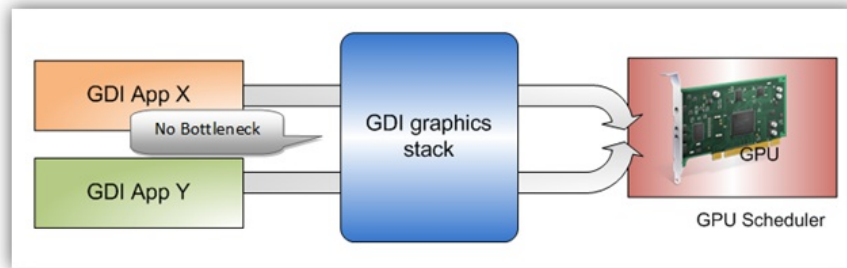


Figure 3. Windows 7 architecture of GDI concurrency.

The solution to the problem was therefore to reduce the lock contention and improve concurrency by re-architecting the internal synchronization mechanism through which multiple applications can reliably render at the same time. Contention due to the global exclusive lock is avoided by implementing a number of fine-grained locks which are not exclusive but aid parallelism. The increased number of fine-grained locks adds a small overhead for scenarios where only a single application is rendering at a time.

Special attention was paid to GDI application compatibility as changing internal synchronization mechanism in the most widely used API stack could potentially give rise to timing issues such as deadlocks and rendering corruption.

This work also resulted in better rendering performance of concurrent GDI applications on multi-core CPUs. Multi-core Windows PCs benefit from these changes as more than one application can now be rendering at the same time.

After the GDI concurrency work was implemented in the Windows 7 builds leading to the Beta, we saw a large reduction in the number of desktop responsiveness issues reported by our testers which were caused by one application blocking another one due to GDI. To further validate the scalability of our new implementation, we wrote tests that draw 2D GDI primitives and measured the rendering throughput by launching simultaneously multiple such applications. The throughput is measured by adding together the frame rate (FPS) of each application window. Below is a sample of these results on a quad-core CPU system.



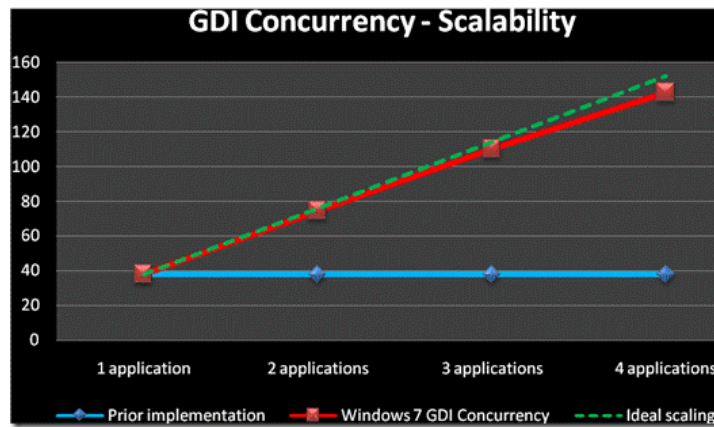


Figure 4. GDI Concurrency and Scalability.

Without the Windows 7 GDI concurrency, the rendering throughput of these applications is effectively limited to the performance of a single CPU core. Since only a single application can acquire the global exclusive lock while the others are waiting, this scenario doesn't benefit from multiple CPU cores. This demonstrates that GDI applications in Windows 7 are now much less dependent on one another. This benefit will not need any new display drivers; it will work on any Vista (WDDM 1.0) and newer display drivers.

### Desktop Graphics - Reduced Memory Footprint

Another area which affects system responsiveness is memory usage. Simply put, increased system memory (RAM) usage leads to an increased paging activity which directly leads to *reduced* system responsiveness. Thus, for the best responsiveness, all applications, processes and OS components need to use as little system memory as possible.

In Windows Vista, the amount of memory required to run multiple windows scales linearly with the number of windows opened on the system. This results in more memory pressure when there are more windows or if the monitors have higher resolution. It gets worse if you have more than one monitor. As part of investigating various means to improve system responsiveness, we saw a great opportunity in reducing the usage of system memory by DWM. In Windows Vista, every GDI application window accounts for two memory allocations which hold identical content – one in video memory and one in system memory. DWM is responsible for composition of the desktop through the graphics hardware. Hence, it requires a copy of the same allocation in video memory, which is easily accessible by the graphics hardware. The duplicate copy present in system memory is required because GDI is being rendered utilizing the CPU completely in the operating system without any assistance or “acceleration” by the graphics hardware. As the CPU performs all the tasks for rendering GDI applications, it requires an easily accessible cacheable copy of memory.

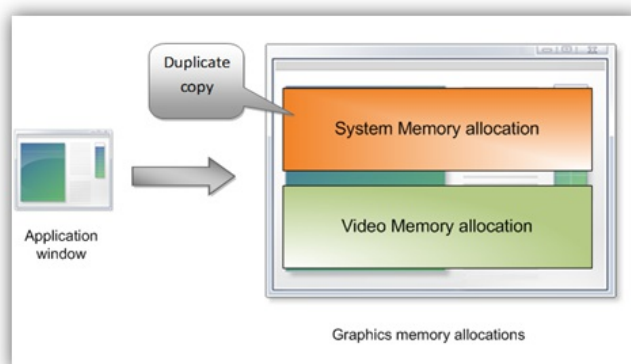


Figure 5. Existing memory allocations.

Windows 7 saves one copy of the memory allocation per application window by getting rid of the system memory copy entirely. Thus, for a GDI application window visible on the desktop, the memory consumed is cut in half.

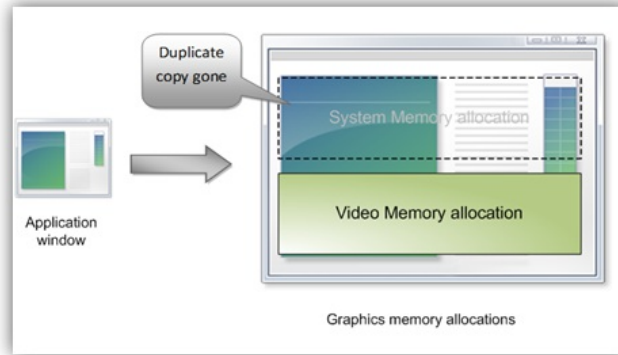


Figure 6. Windows 7 memory allocations.

We achieved the reduction in system memory by accelerating the common GDI operations through the graphics hardware - the WDDM drivers accelerate these to minimize the performance impact of the CPU read-back of video memory. This was necessary as performing these operations otherwise on the CPU would incur a heavy performance penalty. In order to decide which GDI operations to accelerate, it was important to understand the usage pattern of various GDI applications. We profiled the top 100 GDI applications to learn more about their calling patterns and frequency and nature of the GDI operations.

Process	TotalElap	DDI	ElapsedTi	NumDDI	WaitReac	CpuTime	NumReac	WaitTime	NumFlus	FlushTim	N
realplay.exe (68)	59457	DrvBitBlt	32152	230	2907	26898	8	2969	11	1910	
buildscout.exe (	1242	DrvBitBlt	515	10	121	181	2	121	1	128	
Virtual PC.exe (	70166	DrvColorF	25477	838	12859	6101	70	12859	28	3677	
SqlWb.exe (410)	14288	DrvBitBlt	8123	202	2366	4582	19	2374	13	488	
Prwin12.exe (35	464466	DrvStroke	2096	74	898	998	3	898	1	20	
WINZIP32.EXE (	53384	DrvLineTo	2427	54	908	316	5	908	5	1161	
CorelPP.exe (31	48754	DrvStroke	3577	46	2446	578	6	2446	6	375	
RealPlayerTest.	597	DrvBitBlt	580	3	342	96	3	342	0	0	
AcroRd32.exe (5	70153	DrvColorF	2777	418	498	2008	4	498	1	125	
devenv.exe (60	63824	DrvStroke	637	5	101	133	5	101	5	235	
explorer.exe (1	23322	DrvColorF	653	32	255	266	4	255	0	0	
CorelDRW.exe (	95371	DrvStroke	5531	51	3820	703	13	3820	9	491	
guidgen.exe (12	3851	DrvColorF	819	16	334	164	4	334	0	0	
Dreamweaver.e	552864	DrvBitBlt	361520	5599	8793	268883	9	9640	149	15666	
pi.exe (3100)	463068	DrvColorF	16599	1730	3623	10549	52	3623	3	183	
InocIT.exe (320	30867	DrvBitBlt	24898	216	17522	4587	24	17522	22	2173	
Shellscln.exe (31	10284	DrvBitBlt	5194	70	2244	916	16	2244	16	1136	
Streets.exe (51	168127	DrvStroke	1065	2	455	79	2	455	1	444	
Contribute.exe	91805	DrvLineTo	1210	118	635	505	1	635	1	47	
avginet.exe (44	39493	DrvBitBlt	1251	4	939	150	1	939	1	152	
winamp.exe (44	83630	DrvStroke	1302	6	672	202	6	672	3	223	
Prwin12.exe (35	464466	DrvStroke	1352	3	749	318	3	749	3	73	

Figure 7. Calling patterns and frequency of GDI operations for 100 GDI-based applications.

Based on real-world application statistics, a tiny snapshot of which is seen above, we worked with our graphics IHV partners to provide support

in their drivers to accelerate the most commonly used GDI operations. Windows 7 systems with these updated drivers, called “WDDM v1.1” will thus benefit from this memory savings work. Please note that WDDM 1.0 drivers continue to function and are fully supported on Windows 7. You might have seen Windows Update providing these 1.1 drivers during the Beta—these drivers are themselves in Beta.

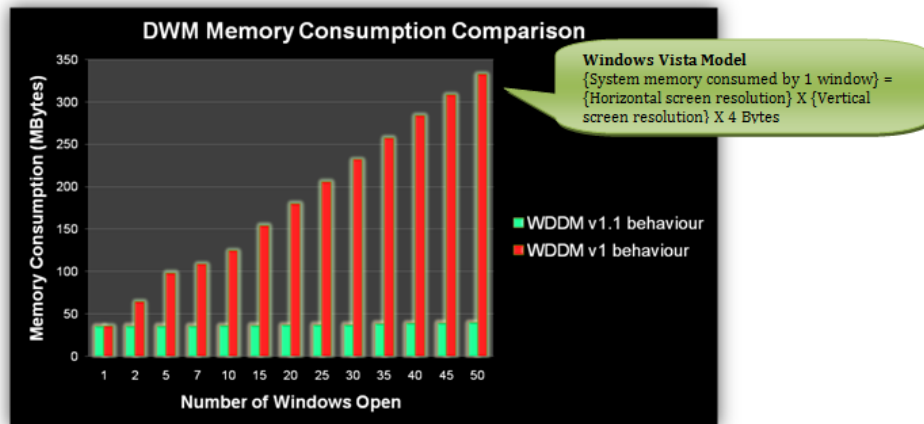


Figure 8. Desktop Window manager memory consumption comparison using WDDM 1.1 v. WDDM 1.0.

The above data shows that the memory savings become more and more pronounced when you have multiple application windows visible on the desktop. Since you save a lot of system memory, the paging activity gets reduced – as a result, your system responsiveness improves for the same workload.

Certain trade-offs had to be made for the desktop responsiveness improvements which benefit a wide range of systems. For example – the elimination of the duplicate system memory copies which “speed up” certain operations introduced slightly reduced performance as the CPU now has to read data back from the video memory. An analysis of real-world application statistics showed that these operations were rare. However, certain GDI micro-benchmarks which issue these operations show some performance degradation. This is something to note if you are running existing benchmarks that stress specific GDI operations repeatedly, which isn’t necessarily a reflection of real-world performance. Our observation has been that these slow-downs do not impact the end-user functionality directly and that the memory savings directly result in Windows 7 being much responsive overall. The improvements overall are definitely noticeable on memory constrained PCs with shared memory graphics.

## Gaming Performance

No article on *graphics performance* is complete without talking about gaming, which is still the most widely analyzed and discussed aspect of graphics performance. There are a number of popular benchmarks such as 3D Mark as well as in-game benchmarks which are really a mode in which you can run your game where it renders the game scenes and animations without any user interaction. This area has thus been well tracked by the gaming industry through various industry benchmarks, which are pretty realistic and representative of actual games. The different benchmarks and tests are widely discussed and gamers all well-versed in the subtleties of these measurements and translating them into recommendations depending on their hardware, drivers, and gaming expectations.

For Windows 7, we have worked closely with our Graphics IHV partners, helping them improve the WDDM drivers’ gaming performance with specific changes to how Windows 7 works under the hood, while maintaining the same driver model and compatibility. Our continued investments in performance tools has helped us and our IHV partners track down and analyze various gaming performance bottlenecks and fix them in subsequent driver releases. The fundamentals of the Windows Display Driver Model remain unchanged in Windows 7. Some policies around GPU scheduling and memory management were changed to enable better performance in certain scenarios.

Because these benchmarks are very sensitive to the specific hardware, firmware, drivers, and overall system and because these are so widely measured and discussed elsewhere we are going to leave these comparisons to third parties. Like many areas in Windows 7, our commitment is to engineer even better performance across many dimensions. We believe it is better for you to experience these efforts directly. In comparing Windows 7, we would encourage the comparison using Windows Vista SP1 and keep in mind the difference you might see in WDDM 1.0 v. 1.1 and that the 1.1 drivers are still under development.

### *In Closing...*

As you can see, in engineering Windows 7 we have worked hard to improve the architecture for graphics for real-world performance. It benefits both ends of the hardware spectrum – by enabling low physical memory systems to run a leaner and faster Windows and at the same time enabling multi-core PCs render multiple graphics applications much more efficiently.

-Ameet

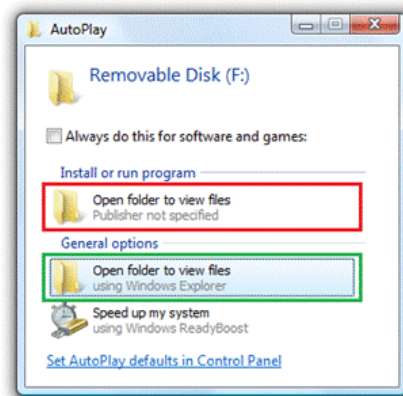
# Improvements to AutoPlay

Steven Sinofsky | [2009-04-27T03:00:00+00:00](#)

*As mentioned before on this blog (regarding our [UAC changes](#)) and on the IE blog (regarding the [SmartScreen® filter for malware](#)), we have an increased focus to enable customers to be in control and feel confident about the software that they choose to run on their computers. Folks on this blog have also commented about the concerns they have specifically in the AutoPlay area. This blog entry addresses some of the changes that we have made to increase customer confidence when using their media and devices with Windows. It is authored by Arik Cohen, a program manager on the Core User Experience team. –Steven [Note: There was a technical problem so this post was reposted in its entirety.]*

Certain malware, including the Conficker worm, have started making use of the capabilities of AutoRun to provide a seemingly benign task to people – which masquerades as a Trojan Horse to get malware onto the computer. The malware then infects future devices plugged into that computer with the same Trojan Horse. For further information about Conficker please visit <http://www.microsoft.com/protect/computer/viruses/worms/conficker.mspx>

In the following example for a USB flash drive that has photos, malware registers as the benign task of “Open folders to view files.” If you select the first “Open folders to view files” (circled in red), you would be running malware. However, if you select the second task (circled in green), you would be safe running the Windows task.



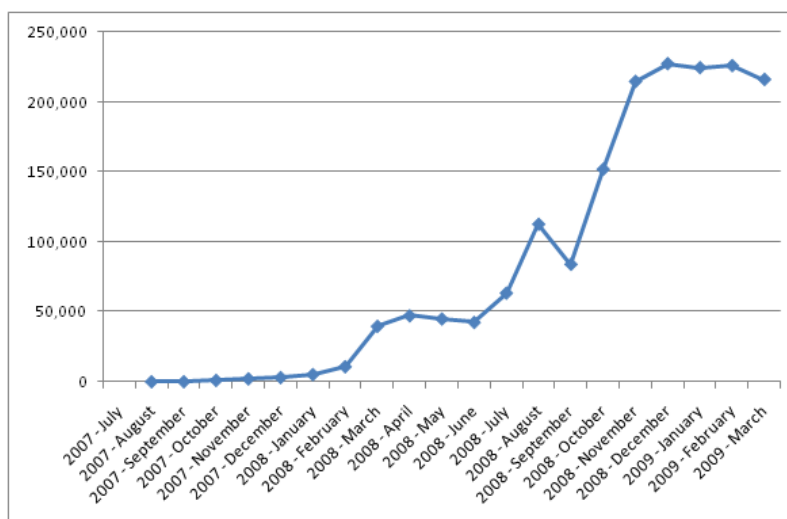
**Infected USB AutoPlay**

People are confused why they have two tasks that appear to do the same thing – and even a knowledgeable person who is careful not to run software from an untrusted source can easily make the mistake of selecting the first task. As a result, people lose confidence and don't feel in control.

## A growing attack

While presenting an AutoRun task in AutoPlay has been available since Windows XP, we have seen a marked increase in the amount of malware that is using AutoRun as a potential method of propagation. According to the [Security Intelligence Report](#), an enterprise study by Forefront Client Security found that the category of malware that can propagate via AutoRun accounted for 17.7% of infections in the second half of 2008 – the largest single category of malware infections.

The chart below shows the increasing amount of detection reports by Microsoft anti-virus software of the class of infections that spread via AutoRun. (Note: The actual method of infection cannot be determined.)



### Infection Detections of Malware that Spread via AutoRun

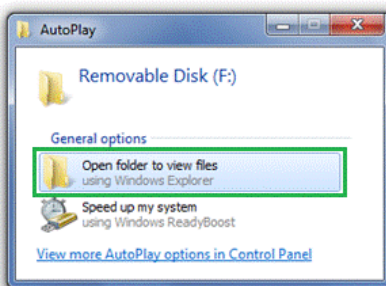
Currently, disabling AutoPlay completely is the only solution for consumers and enterprises to gain confidence with the use of USB flash devices on their computer. Guidance on disabling AutoPlay is available [here](#).

#### Increasing customer confidence

Windows 7 introduces key changes to AutoPlay that keep you from being exposed inadvertently to malware like Conficker when doing your common scenarios with devices (e.g., get to the files on your USB flash drive, download pictures from an SD card, etc.).

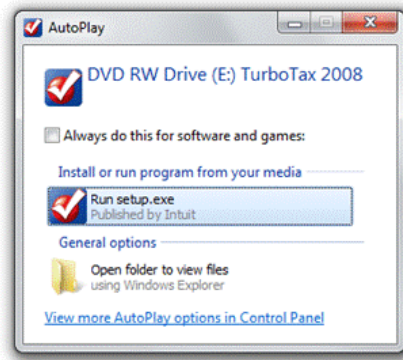
In particular, Windows will no longer display the AutoRun task in the AutoPlay dialog for devices that are not removable optical media (CD/DVD.) because there is no way to identify the origin of these entries. Was it put there by the IHV, a person, or a piece of malware? Removing this AutoRun task will block the current propagation method abused by malware and help customers stay protected. People will still be able to access all of the other AutoPlay tasks that are installed on their computer.

With these changes, if you insert a USB flash drive that has photos and has been infected by malware, you can be confident that the tasks displayed are all from software already on your computer:



#### Infected USB AutoPlay after AutoPlay changes

On the other hand, if you insert a CD that offers software to install, Windows will still display the AutoRun task provided by the ISV during their media creation process. For example:



### **AutoPlay for a CD that offers an AutoRun Task**

You will first see this updated AutoRun experience in the Windows 7 RC build, and we will be bringing this change to Vista and XP in the future.

#### **Ecosystem Impact**

We are working with our ecosystem partners to help mitigate situations where this AutoRun change will have an impact on them.

CDs and DVDs (including CD emulation), where the IHV specified AutoRun task authored during manufacturing, will continue to provide the AutoRun choice allowing customers to run the specified software. IHVs of generic mass storage devices should expect that people will browse the contents of the device to launch any software. The new behavior will allow customers to continue to use AutoPlay (including all Windows and ISV installed tasks) to access their media and devices while not being presented with tasks from malware. Additionally, device classes, such as portable media players and cell phones, now support Device Stage™ on Windows 7. Device Stage offers the IHV a multifunction alternative to AutoPlay where they can present links to software and common tasks, and provides additional features as you use the device.

As you try out the Windows 7 RC, we hope these changes will make you feel more confident and in control when using your media and devices.

-Arik Cohen

# A Little Bit of Personality

Steven Sinofsky | [2009-05-02T03:00:00+00:00](#)

---

*Greetings! Based on the data we're seeing we know a lot of folks on MSDN/TechNet/Connect are probably busy using the RC (Release Candidate) for Windows 7. Thank you!!! And of course many folks are looking forward to downloading the RC and using it as we expand the downloads—we're looking forward to the participation and seeing the data that will help us validate the RC. We've talked about making sure that you are "in control" of Windows 7 and one of the ways that people are in control of their PC is to personalize the experience. With the RC you're going to see some of the new personalization "elements" in Windows 7. In this post, Denise Trabona and Samuel Moreau of our product design team provide a behind the scenes look at some of the work. Be sure to check out the links below the images as you can see a lot more work by these talented artists. Note, these are just thumbnails for this post so be sure to enjoy the full screen images in the RC. --Steven*

*PS: Just a reminder, that just as with the pre-beta and beta we'll be testing out Windows Update and the system for doing patches and updates. So along with new drivers you might also see some other updates flowing through the system.*

One of the most exciting parts of engineering Windows 7 has been the wide variety of work that gets done over the course of a full product cycle. As evidenced by the variety of topics just in this blog, one can see that we are hard at work at all levels of the product. For fun, we thought folks might enjoy hearing some of the story behind the new personalization work in Windows 7.

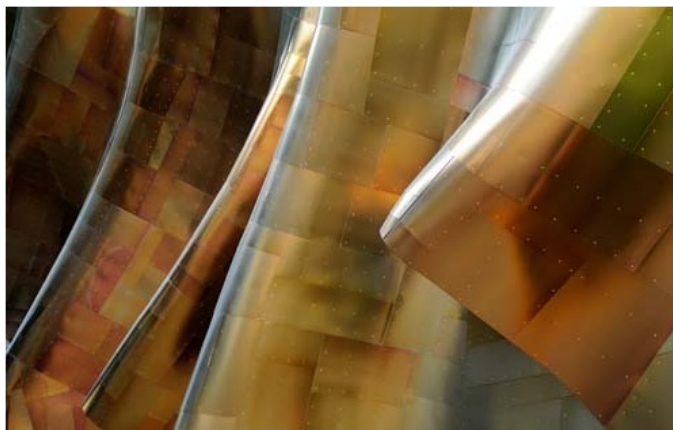
As some folks have noticed, we are unveiling some new personalization content (wallpapers, glass colors and sounds schemes) in the RC build which allows people greater flexibility to personalize their experience. One thing we know is that Windows users love to express themselves by changing the desktop background and like many past releases, Windows 7 includes content in the box that allows you to begin customizing your experience immediately.

## **A picture speaks a thousand words**

In developing the personalization features, we knew that we wanted great content for people to express their personal style. Because the desktop background is such a vibrant surface, we wanted to focus on providing quality content that demonstrated how creative people could be with this feature. When folks send us screenshots using the feedback button, we are regularly inspired by the rich diversity and personality of the wallpapers that people choose.

As we thought about how we wanted to approach personalization in Windows 7, we knew one way was to honor our lineage. In the past photography has been featured heavily Windows. Some of that photography has been quite beautiful and has become a proud tradition we wanted to maintain. In addition, we also wanted to explore new territory and expand our visual palette. In the realm of photography, we kept a theme focused on landscape photography which is our tradition, but added new themes for architecture and nature. Much of the this imagery is from our stock imagery partners, but we also had the good fortune to work with a talented local photographer named [Will Austin](#), who has photographed all over the world on many subjects with an emphasis on architecture. Will's photos provide a little bit of the local flavor of the Seattle area that we are proud to call home.





### **Raising the bar of inspiration and delight**

With the photography covered, we tried to broaden our coverage to include additional images that would inspire, delight and invigorate people's imaginations. We wanted to stretch into some new content that felt unique, timely, and with a distinct point of view. Our goal was content that balanced the timelessness of great photography with graphical illustrations that are energetic, modern, and fresh. On top of it all it was also important to achieve a rich variety in the illustrations to appeal to different tastes, genders and ages, color ranges from quiet to loud, and from large compositions to small and detailed.

Inspired by [our neighbors](#) in Zune, we worked with an agency called [72 and Sunny](#) to search for illustrators around the world to create one-of-a-kind art work for you to have in Windows 7. In the process of looking through tons of samples, we sought a group of artists whose styles seemed both incredibly varied, to cover the broad diversity we were after, and maintained a common thread that we felt was applicable to the overall tone we were striving to achieve. Then began the fun part, with little more than some simple guiding words (light, energetic, inspiring, optimistic, etc.), the artists went off with a blank canvas to create concept sketches of their original pieces.

### **Iterate and refine**

We still remember the first chance we got to review the artist's initial sketches and concept work, and right from that moment, we knew that these images were going to be a lot of fun. The next step was to iterate back and forth a few times to make sure certain goals were achieved and get little details just right. For example, a couple of things that were important to us were how the image flowed under the new task bar and striking the right balance between visually compelling, and not too distracting when it came to finding that important file on your desktop. It's tricky to find the right balance and we were fortunate to have an amazingly talented set of artists and our friends at 72 and Sunny to work with on this project.

**Windows is for the whole world**

Finally, we wanted to recognize the global audience of Windows by seeking out illustrators with varied backgrounds and styles with the intention of representing and appealing to people all around the world.

With that, we are honored to introduce the amazingly talented artists and the work that they contributed to Windows 7 personalization.



[Yuko Kondo](#)

From Japan, now resides in London, England



[Katharina Leuzinger](#)

Born to Swiss and Japanese parents in Zurich, Switzerland, Katharina Leuzinger now resides in London, England



[Osmand Nosse](#)

Wicklow, Ireland



[Klaus Haapaniemi](#)

From Finland, now based in London, England



[Chris Sickles of Red Nose Studios](#)  
Indiana, United States



[Punga](#)  
Buenos Aires, Argentina



[Pomme Chan](#)

Born and educated in Bangkok, Pomme Chan now resides in London, England.



[Kustaa Saksi](#)

Amsterdam, Netherlands



[Paul Hwang and Benjamin Lee of Nanosphere](#)  
Los Angeles, California



[Adhemas Batista](#)

From Sao Paulo Brazil, now resides in Los Angeles, California



[Kai and Sunny](#)  
London, England



[Nan Na Hvass](#)  
Born to a Danish father and Chinese mother in Swaziland, Africa, Nan Na Hvass now resides in Copenhagen, Denmark.

We hope this post has given you some insight into the Windows 7 content. We also hope that we achieved the goals we set out for ourselves with this element of Windows 7.

-Denise Trabona and Samuel Moreau

# Support and Q&A for Solid-State Drives

Steven Sinofsky | [2009-05-05T03:00:00+00:00](#)

---

*There's a lot of excitement around the potential for the widespread adoption of solid-state drives (SSD) for primary storage, particularly on laptops and also among many folks in the server world. As with any new technology, as it is introduced we often need to revisit the assumptions baked into the overall system (OS, device support, applications) as a result of the performance characteristics of the technologies in use. This post looks at the way we have tuned Windows 7 to the current generation of SSDs. This is a rapidly moving area and we expect that there will continue to be ways we will tune Windows and we also expect the technology to continue to evolve, perhaps introducing new tradeoffs or challenging other underlying assumptions. Michael Fortin authored this post with help from many folks across the storage and fundamentals teams. --Steven*

Many of today's Solid State Drives (SSDs) offer the promise of improved performance, more consistent responsiveness, increased battery life, superior ruggedness, quicker startup times, and noise and vibration reductions. With prices dropping precipitously, most analysts expect more and more PCs to be sold with SSDs in place of traditional rotating hard disk drives (HDDs).

In Windows 7, we've focused a number of our engineering efforts with SSD operating characteristics in mind. As a result, Windows 7's default behavior is to operate efficiently on SSDs without requiring any customer intervention. Before delving into how Windows 7's behavior is automatically tuned to work efficiently on SSDs, a brief overview of SSD operating characteristics is warranted.

## Random Reads: A very good story for SSDs

SSDs tend to be very fast for random reads. Most SSDs thoroughly trounce traditionally HDDs because the mechanical work required to position a rotating disk head isn't required. As a result, the better SSDs can perform 4 KB random reads almost 100 times faster than the typical HDD (about 1/10<sup>th</sup> of a millisecond per read vs. roughly 10 milliseconds).

## Sequential Reads and Writes: Also Good

Sequential read and write operations range between quite good to superb. Because flash chips can be configured in parallel and data spread across the chips, today's better SSDs can read sequentially at rates greater than 200 MB/s, which is close to double the rate many 7200 RPM drives can deliver. For sequential writes, we see some devices greatly exceeding the rates of typical HDDs, and most SSDs doing fairly well in comparison. In today's market, there are still considerable differences in sequential write rates between SSDs. Some greatly outperform the typical HDD, others lag by a bit, and a few are poor in comparison.

## Random Writes & Flushes: Your mileage will vary greatly

The differences in sequential write rates are interesting to note, but for most users they won't make for as notable a difference in overall performance as random writes.

What's a long time for a random write? Well, an average HDD can typically move 4 KB random writes to its spinning media in 7 to 15 milliseconds, which has proven to be largely unacceptable. As a result, most HDDs come with 4, 8 or more megabytes of internal memory and attempt to cache small random writes rather than wait the full 7 to 15 milliseconds. When they do cache a write, they return success to the OS even though the bytes haven't been moved to the spinning media. We typically see these cached writes completing in a few hundred *micro*seconds (so 10X, 20X or faster than actually writing to spinning media). In looking at millions of disk writes from thousands of telemetry traces, we observe 92% of 4 KB or smaller IOs taking less than 1 millisecond, 80% taking less than 600 microseconds, and an impressive 48% taking less than 200 microseconds. Caching works!



On occasion, we'll see HDDs struggle with bursts of random writes and flushes. Drives that cache too much for too long and then get caught with too much of a backlog of work to complete when a flush comes along, have proven to be problematic. These flushes and surrounding IOs can have considerably lengthened response times. We've seen some devices take a half second to a full second to complete individual IOs and take 10's of seconds to return to a more consistently responsive state. For the user, this can be awful to endure as responsiveness drops to painful levels. Think of it, the response time for a single I/O can range from 200 microseconds up to a whopping 1,000,000 microseconds (1 second).

When presented with realistic workloads, we see the worst of the SSDs producing very long IO times as well, as much as one half to one full second to complete individual random write and flush requests. This is abysmal for many workloads and can make the entire system feel choppy, unresponsive and sluggish.

## Random Writes & Flushes: Why is this so hard?

For many, the notion that a purely electronic SSD can have more trouble with random writes than a traditional HDD seems hard to comprehend at first. After all, SSDs don't need to seek and position a disk head above a track on a rotating disk, so why would random writes present such a daunting a challenge?

The answer to this takes quite a bit of explaining, Anand's [article](#) admirably covers many of the details. We highly encourage motivated folks to take the time to read it as well as this fine [USENIX paper](#). In an attempt to avoid covering too much of the same material, we'll just make a handful of points.

- *Most SSDs are comprised of flash cells (either [SLC](#) or [MLC](#)).* It is possible to build SSDs out of DRAM. These can be extremely fast, but also very costly and power hungry. Since these are relatively rare, we'll focus our discussion on the much more popular NAND flash based SSDs. Future SSDs may take advantage of other nonvolatile memory technologies than flash.
- *A flash cell is really a trap, a trap for electrons and electrons don't like to be trapped.* Consider this, if placing 100 electrons in a flash cell constitutes a bit value of 0, and fewer means the value is 1, then the controller logic may have to consider 80 to 120 as the acceptable range for a bit value of 0. A range is necessary because some electrons may escape the trap, others may fall into the trap when attempting to fill nearby cells, etc... As a result, some very sophisticated error correction logic is needed to insure data integrity.
- *Flash chips tend to be organized in complex arrangements, such as blocks, dies, planes and packages.* The size, arrangement, parallelism, wear, interconnects and transfer speed characteristics of which can and do vary greatly.
- *Flash cells need to be erased before they can be written.* You simply can't trust that a flash cell has no residual electrons in it before use, so cells need to be erased before filling with electrons. Erasing is done on a large scale. You don't erase a cell; rather you erase a large block of cells (like 128 KB worth). Erase times are typically long -- a millisecond or more.
- *Flash wears out.* At some point, a flash cell simply stops working as a trap for electrons. If frequently updated data (e.g., a file system log file) was always stored in the same cells, those cells would wear out more quickly than cells containing read-mostly data. Wear leveling logic is employed by flash controller firmware to spread out writes across a device's full set of cells. If done properly, most devices will last years under normal desktop/laptop workloads.
- *It takes some pretty clever device physicists and some solid engineering to trap electrons at high speed, to do so without errors, and to keep the devices from wearing out unevenly.* Not all SSD manufacturers are as far along as others in figuring out how to do this well.

## Performance Degradation Over Time, Wear, and Trim

As mentioned above, flash blocks and cells need to be erased before new bytes can be written to them. As a result, newly purchased devices (with all flash blocks pre-erased) can perform notably better at purchase time than after considerable use. While we've observed this performance degradation ourselves, we do not consider this to be a show stopper. In fact, except via benchmarking measurements, we don't expect users to notice the drop during normal use.

Of course, device manufacturers and Microsoft want to maintain superior performance characteristics as best we can. One can easily imagine the better SSD manufacturers attempting to overcome the aging issues by pre-erasing blocks so the performance penalty is largely unrealized during normal use, or by maintaining a large enough spare area to store short bursts of writes. SSD drives designed for the enterprise may have as high as 50% of their space reserved in order to provide lengthy periods of high sustained write performance.

In addition to the above, Microsoft and SSD manufacturers are adopting the Trim operation. In Windows 7, if an SSD reports it supports the Trim attribute of the ATA protocol's Data Set Management command, the NTFS file system will request the ATA driver to issue the new operation to the device when files are deleted and it is safe to erase the SSD pages backing the files. With this information, an SSD can plan to erase the relevant blocks opportunistically (and lazily) in the hope that subsequent writes will not require a blocking erase operation since erased pages are available for reuse.

As an added benefit, the Trim operation can help SSDs reduce wear by eliminating the need for many merge operations to occur. As an example, consider a single 128 KB SSD block that contained a 128 KB file. If the file is deleted and a Trim operation is requested, then the SSD can avoid having to mix bytes from the SSD block with any other bytes that are subsequently written to that block. This reduces wear.

Windows 7 requests the Trim operation for more than just file delete operations. The Trim operation is fully integrated with partition- and volume-level commands like Format and Delete, with file system commands relating to truncate and compression, and with the System Restore (aka Volume Snapshot) feature.

## **Windows 7 Optimizations and Default Behavior Summary**

As noted above, all of today's SSDs have considerable work to do when presented with disk writes and disk flushes. Windows 7 tends to perform well on today's SSDs, in part, because we made many engineering changes to reduce the frequency of writes and flushes. This benefits traditional HDDs as well, but is particularly helpful on today's SSDs.

Windows 7 will disable disk defragmentation on SSD system drives. Because SSDs perform extremely well on random read operations, defragmenting files isn't helpful enough to warrant the added disk writing defragmentation produces. The FAQ section below has some additional details.

By default, Windows 7 will disable Superfetch, ReadyBoost, as well as boot and application launch prefetching on SSDs with good random read, random write and flush performance. These technologies were all designed to improve performance on traditional HDDs, where random read performance could easily be a major bottleneck. See the FAQ section for more details.

Since SSDs tend to perform at their best when the operating system's partitions are created with the SSD's alignment needs in mind, all of the partition-creating tools in Windows 7 place newly created partitions with the appropriate alignment.

## **Frequently Asked Questions**

Before addressing some frequently asked questions, we'd like to remind everyone that we believe the future of SSDs in mobile and desktop PCs (as well as enterprise servers) looks very bright to us. SSDs can deliver on the promise of improved performance, more consistent responsiveness, increased battery life, superior ruggedness, quicker startup times, and noise and vibration reductions. With prices steadily dropping and quality on the rise, we expect more and more PCs to be sold with SSDs in place of traditional rotating HDDs. With that in mind, we focused an appropriate amount of our engineering efforts towards insuring Windows 7 users have great experiences on SSDs.

### **Will Windows 7 support Trim?**

Yes. See the above section for details.

**Will disk defragmentation be disabled by default on SSDs?**

Yes. The automatic scheduling of defragmentation will exclude partitions on devices that declare themselves as SSDs. Additionally, if the system disk has random read performance characteristics above the threshold of 8 MB/sec, then it too will be excluded. The threshold was determined by internal analysis.

The random read threshold test was added to the final product to address the fact that few SSDs on the market today properly identify themselves as SSDs. 8 MB/sec is a relatively conservative rate. While none of our tested HDDs could approach 8 MB/sec, all of our tested SSDs exceeded that threshold. SSD performance ranged between 11 MB/sec and 130 MB/sec. Of the 182 HDDs tested, only 6 configurations managed to exceed 2 MB/sec on our random read test. The other 176 ranged between 0.8 MB/sec and 1.6 MB/sec.

**Will Superfetch be disabled on SSDs?**

Yes, for most systems with SSDs.

If the system disk is an SSD, and the SSD performs adequately on random reads and doesn't have glaring performance issues with random writes or flushes, then Superfetch, boot prefetching, application launch prefetching, ReadyBoost and ReadDrive will all be disabled.

Initially, we had configured all of these features to be off on all SSDs, but we encountered sizable performance regressions on some systems. In root causing those regressions, we found that some first generation SSDs had severe enough random write and flush problems that ultimately lead to disk reads being blocked for long periods of time. With Superfetch and other prefetching re-enabled, performance on key scenarios was markedly improved.

**Is NTFS Compression of Files and Directories recommended on SSDs?**

Compressing files help save space, but the effort of compressing and decompressing requires extra CPU cycles and therefore power on mobile systems. That said, for infrequently modified directories and files, compression is a fine way to conserve valuable SSD space and can be a good tradeoff if space is truly a premium.

We do not, however, recommend compressing files or directories that will be written to with great frequency. Your Documents directory and files are likely to be fine, but temporary internet directories or mail folder directories aren't such a good idea because they get large number of file writes in bursts.

**Does the Windows Search Indexer operate differently on SSDs?**

No.

**Is Bitlocker's encryption process optimized to work on SSDs?**

Yes, on NTFS. When Bitlocker is first configured on a partition, the entire partition is read, encrypted and written back out. As this is done, the NTFS file system will issue Trim commands to help the SSD optimize its behavior.

We do encourage users concerned about their data privacy and protection to enable Bitlocker on their drives, including SSDs.

#### **Does Media Center do anything special when configured on SSDs?**

No. While SSDs do have advantages over traditional HDDs, SSDs are more costly per GB than their HDD counterparts. For most users, a HDD optimized for media recording is a better choice, as media recording and playback workloads are largely sequential in nature.

#### **Does Write Caching make sense on SSDs and does Windows 7 do anything special if an SSD supports write caching?**

Some SSD manufacturers including RAM in their devices for more than just their control logic; they are mimicking the behavior of traditional disks by caching writes, and possibly reads. For devices that do cache writes in volatile memory, Windows 7 expects flush commands and write-ordering to be preserved to at least the same degree as traditional rotating disks. Additionally, Windows 7 expects user settings that disable write caching to be honored by write caching SSDs just as they are on traditional disks.

#### **Do RAID configurations make sense with SSDs?**

Yes. The reliability and performance benefits one can obtain via HDD RAID configurations can be had with SSD RAID configurations.

#### **Should the pagefile be placed on SSDs?**

Yes. Most pagefile operations are small random reads or larger sequential writes, both of which are types of operations that SSDs handle well.

In looking at telemetry data from thousands of traces and focusing on pagefile reads and writes, we find that

- Pagefile.sys reads outnumber pagefile.sys writes by about 40 to 1,
- Pagefile.sys read sizes are typically quite small, with 67% less than or equal to 4 KB, and 88% less than 16 KB.
- Pagefile.sys writes are relatively large, with 62% greater than or equal to 128 KB and 45% being exactly 1 MB in size.

In fact, given typical pagefile reference patterns and the favorable performance characteristics SSDs have on those patterns, there are few files better than the pagefile to place on an SSD.

#### **Are there any concerns regarding the Hibernate file and SSDs?**

No, hiberfile.sys is written to and read from sequentially and in large chunks, and thus can be placed on either HDDs or SSDs.

## **What Windows Experience Index changes were made to address SSD performance characteristics?**

In Windows 7, there are new random read, random write and flush assessments. Better SSDs can score above 6.5 all the way to 7.9. To be included in that range, an SSD has to have outstanding random read rates and be resilient to flush and random write workloads.

In the Beta timeframe of Windows 7, there was a capping of scores at 1.9, 2.9 or the like if a disk (SSD or HDD) didn't perform adequately when confronted with our random write and flush assessments. Feedback on this was pretty consistent, with most feeling the level of capping to be excessive. As a result, we now simply restrict SSDs with performance issues from joining the newly added 6.0+ and 7.0+ ranges. SSDs that are not solid performers across all assessments effectively get scored in a manner similar to what they would have been in Windows Vista, gaining no Win7 boost for great random read performance.

# Our Next Engineering Milestone

Steven Sinofsky | [2009-05-11T13:01:00+00:00](#)

---

Back in [January](#) we released the Beta and updated you on our overall engineering process that will get us from Beta to the Release Candidate. Today, downloading of the Release Candidate started and we're already seeing a lot of installations and a lot of excitement. On behalf of the team, I want to extend a thank you for all of the millions of people who have been running and testing the Beta who have helped to make the Release Candidate possible. The feedback we have received, through all the mechanisms we have blogged about, has been an incredibly valuable part of Engineering Windows 7. We continue to be humbled by the response to Windows 7. Thank you!

This post is about the path from RC to what we call RTM, release to manufacturing. RTM is not one point in time but a "process" as from RTM we enable the PC manufacturers to begin their processes of building Windows 7 images for new PCs, readying downloads for existing machines, and preparing the full supply chain to deliver Windows 7 to customers. Thus RTM is the final stage in our engineering of Windows 7, but the engineering continues from RTM until you can purchase Windows 7 and Windows 7 PCs in stores at General Availability, or GA.

The path to RTM starts with downloads of the RC. The RC is "done" and what we are doing is validating this against the breadth of the ecosystem and with partners. It means, from our perspective, we have run many tests many times and are working to understand the quality of the release in a breadth sense. We're all familiar with this as we have done this same thing as we went from pre-Beta to Beta and from Beta to RC. The primary difference with the RC is that we will not be changing the functionality or features of the product at this point—that's the sort of thing we'll save for a future release. We've gotten tons of feedback on design and features and shown how we have digested and acted on this feedback throughout many posts on this blog. We know we did not do everything that was asked, and we have also seen that we've been asked to do things that are tricky to reconcile. We hoped through the dialog on this blog that we've shown our commitment to listening and balancing a wide variety of inputs, and how we have thought about the evolution of Windows.

What sort of feedback are we looking for in the RC? We are primarily focused on monitoring the behavior of the product through the telemetry, and of course making sure we did not introduce any regressions in any dimension from Beta quality. One of the things we have done since Beta has continued to beef up telemetry—we've put in additional monitoring points in many systems. We're particularly interested in seeing what devices are installed, drivers that are required, and overall system performance. We have telemetry points that monitor the UI responsiveness of the Start Menu, Internet Explorer (recently [posted](#)), Boot, Shutdown, Resume, and across all subsystems. Of course in the final product, this telemetry is optional and opt-in, and it is always private.

There are a series of specific types of reports that we are keeping an eye out for that would constitute changes we would make to the code between now and RTM. Some of these might include:

- **Installation** – We have significant telemetry in the setup process and also significant logging. Of course if you can't set up at all that is something we are interested in and the same holds for upgrades from Windows Vista. For the "enrolled" beta programs we have a mechanism to enlist a connection to Microsoft for these issues and for the broad community the public support groups are monitored.
- **Security issues** – Obviously any vulnerability is a potential for something we would fix. We will use the same criteria to address these issues as we would for any in-market product.
- **Crashes and Hangs** – We are monitoring the "crash" reports for issues that arise that impact broad sets of people. These could be Windows code, drivers, or third party software. This information streams "real time" to Microsoft and we watch it very carefully.
- **Device installation and compatibility** – When you download a driver from Windows Update or install a driver via a manufacturer's setup program this is a data point we collect. We've had millions of unique PnP IDs through the Beta. We also receive the IDs for devices that failed to locate drivers. We are constantly updating this web service with pointers to information about the device (driver availability, instructions, etc.)
- **Software installation** – Similar to devices, we are also monitoring the installation process of software and noting programs that do not complete successfully. Again we have the mechanism to help move that forward and/or introduce compatibility work in the RTM milestone.
- **Servicing** – We will continue to test the servicing of Windows 7 so everyone should expect updates to be made available via Windows Update. This includes new drivers and will also include patches to Windows 7. Test Updates will be labeled as such. We might also fix

any significant issue with new code as well. All of this in an effort to validate the servicing pipeline and to maintain the quality of the RC.

- **New Hardware** – Perhaps the most important category is making sure that we work with all the new hardware being made as we all use 7100. Our PC Manufacturing partners and Hardware partners are engineering new PCs and these are combinations new to the market and new to the OS. We're working together to make sure Windows 7 has great support for these PCs and hardware.

All of the feedback will be evaluated and whether the issue is with Windows itself or with hardware, software, or OEM partner code we will work closely across the entire ecosystem to do what is necessary to deliver excellent fully integrated PCs. This goal is more important than anything else at this point. The depth of this work is new for the team in terms of spending engineer to engineer time across a broad range of partners to make sure everyone is ready together to deliver a great PC experience.

Overall, while many have said that the quality of the Beta was on par with past RCs (remember how some even suggested we release it as final!), we are working to do an even better job with Windows 7. We think we have the tools in place to do that.

While the RC itself was compiled about 2 weeks ago, it takes a bit of time to go through the mechanics of validating all the ISOs and images that are released. In the meantime we continue doing daily builds of the product. The daily builds are incorporating code changes to address the above types of issues that impact enough customers that on balance the code change is more valuable than the potential of a regression. 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.

The RTM milestone is not a date, but a process. As that process concludes, we are done changing the code and are officially "servicing" Windows 7. That means any subsequent changes are delivered as fixes (KB articles) or banked for the first service pack. Obviously our ability to deliver fixes via Windows Update has substantially changed the way we RTM and so it is not unreasonable to expect updates soon after the product is complete as we have done for both Windows XP and Windows Vista.

Between now and the RTM milestone we will make changes to the code in response to the above inputs. We are decelerating and will do so "gracefully" and not abruptly. We do not have a "deadline" we are aiming to meet and the quality (in all dimensions) of the product and a smooth finish are the most important criteria for Windows 7. In addition, we have a lot of work going on behind the scenes to build Windows 7 in nearly 100 languages around the world and to make sure all the supporting materials such as our Windows web site, SDK, resource kits, and so on are ready and available in a timely manner.

Once we have entered the RTM phase, our partners will begin to make their final images and manufacture PCs, and hardware and software vendors will ready their Windows 7 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 and based on the RTM process the volume license products will be available as well.

Delivering the highest quality Windows 7 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. Many are pushing us to release the product sooner rather than later, but our focus remains on a high quality release.

Ultimately our partners will determine when their PCs are available in market. If the feedback and telemetry on Windows 7 match our expectations then we will enter the final phases of the RTM process in about 3 months. If we are successful in that, then we tracking to our shared goal of having PCs with Windows 7 available this Holiday season.

--Steven and Jon

# Media Streaming with Windows 7

Steven Sinofsky | [2009-05-12T03:00:00+00:00](#)

---

*We've blogged about a number of features related to home networking and media in Windows 7. A scenario which brings all these together in a pretty cool way is Media Streaming. This scenario allows you to use a Windows 7 PC as a hub for media sharing—where you can share media with other PCs and devices on your home network via streaming, and even stream this information securely over the internet. Scott Manchester on the Devices & Media program management team coordinated this post, but as you will see it represents work across the Core User Experience, Media Center, Networking, and even Windows Live chose to take advantage of the new APIs in this scenario. This is a pretty detailed post and there's a lot to try out. Those of you using the RC to test things out, you can always install on another PC and use it for the 30-day period without requiring a new PID key. Have fun! --Steven*

Windows 7 includes a number of exciting new media streaming features that enable you to enjoy your media collection on other PCs and devices in the home and while on the road from across the internet. We've created a networked media experience that is more friendly to use and simpler to set up. Now enjoying music, pictures, and video on your network connected PC or media device "just works" without concern for media formats, transports, or protocols.

There are a growing number of Network Media Devices (NMDs) certified to interoperate using an open and widely embraced industry standard called the Digital Living Network Alliance (DLNA). Windows 7 implements this open standard, which means that sharing media between NMDs, Windows PCs, Windows Home Server, and Extenders for Windows Media Center (including Xbox 360) is easier and more natural. Supporting this standard also means that the myriad of NMDs such as electronic picture frames, network radios, televisions, and others are companions to Windows 7 PCs and may seamlessly participate in the whole-home media experience.

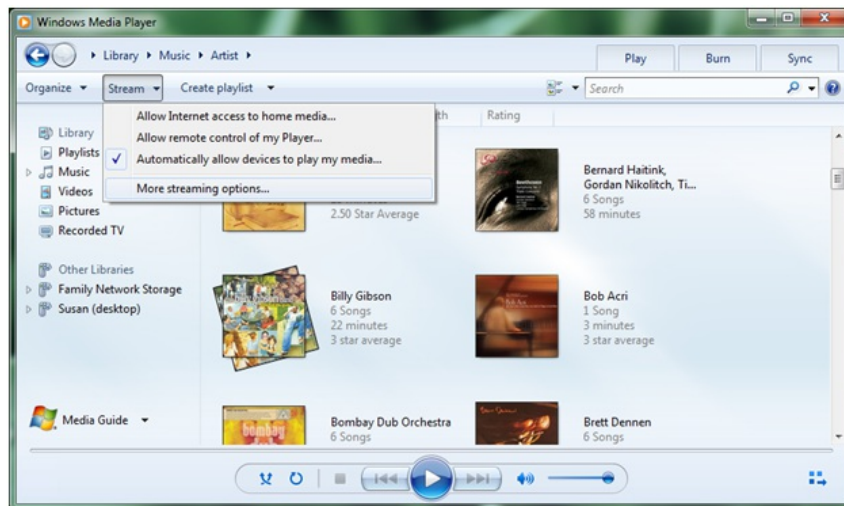
## Not Just for the Techie

We made it much simpler to configure media streaming. Before Windows 7, media streaming features were focused on media enthusiasts. To improve the setup experience, media streaming has been integrated with the new [HomeGroup](#) feature so in a typical home network configuration, media streaming is enabled and works by default. There is also a new "Stream" menu prominently displayed in the Window Media Player user interface (see figure below) that exposes simple scenario-based configuration options. These options allow you to:

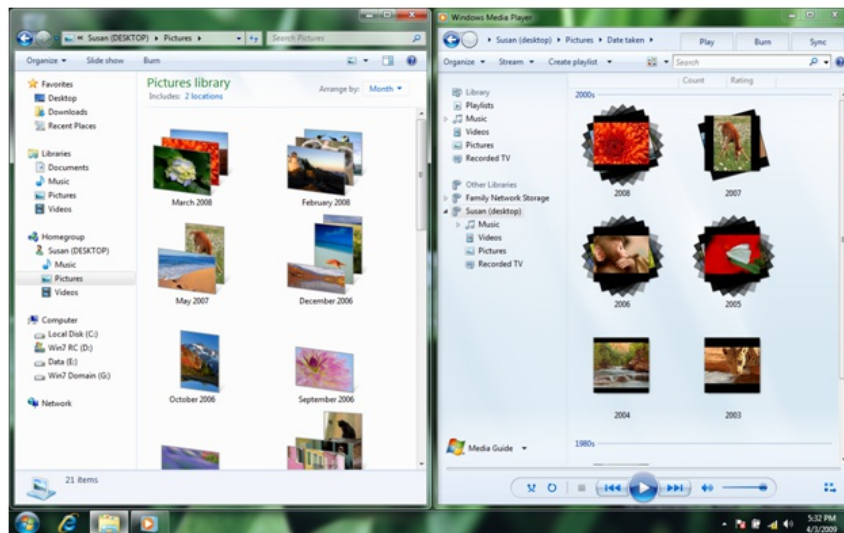
1. Set up your home PC so you can access your media libraries while away from home
2. Allow other Windows 7 PCs and devices to push media to your Player and control it
3. Quickly authorize all home PCs and devices to access your media collection

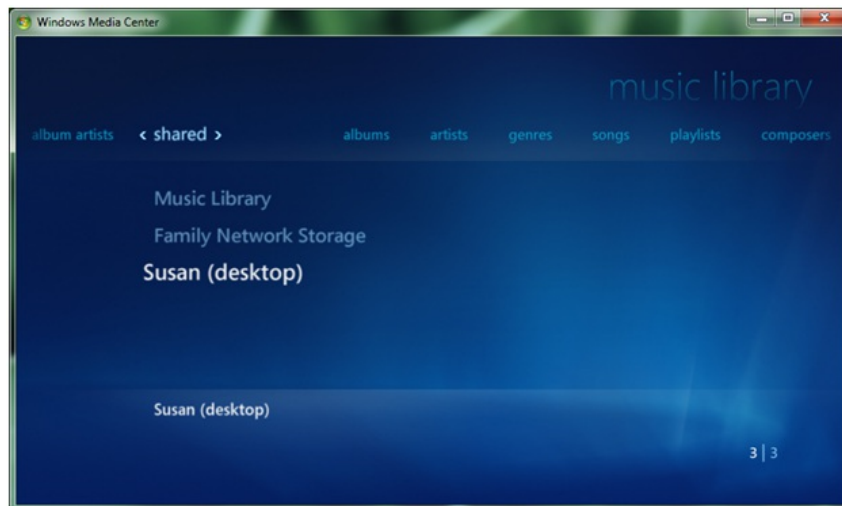
Each of these scenarios will be discussed throughout this post.





HomeGroup introduces the concept of “[shared libraries](#)” for music, pictures, and video. As described in a [previous blog post](#), these shared libraries are accessible from within the navigation pane of Windows Explorer and Windows Media Player, and from the “shared” view of each media category within Windows Media Center (see figures below). The scope of these libraries is the same from each of these views.



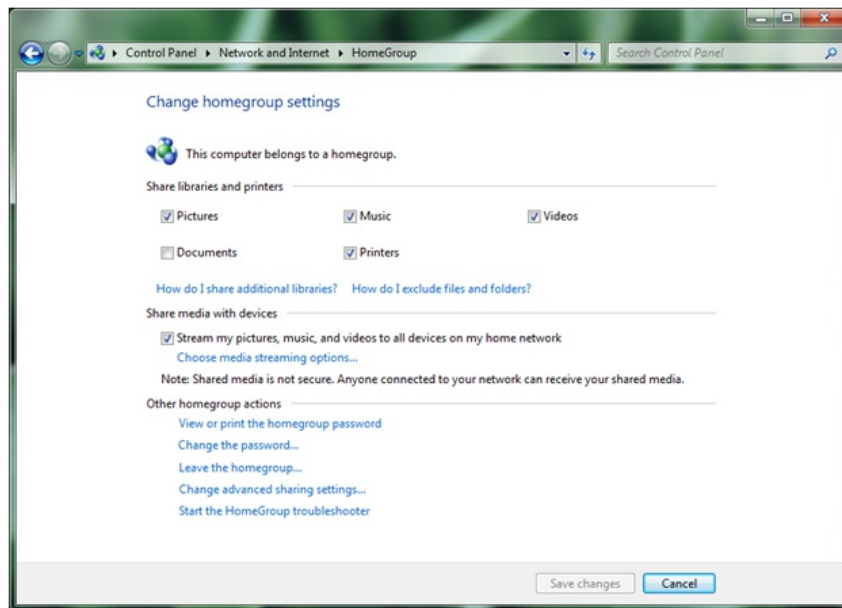


Windows Explorer will automatically discover and provide access to shared media libraries on other HomeGroup PCs. In addition, Windows Media Player and Windows Media Center will automatically discover shared libraries from:

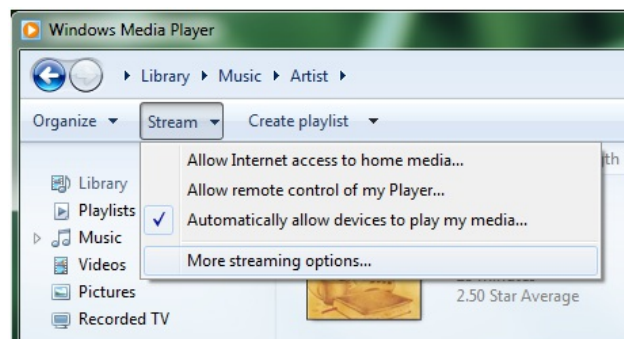
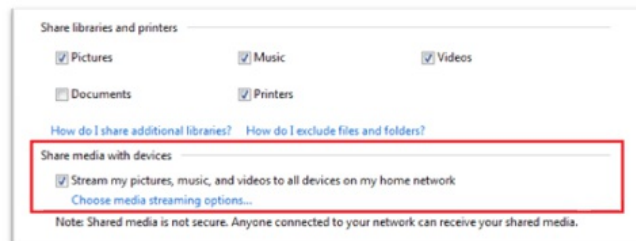
1. Windows Media Player 11 and 12
2. Windows Home Server
3. All DLNA compliant media servers (e.g. network attached storage)

### **Who Can Access My Shared Media Libraries?**

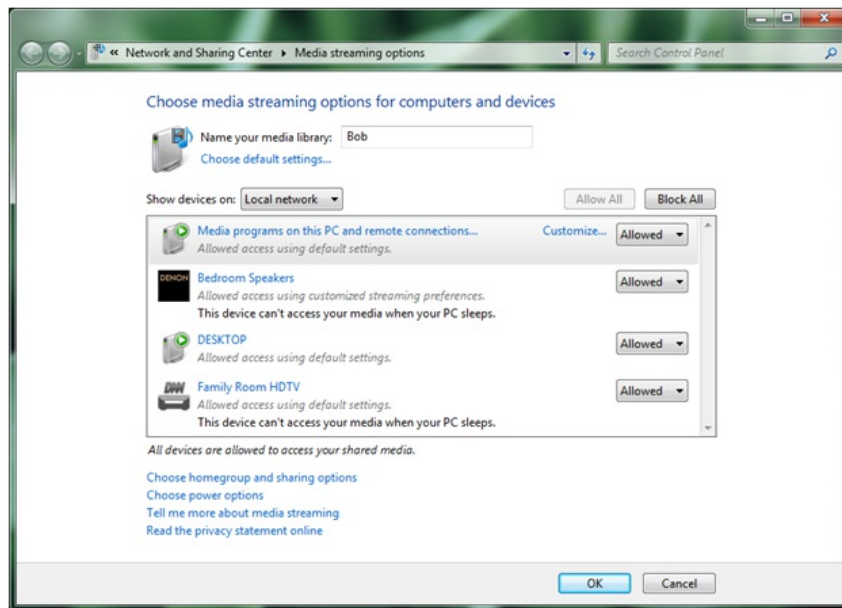
A HomeGroup is a secured set of Windows 7 PCs that can view and consume each other's media seamlessly. Sharing is automatically set up among HomeGroup PCs and HomeGroup settings allow you to choose what types of media you would like to share; for example, you may choose to only share your music library and not your video or pictures.



In addition to all HomeGroup PCs being able to access your media, we made it easy to allow devices to access shared media libraries on Windows 7 PCs. This can be done conveniently from either HomeGroup settings or within Windows Media Player:

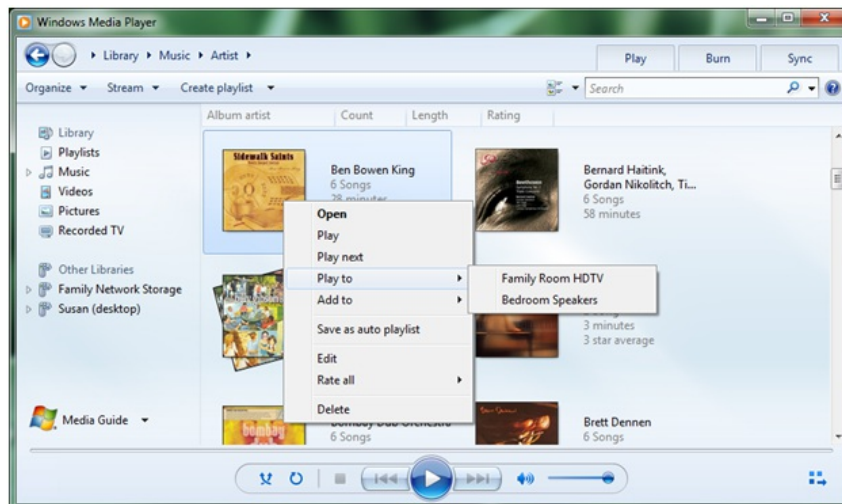


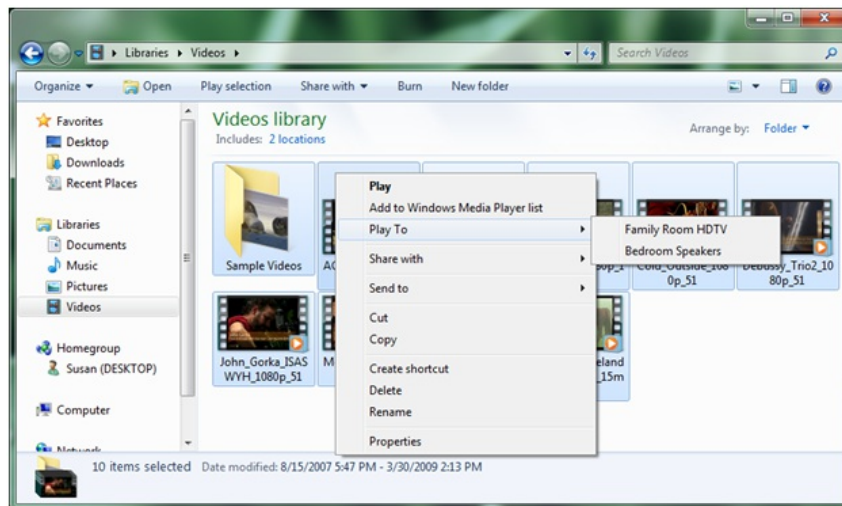
You can also choose to restrict which specific PCs or devices have access to your media by choosing "more streaming options..." from the Windows Media Player "Stream" menu.



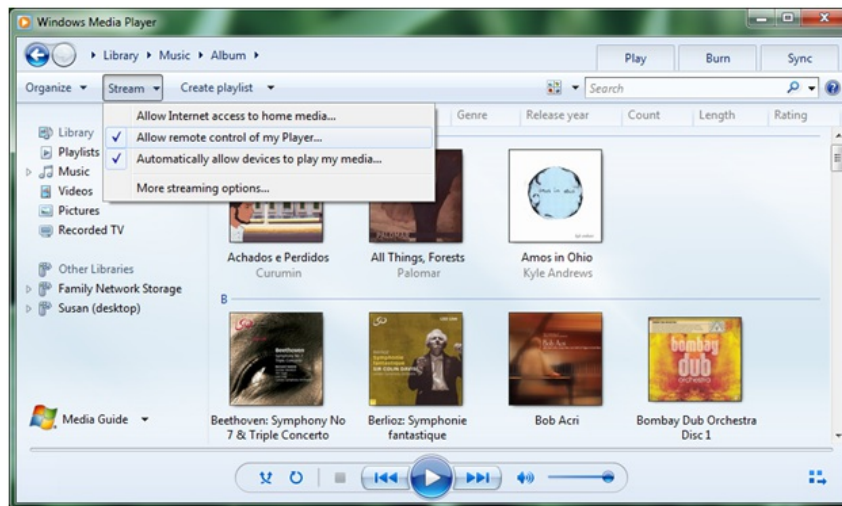
## Play To: Windows 7 as a Universal Remote Control for your Media Collection

In addition to playing media streamed from other shared media libraries within Windows Media Player, Windows 7 can now send media to be played on other Windows 7 PCs and DLNA-certified digital media renderers. We call this feature “Play To.” With “Play To,” you can browse or search from within Windows Media Player or Windows Explorer to find your desired media, and then choose where you want it to be played. A versatile remote control window is presented for each “Play To” session, providing you with the ability to control the entire experience.





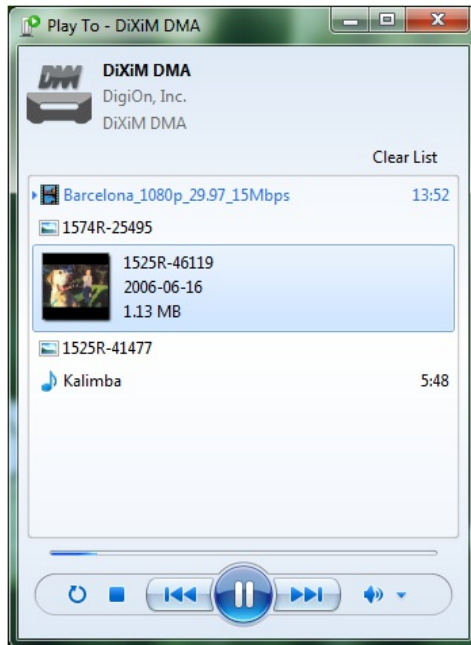
It does not matter where media collections are stored. “Play To” is available for both local media libraries and for shared media libraries. If you would like to send media from one Windows 7 PC to another, choose “Allow remote control of my Player” from the Windows Media Player “Stream” menu on the receiving PC. This will cause Windows Media Player to be discovered in the “Play To” menu of other Windows 7 PCs on the same network.



When media streaming is enabled on your Windows 7 PC, “Play To” will be available in Windows Media Player and Windows Explorer via the right click menu for media items. If Windows 7 has not discovered a “Play To” capable PC or device on the network, this context menu will not be available. DLNA provides guidelines to certify different device categories and roles. Not every DLNA-certified device supports the “Play To” feature. Look for DLNA-certified Digital Media Renderers (DMR), and for the best performance, look for DMR devices that carry the “Compatible with Windows 7” logo.



Once you've selected media items to play on another PC or device, a "Play To" remote control window will launch providing standard controls like play, pause, stop, skip forward and backward, seek forward and backward, volume, and mute. Not every device will support all of the control features and some media types may not support seek. Once the "Play To" remote control window is launched, you can reorder or delete items, add to the queue, or toggle repeat. It's even possible to add new media items from Windows Media Player or Windows Explorer by dragging them into this window.

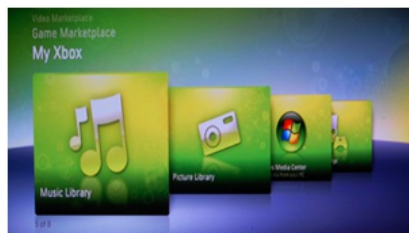


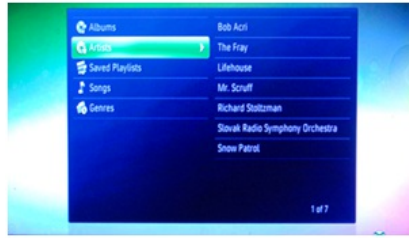
There is no artificial limit to the number of "Play To" sessions you can launch. You may send pictures to a picture frame, video clips to a TV, and music to another Windows 7 laptop all at the same time. Furthermore, different types of media can be sent to a single destination, as shown in the example above.

### What About the Xbox 360 and Extenders for Windows Media Center?

Xbox 360 has two ways to receive media streams from other Windows 7 PCs, which we refer to casually as "dashboard" mode and "extender" mode.

In dashboard mode, Xbox 360 functions in the role of a simple media player. While it's not officially a DLNA-certified device, you can use Xbox 360 to browse the shared media libraries from Windows 7 PCs (there is also support for this in Windows Media Player 11) and *pull* content from those libraries for playback within the dashboard.





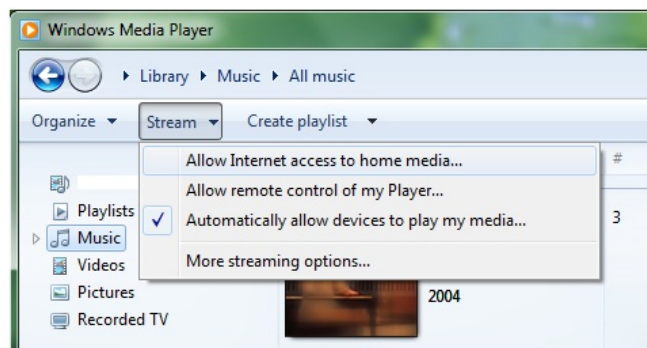
In extender mode, Xbox 360 (and other Extenders for Windows Media Center) is seen by Windows 7 PC's on the network as both a Digital Media Player (DMP) and a Digital Media Renderer (DMR) device. Using the Extender for Windows Media Center on the Xbox 360, you can browse media libraries on other computers and pull that content for local playback, similar to the process of using Xbox 360 in dashboard mode. However, in extender mode Xbox 360 will also support "Play To" so that users of Windows 7 PC's on the network can *push* content to it. All extenders, when associated with a Windows 7 PC, will be discovered in the "Play To" menu of other Windows 7 PCs.

## Internet Access to Home Media

With Windows 7 we've also extended the media streaming experience outside the home and allow you to access your home media from anywhere in the world via the internet. We've made media streaming over the internet a natural extension of the experience within the home. For the experience to be seamless we needed to solve some significant technical challenges, such as:

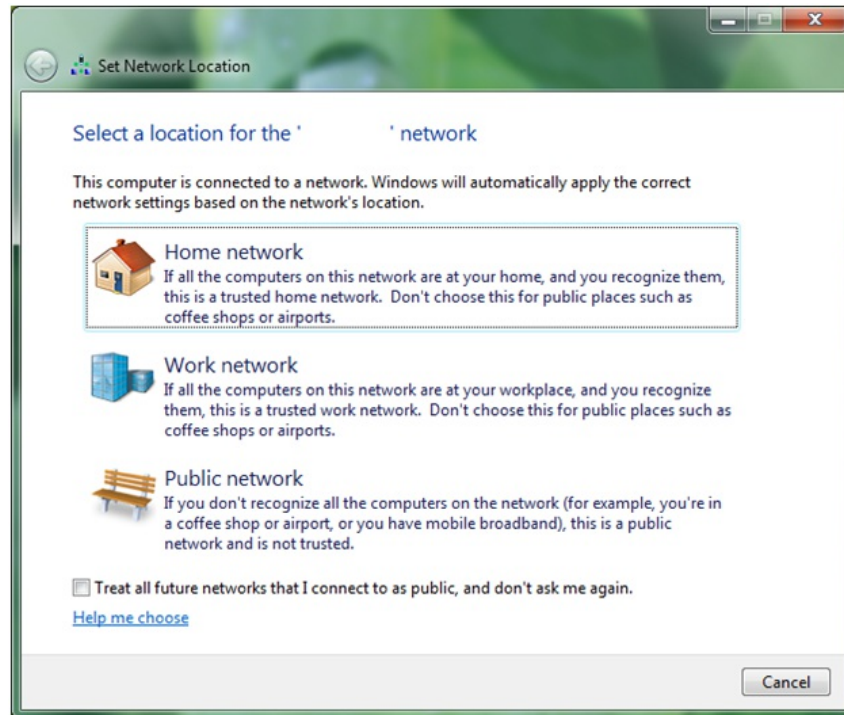
1. **Discovery** – Resolving the computer name at home to a routable IP address
2. **Privacy** – Ensuring the home media is only accessible by authorized users
3. **Security** – Encrypting browsing and streaming of media to prevent eavesdropping
4. **Reliability** – Network connection speeds, media formats and bit rates, and router firewalls all create potential reliability issues for a seamless experience

To overcome these technical hurdles, we designed a model that uses an Online ID Provider to help facilitate discovery, privacy, and security. The new Online ID Provider infrastructure in Windows 7 allows you to link your Online ID (e.g. you@live.com) with your Windows user account. This enables an authentication/authorization server to provide the necessary privacy to establish a protected link between two Windows 7 PCs (e.g. your laptop on the road and your PC at home). Internet access to home media is enabled from the "Stream" menu in Windows Media Player.



The setup process walks you through linking an online ID with your Windows user account, which must be performed on both the home PC and remote PC. The same online ID must be used on both PCs in order to establish the connection between them. In order for remote PCs to access the home media collection, the PC at home (acting as a server) must be on a "Home" network location. Remote PCs (acting as clients) can

browse and receive content streamed from the home PC from any network location (Public, Work, or Home). The network location is chosen when first connecting to any network and can be changed later from the Network and Sharing Center.



## Reliability - Network Connection Requirements

Streaming media over the internet from home works best with an “always on” broadband connection. Broadband uplink speeds vary from a modest 200Kbps to 10Mbps or more. Downlink connection speeds will also vary from crowded hotspots, hotel rooms, and wireless network connections in friends’ homes. Regardless of the uplink or downlink speeds, we wanted to ensure that even high bit rate content (e.g. high definition recorded TV) could be streamed with a good experience. The internet media streaming feature uses advanced bandwidth detection algorithms and end-to-end network heuristics to determine how to stream content that is at a higher bit rate than the smallest link in the network path.

Another challenge with internet access to home media is creating a peer-to-peer connection between the remote client PC and the home PC serving the media. A typical home network will get a single unique IP address from an internet service provider, and this IP address is shared by all the devices and PCs in the home using Network Address Translation (NAT), a function of an Internet Gateway Device (IGD) or Wireless Router. This creates a challenge for a remote PC or device to make an unsolicited connection inside the home, both in terms of resolving the home’s unique IP address and traversing the NAT to communicate directly to a unique PC or device on the home network.

Windows 7 employs some advanced NAT traversal technologies to establish the peer-to-peer connection and, with most IGDs, will allow a reliable connection to the home PC from any remote PC. For best results you should use a wireless router or IGD that has been certified by the Windows Logo program.

## Media Formats

In Windows 7 we let you enjoy the media you want and don’t trouble you with the need to know about file types or codecs in most cases. (For more details, see Table 1 below). In addition to supporting local playback of new formats, we can also ensure that the content will play on devices



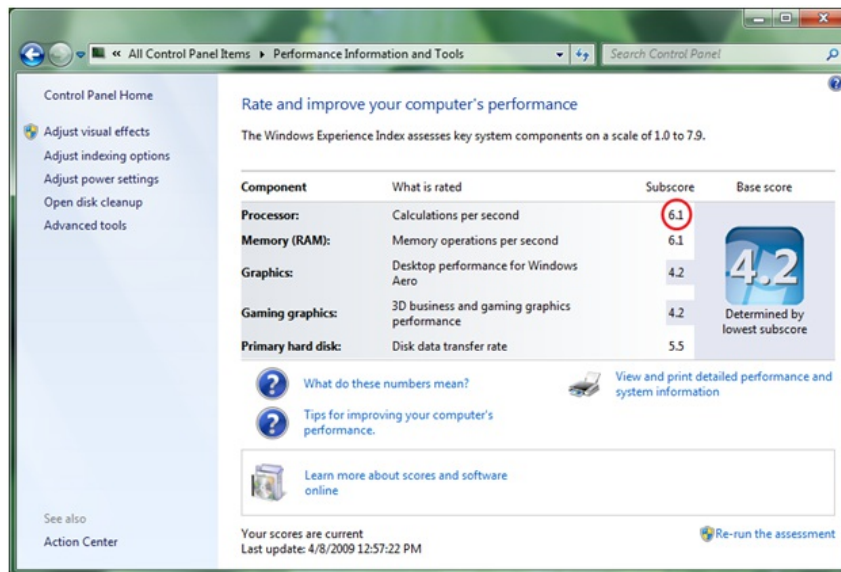
that may not support the codec, bit rate, container, or format of that content. We accomplish this by using the new transcoding support in Windows 7.

Let's say for instance you have a DivX movie you want to watch on your new DLNA certified television which only supports WMV and MPEG2. Windows 7 will determine the capability of the TV (codec, bit rate, etc.) and dynamically convert the DivX video to a format the TV can play. The general rule of thumb is: if Windows Media Player can play the content on the PC then the content will almost always play back on the network connected device. Bandwidth estimation techniques are used for media streaming within the home and over the internet, which enables Windows 7 to transcode using the most optimal format and bit rate.

Generic Format Name	File Extensions	Container	Video Decoders	Audio Decoders	Known limitations
MPEG-4	.mp4 (A, V, A+V) .m4a (A), .mov	ISO MPEG-4, AVI	H.264, MPEG-4 ASP and SP	AAC, MP3	Plays almost all industry standard files; can't play iTunes files that are protected with FairPlay; plays most .mov files from cameras but doesn't play embedded playlists or segmented files – used for most .mov movie trailers
3GPP/3GPP2	.3gp, .3g2 (A, V, A+V)	3GP	H.264, MPEG-4 SP	AAC	While we can play some of these files, most mobile phone video cameras produce files that contain ACELP or AMR audio and H.263 video which Windows 7 doesn't play
AAC	.aac (A)	ADTS		AAC	
ASP in AVI compatible with DivX 4-6 video format, XviD, and 3ivx	.avi (V, A+V)	AVI	MPEG-4 ASP	MP3, MS ADPCM	No support for MKV, special DivX sub-title mechanisms, or files protected with DivX proprietary DRM
AVCHD	.m2t, .m2ts, .mts (A, V, A+V)	MPEG-2 TS	H.264	Dolby Digital, LPCM	Some cameras use proprietary mechanisms to cut AVCHD into chunks which are not automatically reassembled when transferred to the PC
HDV	.m2t, .m2ts, .mts (A, V, A+V)	MPEG-2 TS	MPEG-2	MPEG-1 L2	

**Table 1: New Decoders in Windows 7**

The format and bit rate chosen for transcoding, especially for video, is highly dependent on the CPU performance of the transcoding PC as identified by its Windows Experience Index:



We also created a flexible model for silicon partners to provide hardware accelerators that automatically work with media streaming and other Windows 7 features. This new acceleration model allows hardware developers to build media foundation proxies for media format encoders and decoders that are fully implemented in their hardware (perhaps in a GPU or additional hardware device). With hardware supported encoding and decoding, Windows 7 can offload the computationally demanding transcoding to dedicated hardware as a background task without affecting the CPU performance of the PC.

## Digital Living Network Support in Windows 7

The Digital Living Network Alliance (DLNA) is a consortium of more than 200 companies interested in specifying technologies for exchanging media in home networks. The DLNA architecture is based on the UPnP specification, but in addition, DLNA specifies transport protocols (based on HTTP and RTP) and sets of media formats.

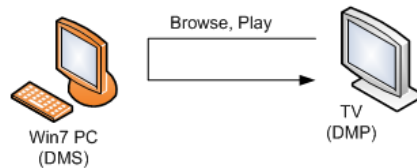
DLNA defines device roles (e.g. servers, players, renderers, etc.) and the protocols that these devices use to discover each other and communicate with each other (e.g. UPnP, HTTP, RTP, etc.). Windows 7 implements several of the DLNA device roles (see table 2 below) and it also implements the DLNA protocols required for communications and media exchange. With Windows 7, your PC will be able to interoperate with a broad variety of DLNA certified devices like TVs, stereo systems, cell phones, DVRs, game consoles, etc.

DLNA Device Class	Acronym	Description
Digital Media Player	DMP	This role finds content on <b>digital media servers (DMS)</b> and provides playback and rendering capabilities. Windows Media Player and Windows Media Center act as a DMP when browsing shared media libraries.
Digital Media Renderer	DMR	This role plays content received from a <b>digital media controller (DMC)</b> , which will find content from a <b>digital media server (DMS)</b> . Windows Media Player acts as a DMR when configured to allow remote control of the Player.
Digital Media Server	DMS	This role stores content and makes it available to networked <b>digital media players (DMP)</b> and <b>digital media renderers (DMR)</b> . When media streaming is enabled, Windows acts as a DMS.
Digital Media Controller	DMC	This role finds content on <b>digital media servers (DMS)</b> and plays it on <b>digital media renderers (DMR)</b> . The "Play To" feature from Windows Media Player and Windows Explorer launches a DMC to control the media playback experience.

Table 2: DLNA Device Profiles Supported by Windows 7

Because Windows 7 implements several device roles, there are different ways in which you could choose to use a Windows 7 PC at home. The remainder of this section explains the different scenarios.

Scenario 1: You store your music, video, and pictures on a Windows 7 PC. You've recently acquired a TV with a DLNA logo. Using the TV, you can browse the media library available on the Windows 7 PC. You can use the TV to watch the video and pictures, and listen to music stored on the PC. Figure 1 illustrates this scenario. In this case, the Windows 7 PC behaves as a DMS. Notice that this scenario was already available in Windows Vista and in Windows XP using Windows Media Player 11.



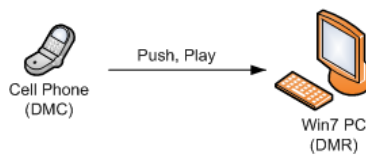
**Figure 1: The TV unit browses and plays content stored in a PC**

Scenario 2: You have a Network Attached Storage (NAS) device where you store your music, video, and pictures. The NAS device implements a DMS. You open Windows Media Player on a Windows 7 PC. You can find the NAS device using Windows Media Player, and you can browse the media library available on the NAS device. You can watch the video or pictures, and listen to music stored on the NAS device. Figure 2 illustrates this scenario. In this case, the Windows 7 PC behaves as a DMP.



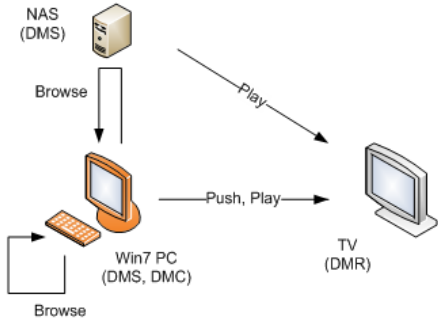
**Figure 2: A Windows 7 PC browses and plays content stored on a NAS device**

Scenario 3: You have a cell phone that not only takes pictures but can push the pictures to a Windows 7 PC. You can show the pictures to your friends using the large-screen display of the PC without the need to physically transfer the files to the PC with a USB thumb drive, for example. Figure 3 illustrates this scenario. In this case, the cell phone acts as a DMS and a DMC and the Windows 7 PC behaves as a DMR.



**Figure 3: A cell phone pushes pictures for display on a Windows7 PC**

Scenario 4: You've acquired a stereo system with the DLNA logo. On his Windows 7 PC, you've accumulated a vast collection of music with thousands of songs. Because your collection is large, you prefer to search, organize, and select songs using the rich capabilities of the Windows Media Player. Once you select the songs, you simply push the songs to your stereo system using "Play To." You also have a NAS device containing an additional collection of music and video. You can use the Windows 7 PC to browse the content on the NAS device and push it to the stereo system. Figure 4 illustrates this scenario. In this case, the Windows 7 PC behaves as a DMS and a DMC.



**Figure 4: A Windows 7 PC browses local content or shared content on the network. The PC then pushes the content for playback in a TV unit (DMR).**

There's definitely a lot to enjoy here. Have fun!!

-- Scott, Tim and the Devices & Media team

# Safeguarding Windows 7 – Parental Controls

Steven Sinofsky | [2009-05-26T03:00:00+00:00](#)

---

*As you can imagine, our team is quite busy working through this next phase of Windows 7. We definitely appreciate the millions of downloads and installs of the Windows 7 RC. Things are going as we expect at this point. On a personal note, I wanted to thank all the folks who have been sending me mail. I've received a lot of kind words and support regarding the RC and quite a few people saying "hurry up and just release it". We outlined the steps we're taking for this next milestone and aren't going to rush things. We've got a lot of work for sure! Not that I'm counting, but I just crossed over 3,000 emails sent via the contact link in this blog. While I haven't answered all of them, I've done the best I can, and appreciate each and every exchange.*

*Windows 7 includes a set of features for safeguarding your PC when used by children. This post is by Vladimir Rovinsky, a program manager on our Safety Team, who details the features in Windows 7 specifically around Parental Controls. This work is in addition to the safety of the OS itself and of course the features built into Internet Explorer to provide safety and security while browsing. You might also want to check out Windows Live Family Safety which is part of Windows Live Essentials (<http://download.live.com>) which provides even more for safety and parental controls. --Steven*

Today, children are exposed to digital hazards more easily than any time in the past. Especially with the help of powerful search tools, convenient social networking applications, low cost tools and services for publishing videos and photographs, the web is awash with content that's inappropriate for children, and full of people that parents want to bar from contacting their children.

These digital hazards are accessible to children through a variety of applications, including web browsers, instant messaging applications, media players, games, and email applications. Many of these applications have attempted to offer parental control features. However, they offer this functionality through variety of user interfaces, locations and include varied terminology. The duplication and inconsistency of parental control settings management can make it difficult for parents to maintain the correct settings across multiple applications.

Windows Vista Parental Controls provided a framework to solve these problems by offering:

- A single, central location in the Windows Control Panel to configure and manage parental control settings and activities;
- Built-in restrictions on web content and file downloads, time spent on the computer, application usage, game usage as well as the ability to log and view user activity.
- The Windows Parental Control platform public application programming interfaces (API) which expose in-box restriction settings and logging functionality to any application. For instance, Internet Explorer and Mozilla Firefox 3.0 are using these APIs to determine if file downloads should be blocked for a user.
- Integration with the User Account Control (UAC) to enforce standard user accounts for parentally controlled users; promotion of best practices for keeping kids safer on a Windows computer; for instance, encouraging the creation of separate standard accounts for managed children, password creation for parent accounts (administrators), etc.

To get a quick demo of Windows Vista Parental Controls in action, check out this [video](#).

For more information about developing software for Windows Vista Parental Controls, see [Using Parental Controls APIs](#).

## **Key Design Decisions for updates to Windows 7 Parental Controls**

Responding to customer feedback and evolving nature of the web and challenges it poses to the parents, we strive to provide families with flexible

and effective safety features. Our efforts for the Windows 7 release of Parental Controls were focused on the following objectives:

**1. Further developing the extensibility of the Parental Controls platform to enable third-party developers to create richer Parental Control capabilities that integrate well with Windows 7 Parental Controls.**

The Windows 7 Parental Controls platform was modified to allow multiple independent providers of Parental Controls functionality to be installed on the system and augment or fully replace the parental controls provided by Windows 7. Windows Vista allowed partial replacement of Windows Parental Controls; the web filter was replaceable. In Windows 7, in addition to the web filter components, the entire Windows 7 Parental Controls user interface can be replaced by third-party providers. The underlying enforcement of the offline restrictions will still be performed by Windows Parental Controls platform. Allowing a third party provider to replace the entire Windows Parental Controls user interface creates a consistent user experience that seamlessly combines existing Parental Controls functionality with the new ones introduced by the third-party provider.

The Windows Control Panel Parental Controls screen still remains the central location and launching point on Windows 7 for Parental Controls functionality regardless of whether it is provided by default (system) or by a third-party provider.

**2. Removal of web content restrictions and activity viewing functionality from default (system) Parental controls provider and reliance on Windows Live or third-party providers for these capabilities.**

The web is changing much faster than we can update the Windows operating system. For example, when Vista was released Social Networking was barely known. Now it has a thriving web presence. We need to keep web focused parental controls up with innovation. Because of this, we have moved them into Windows Live.

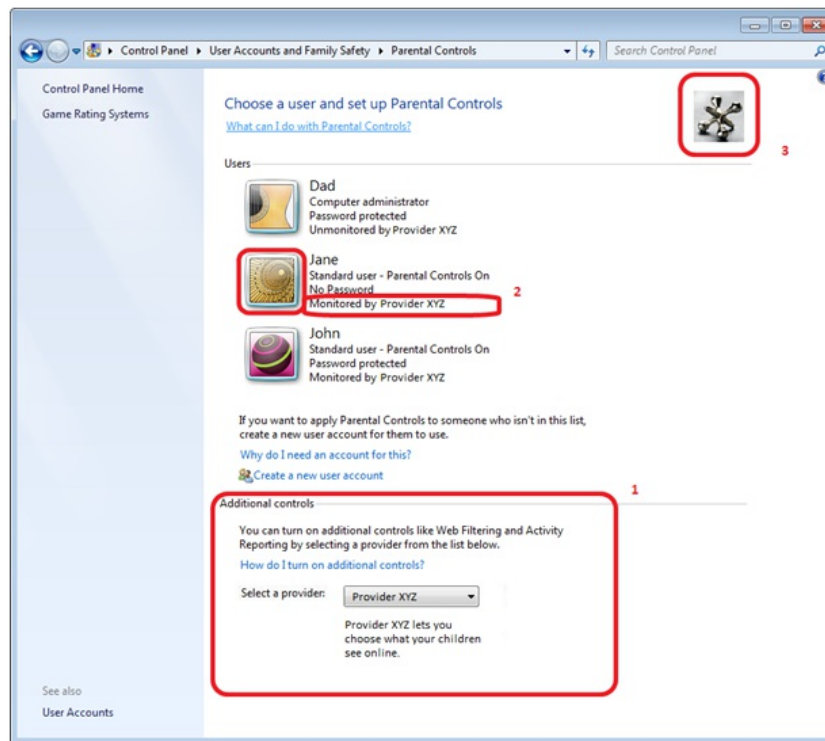
Web filtering and activity viewing capabilities can be more efficiently provided by Windows Live or a third-party solution that implement web based delivery of this functionality. For instance, Microsoft's Windows Live Family Safety free application provides web content filtering, file downloads restrictions, and activity monitoring. It also provides online contact restrictions for children using Windows Live online applications (Windows Live Hotmail, Windows Live Messenger, etc).

You can learn more about Windows Live Family Safety solution [here](#).

More information about Windows 7 changes to the Parental Controls platform can be found [here](#).

**Windows 7 Parental Controls User Interface Changes.**

Elements new to Windows 7 Parental controls top-level screen can be seen on the following screen shot:

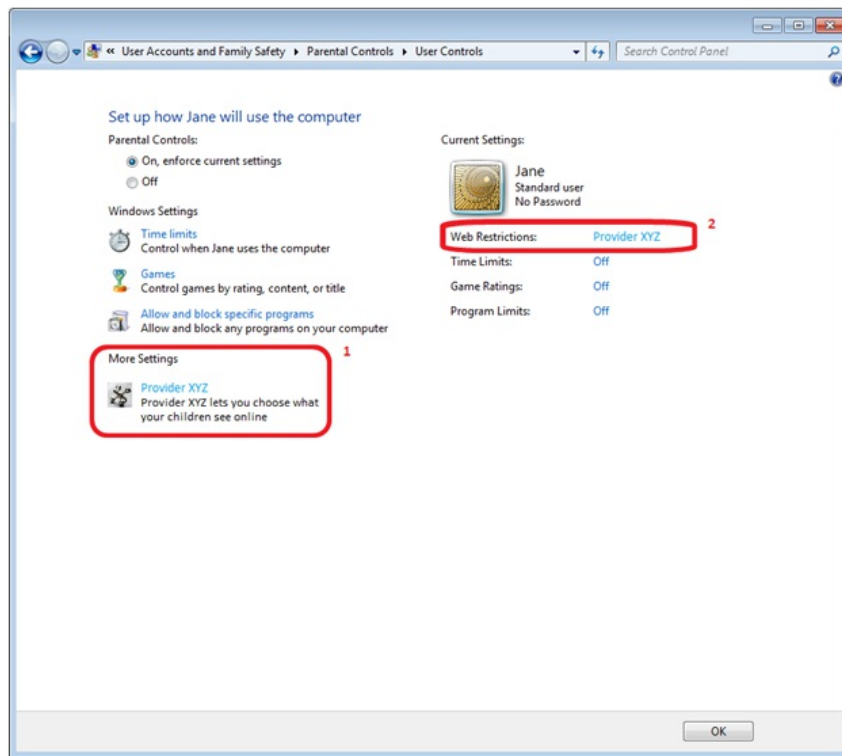


**Figure 1** Windows 7 Parental Controls screen

1. The **Additional controls** section allows users to select a provider for additional controls such as web filtering, activity reporting, online contact management, etc. When a third-party controls provider's installed on the computer, the screen displays the **Select a provider** drop down box that shows the currently selected (active) provider. A description of the provider's functionality, as supplied by the provider, is shown below the drop down.
2. When the user account is selected by clicking user's name or picture, the provider configuration for the user is launched. The provider can take over the default configuration UI for the in-box offline restrictions. Optionally, provider generated status strings for user accounts are displayed under user account pictures.
3. An Icon supplied by provider is shown in the upper right corner of the screen.

Additional control providers can still rely on the default's (system) provider UI for the configuration of in-box offline restrictions. If a provider chooses to do so, the User Controls screen can be presented to configure a user's Parental Controls settings.

If an additional provider is selected and configured, the following new user interface elements are shown on the Windows 7 User Controls screen:



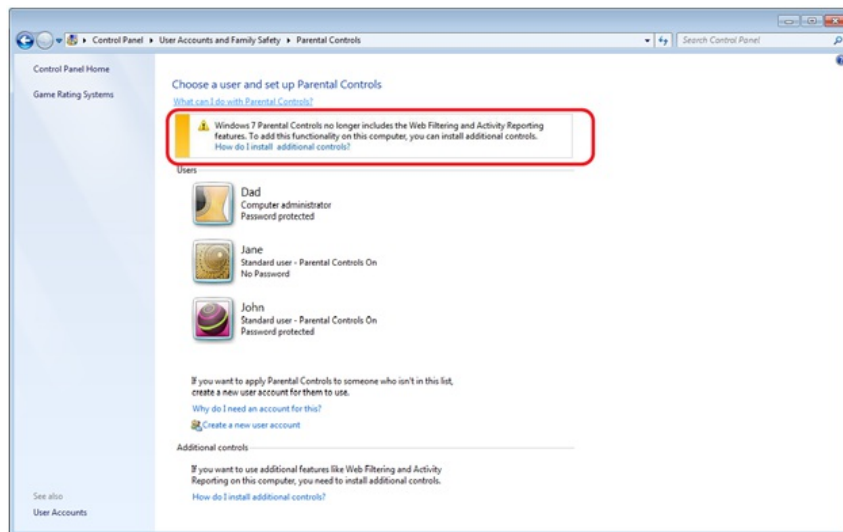
**Figure 2** Windows 7 User Controls screen. Additional controls provider is installed and configured.

1. **More Settings** allows direct access to the currently selected provider's functionality.
2. **Web Restrictions** allows access to the currently selected provider's functionality.

### Windows Parental Controls settings and Vista to Windows 7 upgrade

If a Windows Vista PC which has parentally managed user accounts with enabled web filtering restrictions is upgraded to Windows 7, parents (administrators) are warned during the upgrade as well as when opening the Windows 7 Parental Controls screen, that web filtering and activity reporting functionality is not part of Windows 7 Parental Controls.





**Figure 3** Windows 7 Parental Controls screen. Some users have web filtering restrictions. No additional provider is installed.

Windows Vista Parental Controls settings (including web filtering and activity logs information) are preserved unchanged when upgrading from Windows Vista to Windows 7. Although web filtering settings and activity logs information are not used by Windows 7 Parental controls, their preservation allows third-party provider to honor these settings.

As you start using Windows 7, we hope these changes to Parental Controls capabilities will make you feel more confident and in control of how your family members are using computers and experiencing the web.

--Vladimir

# Creating, Saving, Sharing Themes in Windows 7

Steven Sinofsky | [2009-06-03T03:00:00+00:00](#)

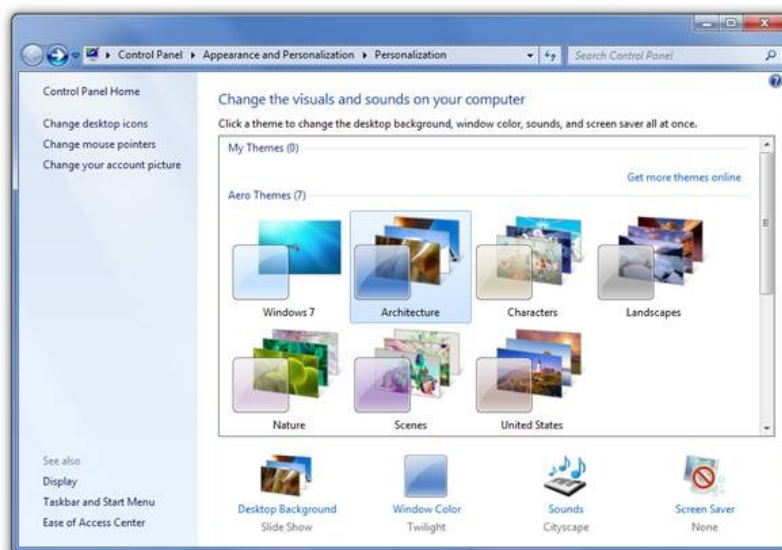
*When we posted the new "inbox" desktop backgrounds, the reactions showed just how personal, personalization can be. Building on that theme of personalization (pun intended), we wanted to share some of the work we did on themes in Windows 7. We've shared data about customization in previous releases of Windows and this post builds on that. This is also an area where we know there is very broad spectrum of desires (needs) for personalization and we definitely had to balance the engineering and design efforts. I've received mail from many folks wanting to personalize (tweak) nearly every pixel on the screen—from border width, to title bar transparency percentage, to height of taskbar, to color/size/location of the close button (I've received each of these in email more than once). At the other end are customers who are enormously happy when they can easily change the background picture and color scheme, and many do. With Windows 7 we picked a group of settings that we believe represent the most satisfying settings to broadly personalize, and would also provide the most robust platform that maintains application compatibility, and made those easy to change. In addition we wanted to make it easy to package up those settings so you could save and share them. We think of this as the start of bringing robust personalization (and customization) to a broader set of customers. Katie Frigon, a program manager on the core user experience team, authored this post.*

--Steven

*PS: Things are "slowing" down as we have talked about in how we will get to the RTM milestone. You might have noticed the announcement we made today in Asia regarding [Windows 7 release and availability](#). Thank you to everyone who has been using the RC and helping to reach the next milestone.*

## Creating and Sharing Windows 7 Themes

In early builds, you may have noticed that Windows 7 includes a variety of themes that change your desktop background, window color and sounds with a single click. These themes are located in the Personalization Control Panel which is easily accessed from the desktop context menu.

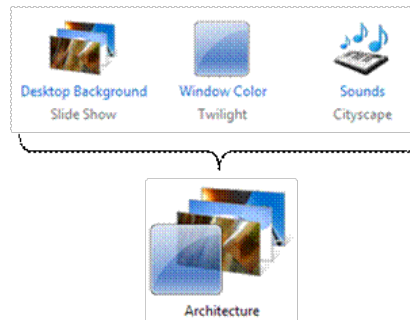


Personalization Control Panel



### Desktop Context Menu

In the RC, you can see a number of new themes, for example the “Architecture” theme. This theme is comprised of six architectural photos which cycle on the desktop background, a complementary “Twilight” window color and the “Cityscape” sound scheme which was inspired by the sounds of an urban jazz club.



**A theme is a coordinated set of Desktop Backgrounds, Window Colors and Sounds.**

Windows provides a set of themes in box and if customers want more there is a prominent link in the Control Panel to get additional themes online. This link takes you to the Windows Online theme gallery where Microsoft provides additional content including a variety of international themes.



**Personalization Control Panel: Get more theme online link**

## Creating a theme

While our customers enjoy the content we've provided both in the box and [online](#) we also know that they enjoy and desire the option to customize their PC's even more than choosing a theme. Windows 7 continues to be about your PC reflecting you and what you do, as well as putting you in control of that experience. So, if you do want to go beyond the options in the box and on the web, it is easy to create and share your own themes. Creating your own theme can be as easy as just changing your desktop background image while keeping the rest of the settings the same or you can change all the settings one-by-one.

From our Beta Customer Experience Improvement Program data we see that customers are changing and creating themes. We also see many users changing the different settings, the most popular being desktop background:

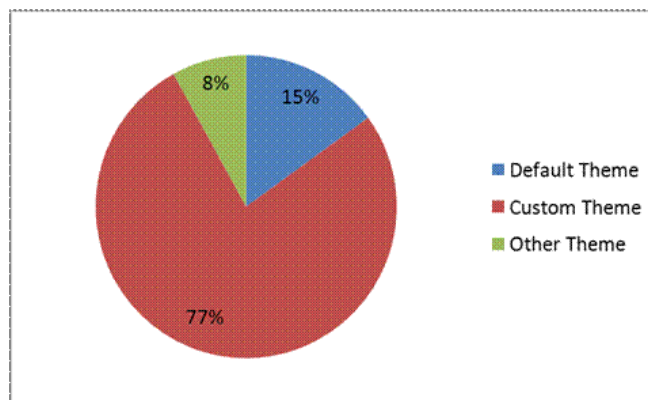
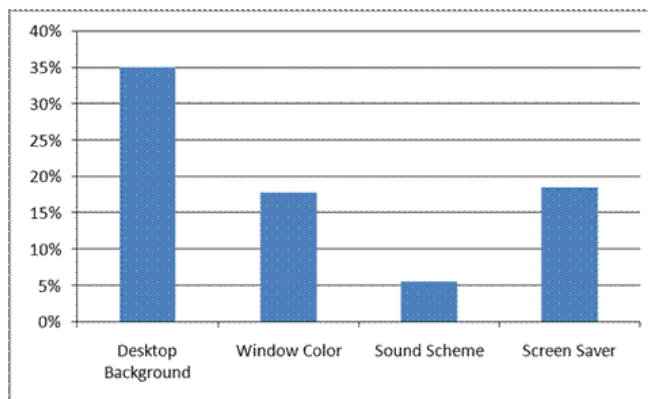


Figure 1: Break out of theme type

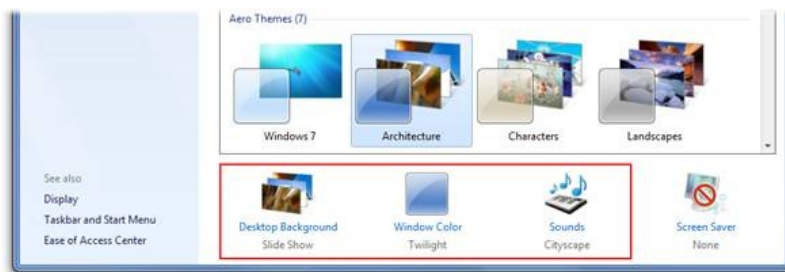
Note: Only 15% of the beta users kept the default theme. 77% of the beta users created a custom theme by changing one or more elements of the inbox themes.



**Figure 2: Percentage of Beta users selecting each theme component in a session**

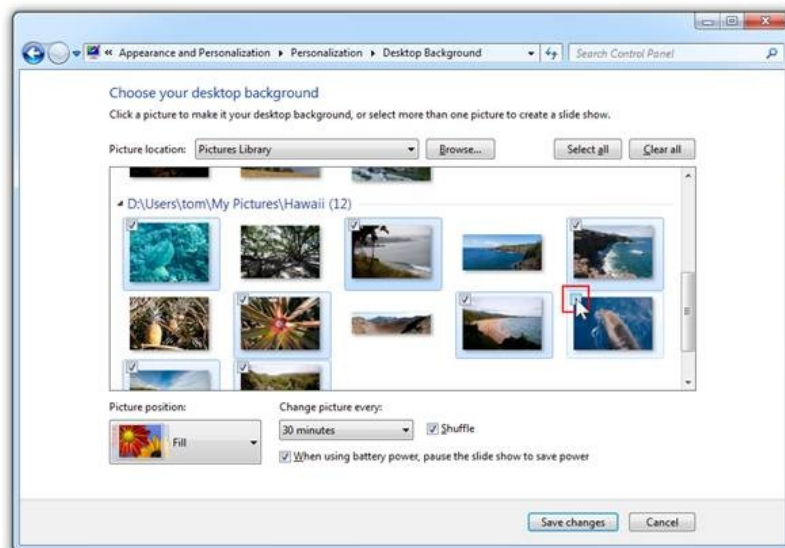
Note: 35% of beta users who opened the Personalization CPL clicked on “Desktop Background”.

Now let’s look at how you can change the different settings and save a custom theme. To start, you can change any of the theme settings by starting in the Personalization Control Panel.



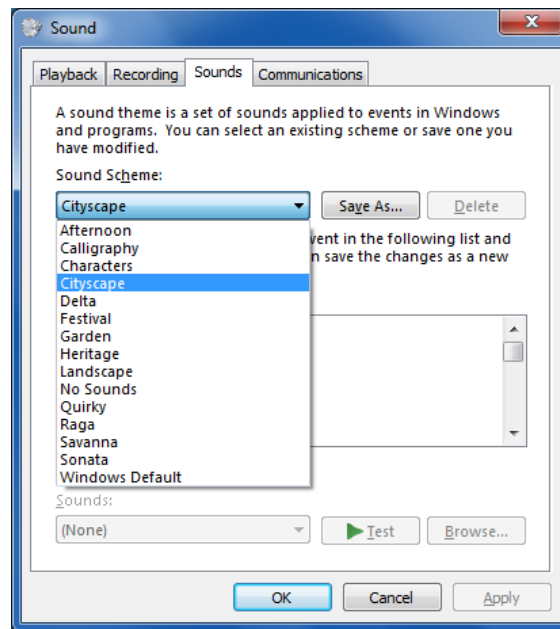
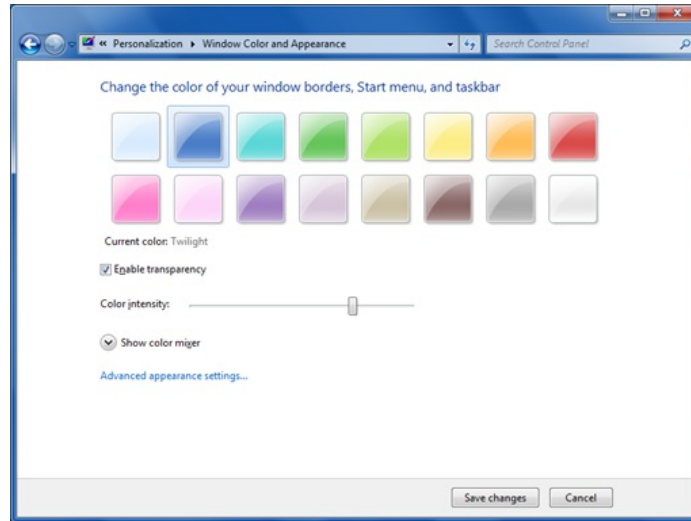
**Personalization Control Panel: Click on the items beneath the theme gallery to change your theme settings.**

Let’s start with the desktop background control panel. This control panel has been enhanced for Windows 7 to support the pictures library and the new desktop background slideshow capabilities. If you choose the “Pictures Library”, we will show all of the pictures in that library including subfolders. All you need to do is select more than one photo to have them cycle as your desktop background slideshow. In this example, I have selected some of my favorite photos from a recent trip to Hawaii to use as my desktop background.



**Desktop Background Control Panel: Windows 7 adds support for libraries and desktop background slideshows. I’ve selected the pictures I want to use in my theme.**

When personalizing your PC, you might want to go further than just changing your background. Changing your window color or sound scheme is simple, just click on the items beneath the themes gallery. We provide 16 window colors to choose from and the ability to pick a custom color as well. New to Windows 7, we include 14 sound schemes with the OS inspired by a variety of regional music traditions, so you have plenty to choose from. If that isn't enough, you can include your own sounds if you want.



**Window Color and Sound Control Panels: It is also easy to change your window color or pick from 14 diverse sound schemes.**

After you change the desktop background, window color or sound scheme, you will notice that we have created a new “unsaved theme” that contains your changes. Your unsaved settings will be preserved when trying other themes in the gallery so you can get back to your most recent

customizations. If you are happy with your personalization settings, you can ensure that they are always available in the themes gallery by clicking "Save theme".



**Personalization Control Panel: I clicked "Save Theme" to ensure that my current personalization settings will always be available in the themes gallery.**

## Sharing themes

After saving your personalization settings for your own use, you might want to share these settings with friends and family or bring the settings to another PC. Windows 7 allows you to share your themes by right-clicking on your current theme and selecting "Save theme for sharing". After specifying a name and folder destination for your theme, Windows will collect all of your custom desktop background images, sounds, mouse pointers and icons into the new .themepack file format that can be applied on another computer running Windows 7.

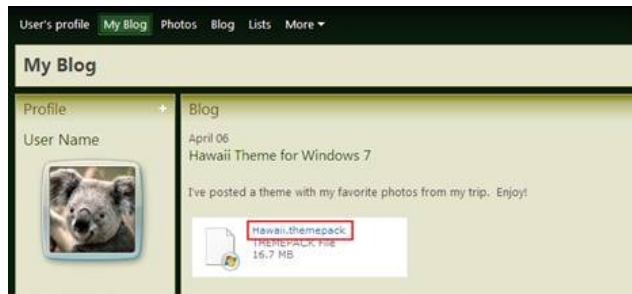


**Personalization Control Panel: When I'm ready to share my theme with Friends, Family and on the Web, I right-click on my current theme and select "Save theme for sharing".**

Sometimes after I take a fun vacation I like to create a theme that reminds me of the trip. To do this I select the best photos from the trip to rotate as my desktop background and then pair those with a matching window color and Windows 7 sound scheme that best matches the mood of the trip. After I save as a new .themepack I can either share this file via Windows Live to friends and family or use it from another PC in my house via Homegroup.

## Sharing with Windows Live

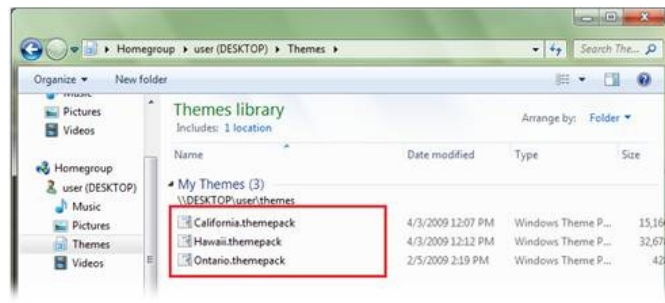
Since all of the personalization settings are now contained in a single file, it's easy to upload the theme to [Windows Live Skydrive](#) and post a link to the theme on a [Windows Live Spaces](#) blog. Once my friends and family upgrade to Windows 7, they will be able to download themes from trips that we went on together so they can enjoy my photos on their desktop background.



**Windows Live: I can also upload my theme to my Windows Live Skydrive and add a link to the theme on my blog.**

### Sharing via Homegroup

In Explorer you can create a themes Library. Then from another computer in a Homegroup you just browse to the shared location and click on the desired theme to apply those settings with a single click.



**Explorer: I created a themes library on one of my PC's and shared it with my Homegroup. From another PC in the home, I can click on any of these themes to apply them.**

### But wait...there's more.

One additional way we've added value with Windows 7 themes is by capitalizing on the growing popularity of RSS photo feeds to share photos. Enthusiasts can create a theme where the desktop background slide show points to an RSS photo feed. For example, my sister lives across the country and we only see each other about once a year. An easy way for me to keep her up to date on my family is to send her a Windows 7 theme which points to my RSS photo feed. When I upload new photos they will appear on her desktop automatically.



Because there are a few different ways to create an RSS photo feed, the process to include an RSS photo feed in a Windows 7 theme will only work if your RSS photo feed links to the high resolution photos using the “enclosures” method. The feed should only reference picture formats such as JPEG or PNG. Due to this limitation themes must be created manually when including an RSS photo feed.

So, to create one of these themes you can follow these steps:

1. Download the template from [MSDN](#).
2. Open the template using Notepad.
3. Replace {themename} with the name you want to appear in the Personalization Control Panel themes gallery.
4. Replace {rssfeedurl} with the full path to your compatible RSS photo feed.
5. Save the changes as a file with the “.theme” extension.

It is ready for you to share! Send the file via email, etc. to your friends and family.

Photo sharing sites can also offer these Windows 7 RSS photo themes which provide more ways to connect their customers.

## **Looking ahead**

Themes in Windows 7 make it possible for you to make the PC reflect you. Beyond my example of sharing personal photos as a theme, we hope that users will find new and creative ways to use themes in Windows 7. Wedding photographers can include Windows 7 themes in the packages they deliver to their clients, Artists can create themes that showcase their creative style and businesses can create themes that promote their brand. We look forward to seeing how you are using themes to Personalize these aspects Windows 7.

--Katie

PS: We've posted some additional themes you can download and use on <http://windows.microsoft.com/en-US/Windows7/Personalize> which is the US English link from the Themes control panel.

# Improving Audio Glitch Resilience in Windows 7

Steven Sinofsky | [2009-06-17T03:00:00+00:00](#)

---

*Delivering excellent audio playback on a PC is one of those “much harder than it looks” technical challenges. Unlike dedicated audio / video devices, PCs have a lot going on during playback of audio and the playback happens on an incredible array of hardware and software. Many of you might be familiar with “glitches” that occasionally happen. In this post, Kristin Carr, a program manager on our Devices and Media team, describes some of the engineering in Windows 7 to improve this area representing the work of a number of folks across the team. One lesson I learned early in the product cycle is that we don’t say “glitch-free” but rather “glitch-resilient” and hopefully that will make sense as you read this. --Steven*

Have you ever used your PC to play an MP3 or a DVD? If you answered yes, you’re among the overwhelming majority of PC customers who use their computer for audio and video applications, encompassing everything from watching a movie to playing a game to viewing a YouTube clip. But you may have also had an experience where your audio or video wasn’t quite perfect – perhaps the video was a bit choppy or the audio stuttered. We call this a ‘glitch’ – a perceived discontinuity in your audio or video that interrupts the playback experience. In this blog post, we’ll be focusing on audio glitching; we’ll examine the ecosystem challenges that can cause glitches, and we’ll discuss the work we’ve been doing to improve the Windows 7 experience.

## What Causes Glitching?

In previous posts, we’ve touched on a variety of ecosystem initiatives and challenges that we’ve undertaken for Windows 7, including [application compatibility](#), [accessibility](#), and [system performance](#), among others. Tracing the root cause of audio glitching leads us to a similar place: because Windows runs on a huge variety of hardware configurations and multitasks between dozens of applications, it is challenging to ensure that all of the programs and drivers running on your computer will work together in exactly the way you expect.

Audio is especially sensitive. In order for you to hear music from your speakers, data needs to be delivered to your audio hardware approximately every 10 milliseconds, or 30 times in the blink of an eye! The challenge is that your PC is usually doing a lot of other things at the same time you’re listening to music, such as streaming that YouTube video or downloading that new song, and many of these other tasks have complex timing requirements as well. As you can imagine, it doesn’t take much – a slow network driver or a graphics driver that requires plenty of CPU time – to prevent your audio from reaching your ears in a continuous fashion.

So what are we doing to address this challenge? The answer is ‘lots!’ – and the remainder of this blog post will be devoted to discussing these things:

1. Gathering data in order to characterize the problem
2. Developing a systematic method to detect and analyze glitches
3. Getting these tests and tools widely deployed, both at Microsoft and by our Windows partners
4. Engaging with partners to detect, diagnose and fix glitching issues

## Who Experiences Glitching?

In studying this during the Windows 7 development cycle, our first order of business was to gather data. We’d heard reports of audio glitching, but we didn’t know the exact scope of the problem. How often do users hear their audio glitching? Are there certain machines that were worse than

others? With these questions in mind, we set out to understand our problem space a bit better.

We gathered data by using the [telemetry infrastructure](#) built into Windows, which allows our users to report back to Microsoft with performance data and other statistics that help us improve the OS. For each machine that opted to contribute data to Microsoft, we measured the number of times that the underlying audio hardware was being starved for data (i.e., when the user might hear a glitch). This data was grouped into “sessions,” each of which represents the data collected on a single machine for a single day or the data collected between machine reboots, whichever is shorter.

Let’s dive into some of the results. First, let’s look at the overall rate of audio glitching:

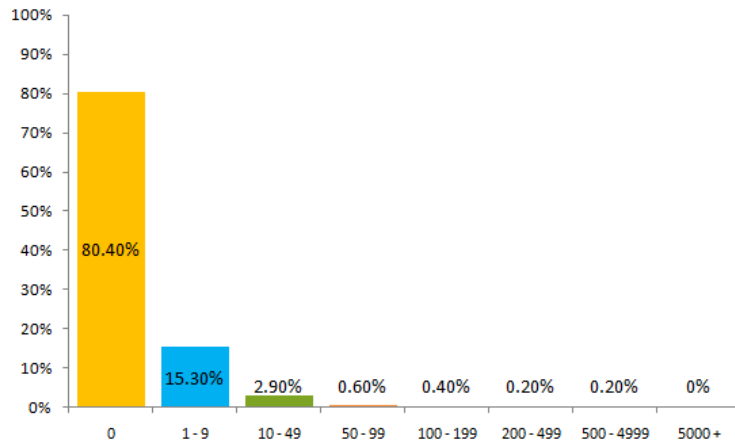


Figure 1: Distribution of Glitch Counts per Session

The chart above shows data from external (non-Microsoft) RC users. Approximately 80% of sessions showed no glitching at all, but 4.3% showed 10 or more glitches, which indicates that audio glitching affects a significant number of users.

Once we figured out how often glitching occurs, we started looking into *why* it occurs. First, we broke the data down by laptop/desktop form factor:

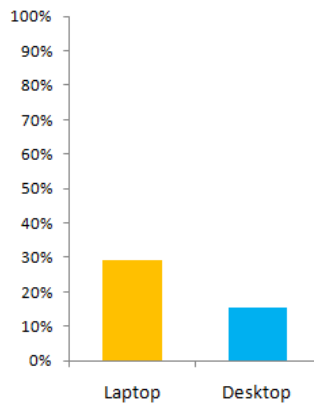


Figure 2: Glitching Likelihood by FormFactor

From this data, we noticed that laptops were almost twice as likely to experience audio glitching. As a result, we've made sure to address and target mobile PCs as well as mobile scenarios (for example, playing music while running on battery) for better coverage in our glitching tests and diagnostic tools.

Next, we looked at glitching likelihood by PC manufacturer:

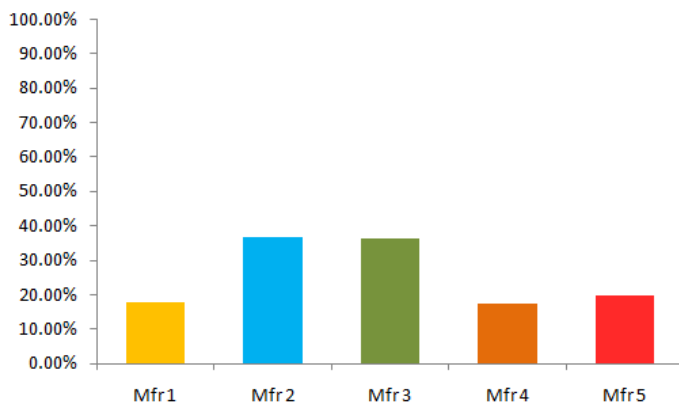


Figure 3: Glitching Likelihood by PC Manufacturer (Mfr)

This data showed that certain manufacturers were more likely to be susceptible to audio glitching than others. As a result, we made sure to spread our testing efforts across a wide spectrum of machines and manufacturers. In addition, we are using this data to work with manufacturers to see if we can identify components or specific causes that would result in higher glitch incidents.

Finally, we looked at glitching on a wide variety of PC models:

PC Model	Percentage of Glitching
Machine A	5.7%
Machine B	1.6%
Machine C	1.4%
Machine D	1.4%
Machine E	1.0%
...	...
+ hundreds of machines	...
...	...

Figure 4: Breakdown of All Glitch Sessions by PC Model

In the chart above, we examined all of the sessions that had at least one glitch, and we looked for any correlation with the PC make and model as

shown in the table above (actual machine names have been anonymized). The first thing to notice is that Machine A is responsible for more than three times as much audio glitching as any of the other machines on the list. This data confirmed earlier reports of audio glitching on this particular machine, which we traced to a graphics card that shipped in a faulty configuration. As a result, we were able to work with the manufacturer to improve the configuration.

This chart also helps to show how widespread the issue is. There were hundreds of PC models that showed evidence of glitching – in fact, it seemed difficult to find a single PC model for which audio glitching did not ever occur. On the other hand, most individual machines didn't show any problems at all. The conclusion that we drew was that audio glitching was not caused by any one hardware configuration, but was *dependent on all the different hardware and driver permutations a user could possibly encounter on their machine*. It was clear that no machine was immune, and in order to improve the experience, we were going to need a far-reaching, system-wide solution to this problem.

## Developing Tools to Diagnose Glitching

Once we had data on when and why glitching occurs, the Windows Devices & Media Performance team developed a comprehensive suite of tests that were centered around media playback scenarios and were designed to assess how well a PC performed at that scenario. During media playback, these tests recorded thousands of statistics about the system's performance, including CPU load, the activity of all components on the system and their corresponding interactions, and whether glitching occurred, among other things. We intentionally covered a huge range of scenarios and configurations, including laptops running on battery power, hardware under stress, hundreds of media content types, and many more. The goal was to exercise each PC in a wide variety of user scenarios in order to uncover and isolate audio glitches.

In addition, the Devices & Media Performance team created a graphical tool to highlight glitches as well as the CPU activities that occurred before and during an audio glitch, which allows us to quickly diagnose any glitching problems that we uncover. For example, in the figure shown below, we can see a visual representation of when glitches occurred, and we can display related measurements that occurred at the time of the glitching in order to easily pinpoint any suspicious behavior.

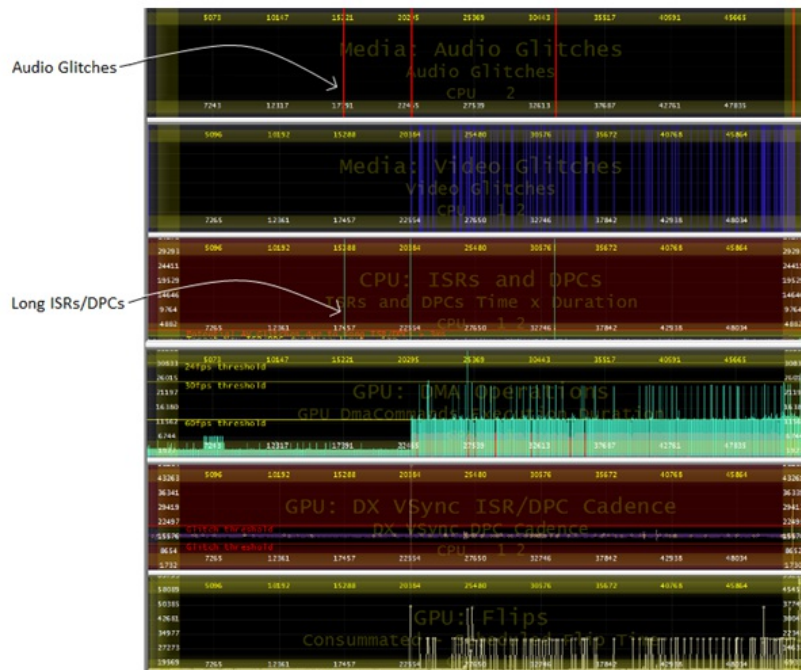


Figure 5: Example Graphical View of Audio Glitch Troubleshooting

In this case, you can see four audio glitches (shown by red vertical lines in the top panel). Two panels down, we have displayed calls to the CPU that took longer than 3ms (called long ISRs/DPCs). In this example, you can see a direct correlation between audio glitches and long ISRs and DPCs, which are procedure calls executed by the operating system that have the potential to hog the CPU and produce audio glitches. From here, we can track down the components responsible for these calls in order to reduce or eliminate the glitching. This figure shows additional information than what we used to diagnose the particular problem discussed above; however, this information and the many other measurements are available to diagnose other glitches and media performance issues from across a wide range of sources.

## Putting the Tools to Work

Armed with these tests and tools, our next step was to deploy them on as many systems as possible. As part of this effort, we are participating in a Windows-wide initiative to help OEMs test their PCs at or before ship time. Hundreds of OEM machines get shipped to Microsoft for use in our Windows lab where we run thousands of tests in order to validate and ensure the best user experience. What this means is that if we notice that a particular machine or configuration might be susceptible to glitching, we can work with the OEM to try to fix the problem before the consumer ever sees their new PC.

By running these tests and analyzing the results with our new tools, we've been able to find hundreds of potential issues that would result in audio glitches. In some cases, this analysis resulted in changes to the Windows code. In other cases, we have identified components developed by our partners that can lead to audio glitching.

## Engaging with Windows Partners

Since the issues we identify with these tools often involve components from many different partners, an important aspect of this work is engaging with these partners. Until now, it has been almost impossible for manufacturers to know how their components will affect the system as a whole, but by making these tests and tools available, we are attempting to enable these partners to see how their components interact and what the final impact on users will be.

As part of this effort, we have been working to ensure that our partners can take full advantage of these new tools and tests. We've talked with OEMs, ODMs (original design manufacturers, who traditionally assemble the PC for the OEM), hardware manufacturers, and software vendors. We've given presentations and tutorials, written whitepapers, and held video conference workshops. Our goal has been to make it as easy as possible to create glitch-resilient software and hardware.

In summary, this effort includes:

1. **Sharing audio glitching telemetry data** with our partners. Our partners have had very little concrete data on the prevalence of audio glitching. With the data we are now collecting, we can help them to diagnose problems and improve their products.
2. **Running our suite of audio and video performance tests** on the hundreds of machines that OEMs send us and communicating the results to our partners. By assessing as many systems as possible and providing these results, we begin to tackle the causes of audio glitching.
3. **Providing the tools and support** that enable our partners to understand how their components are interacting with everything else on a PC and enable them to more easily address the subtle issues that can result in audio glitching.

## What's Next

Ultimately, we and all of our Windows partners share a common customer (you!); by working with our partners, calling attention to these issues, and providing more insight into the root causes of audio glitching, we are continue to improve the audio experience for everyone.

# Engineering Changes to ClearType in Windows 7

Steven Sinofsky | [2009-06-23T03:00:00+00:00](#)

---

*One of the many passions held by Bill Gates is a passion for reading and so his desire to make reading on PCs a fantastic experience has been an effort ongoing for many years. In the [1998 COMDEX](#) show, Bill Gates unveiled ClearType – hard to believe it was that long ago. Back when it was announced, very few of us had LCD monitors and those that did invested several thousand dollars in one that was 15” and 1024x768 (today one like that costs less than \$100). The notions of smoothing and anti-aliasing have been around for many years and are common in the world of typography, animation, and games. ClearType took this to new levels by building on the properties of LCD panels. ClearType was subsequently included in Windows XP and continues in Vista and Windows 7—each release saw changes in the underlying technology, the fonts that support the technology, and the APIs available to developers. It is fair to say that over the years we have learned that there are a set of customers who simply find ClearType rendered screens less than appealing and wish to turn it off. We recognize this and want to make sure we provide the appropriate controls. ClearType is also part of the Windows platform and provides APIs callable and controllable by developers of applications. There is a conventional view that ClearType is a "visual preference" and through this post we want to show how there are elements that are such a preference but there are also elements that are APIs used by applications, just like applications can choose fonts, colors, and other attributes as required. This post goes into the details of Windows 7's implementation along with some history and background. Greg Hitchcock is the development lead on ClearType and has worked on it since the start. He's also one of the most tenured members of the Windows 7 engineering team with only 6 folks having been at Microsoft longer – [Larry](#) is one of them ??!*

--Steven

Based on feedback, we want to clarify how font rendering works in Windows 7 and give some background on how we chose ClearType font rendering to be the default in Windows. For those that dislike ClearType and want to change the system *default* setting to bi-level rendering, as were defaults in Windows Millennium, the quick answer is:

- Enter **Appearance** into the start menu search
- Select **Adjust the appearance and performance of Windows** from the Control Panel
- The setting that should be changed under the **custom** option is: *Smooth edges of screen fonts*, which should be turned off

The longer answer, as we will describe in this post, shows that changing the default setting is not as “black and white” as it may seem. As you have noticed, Windows 7 also includes a new ClearType tuner in the control panel which affords fine-grained control over rendering—we’ll talk about that some below as well.

## ClearType

ClearType is a technology developed to improve both the appearance of font rendering and reading performance on computer displays. As most people spend over 80% of their time on computers reading on the screen, improvements in this area greatly improve the overall experience of Windows. The ClearType technology has continued to evolve and the latest improvements have been made in Windows 7, as discussed in this [earlier E7 post](#).

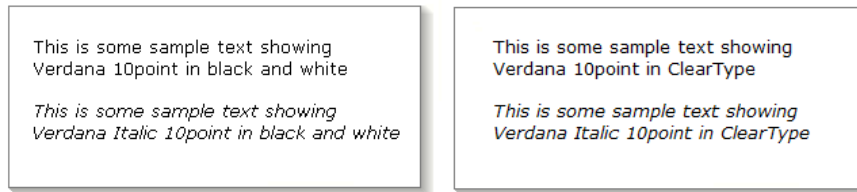
In simple terms, ClearType works by using the underlying geometry of colored sub-pixels in the display as if they were full pixels—gaining extra resolution while at the same time using principles of human vision to remove the perception of color artifacts. Further details on the technology and how it uses human visual perception are described [here](#). More specifically, the ClearType technology is optimized for LCD panels with red, green, and blue (RGB) striped sub-pixels that are oriented vertically, although it performs reasonably well on CRT displays (especially those that are aperture grille based) and even LCD panels with horizontally oriented RGB stripes. Although this might seem counterintuitive, through informal studies, we’ve found that about 70% of users prefer ClearType even on these non-optimal displays. Of the 30% who preferred other rendering techniques, their biggest concern with ClearType in these non-optimal cases was the loss of text contrast.

## Other Types of Font Rendering in Windows

Given the complex world of many display types and a wide variety of users and their visual systems, how did we go about implementing ClearType into Microsoft Windows? Microsoft did not rush headlong into making ClearType the default rendering. The technology was first released in 2000 with the Windows CE product. The Windows CE environment is usually quite controlled in terms of the hardware used, so it was quite easy to verify that ClearType worked properly on each device, and either tune ClearType or adjust the hardware to optimize the onscreen reading experience. The first release of ClearType on the Windows platform was with Windows XP in 2001.

### Bi-Level Rendering

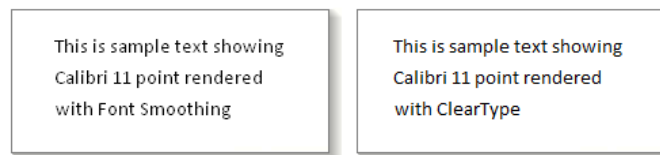




Example of Bi-Level rendering. Note if your browser scales this image the text will not be correctly represented.

Prior to Windows XP, two types of font rendering were supported in Windows. The first type of font rendering supported was bi-level rendering, more commonly referred to as “black and white” rendering, but sometimes also called aliased text. With bi-level rendering, two colors represent the font, the foreground color and the background color. This was the first type of rendering supported by TrueType when Windows 3.1 was released, and also the essential method of displaying fonts in bitmap form from Windows 1.0. Bi-level rendering, especially when generated from outline technologies like TrueType, is very difficult to optimize for low screen resolutions. Significant effort needs to be put into the [font hinting for TrueType](#) in order to get the best bi-level quality. It can reasonably take a skilled person 6 months to a year per font of hinting time to get this level of rendering detail. That would be multiplied by four for a four-member family. If the character sets are larger, as in some system fonts, it can take even longer.

#### Font Smoothing / Grayscale



The second form of rendering, known as font smoothing, became the default rendering in Windows 2000 and was first released as an option for Windows 95 in the *Plus!* Pack. Font smoothing is a hybrid grayscale anti-aliasing technique designed to improve the contrast of fonts over traditional anti-aliasing techniques. There are two factors that differentiate font smoothing from more traditional text anti-aliasing.

First, traditional anti-aliasing works by overscaling the font outline data and then downsampling. Font smoothing uses the same technique, except that it applies font hints prior to overscaling the outline data. Although we don't have enough space here to fully describe font hinting, I can simplify it enough to say that it often uses a method called “grid fitting” to snap the vertical and horizontal edges of the font outlines so that they are aligned with the pixel grid. In this situation, most horizontal and vertical stems of a font, when overscaled, cover 100% of the underlying pixels, and when downsampled return the text foreground color, which is usually black. Diagonal and round features of the font will not have full coverage of the pixel and thus will return some shade of gray, reflecting the coverage of the underlying pixel. It should be noted that when text displays the “jaggies” (or more formally aliasing) this usually occurs on round or diagonal parts of the glyphs—exactly the areas receiving gray coverage with this method. This way of anti-aliasing is beneficial due to the higher contrast of the stems in the font at a slight cost of some spatial accuracy.

The second differentiating factor is that the fonts determine the exact size that the font smoothing turns on or off. Most fonts that provide this level of information turn on grayscale anti-aliasing below 9 pixels per em (PPEM.) That is the equivalent of 7 points on a 96 PPI screen. Above 9 PPEM, anti-aliasing is turned off until the main stems of the font are around two pixels wide, which is around 13 to 20 points, depending on the typeface. Once the main stems are two pixels wide, anti-aliasing remains on as the sizes increase. Two pixel wide stems are usually chosen because there is usually enough “backbone” of foreground colored pixels to keep the stem contrast high. If the font does not have specific sizes for font smoothing, system defaults are used. There are independent defaults for both regular and bold typefaces. So although font smoothing was the default, most fonts, when displaying text at typical reading sizes, would render them bi-level.

#### Defaults for Font Rendering

With the addition of ClearType to Windows XP, there were now three types of font rendering available (Bi-level, Font Smoothing, ClearType). During the time period that Windows XP was being developed there was a clear transition underway from desktop systems with CRT monitors to laptops and desktops with LCD displays. Since this transition was far from complete, we felt that the default value for font rendering in Windows XP should be grayscale font smoothing, the same as Windows 2000. OEM's who provide Windows XP on their systems could change this default, and in fact, by the time Windows XP SP 2 shipped, many of them had set the default to ClearType. It should be noted that OEMs can always change these settings as part of configuring a PC.

In Windows Vista, the system's default font rendering was changed to ClearType. It is important to clearly understand what is meant by default font rendering. In Windows, the default font rendering is the rendering used when the application does not choose an explicit type of rendering. Some have confused this default value to mean that all applications must use this value. This view is inconsistent with the way text APIs worked when introduced in Windows 95's font smoothing. In GDI, the API for choosing the current font has the rendering type explicitly as input. It is expected that there are situations where the application knows best what type of rendering should be used. For example, in displaying a print preview page with small text, traditional font smoothing might be the best choice for rendering. Likewise, if an application was targeting on-screen

reading, it might be best to use ClearType as the rendering for that application. In some situations, like remote terminal services, the application might choose to use bi-level rendering to reduce the bandwidth of text information that needs to be sent to a remote client.

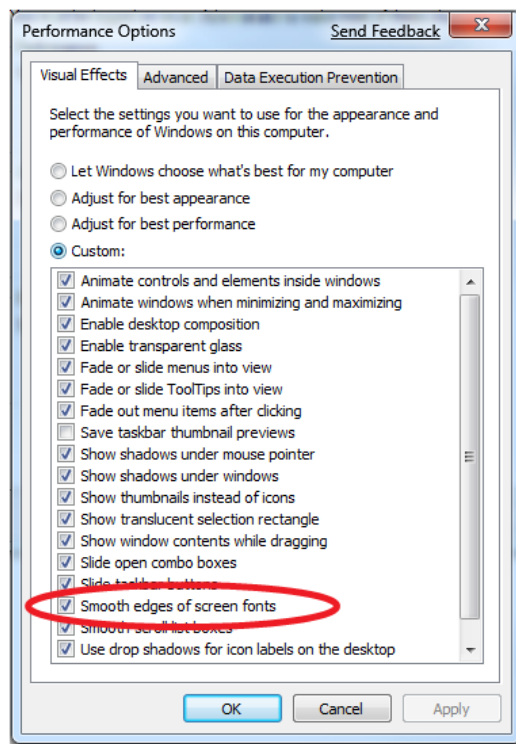
There are many examples where applications decide one way or another to use rendering other than the system default—just as applications that choose to use different fonts, colors, sizes, or other text attributes. The most typical example is in applications that wish to have reproducible layout and flow of documents. By being specific about which way to render text, the applications can be certain of how text will flow across different PCs. Another common example, as mentioned above, is Print Preview where the ability to properly render representations of higher resolution output, particularly for small text sizes, is much improved. We recognize that for some it is counter-intuitive that an aspect viewed as a “display” property is something that applications can choose to “over-ride”. We’ve designed rendering so that the default case is to respect the setting, but applications, including Windows itself, might have elements that require explicit rendering techniques.

Although each application can make the choice on a per-font basis of which rendering to use, the majority of applications choose the default rendering. Therefore, making the decision to change the default for Windows Vista was not taken lightly. The trends in the hardware displays were strongly showing a rapid movement from CRTs to LCD-based displays as we have shown in earlier blog posts based on the Windows XP and Windows Vista real-world telemetry. Even though there were still CRTs in use, feedback from Windows XP customers was positive on the quality of ClearType rendering on CRTs. After we made the choice, the feedback on the decision to enable ClearType as the default for Windows Vista was overwhelmingly positive.

Even with the default rendering set to ClearType, there are some scenarios that can change the default. If an OEM is providing Windows on their hardware, they can change the default. In some situations, and this was more common with font smoothing in Windows 95, the hardware may not meet the minimum requirements for the rendering technology. In the case of both font smoothing and ClearType, a minimum of sixteen bits per pixel display resolution is required. (When rendering to an off-screen bitmap in GDI, it is important that it not be the default color depth of 1 bit per pixel if you desire to capture ClearType text.) In some cases, when optimizing for system performance, font smoothing (both ClearType and grayscale) can be disabled. In a similar fashion, using Remote Desktop to connect to another computer or session usually disables ClearType by default.

### Changing the Default Rendering in Windows 7

Windows 7 maintains the same defaults as Windows Vista. There are several ways for the user to change the default values for font rendering in Windows 7. For those that want the default rendering to be bi-level, the user interface for this selection is in the performance section of the Control Panel. From the root of the control panel you can access it by selecting *System and Security* → *System* → *Advanced System Settings* → *Performance (Settings...)*. An easier way is to enter “Appearance” into the start menu search, and then select “Adjust the appearance and performance of Windows.” The setting that should be changed under the custom option is: *Smooth edges of screen fonts*, as shown in the figure.

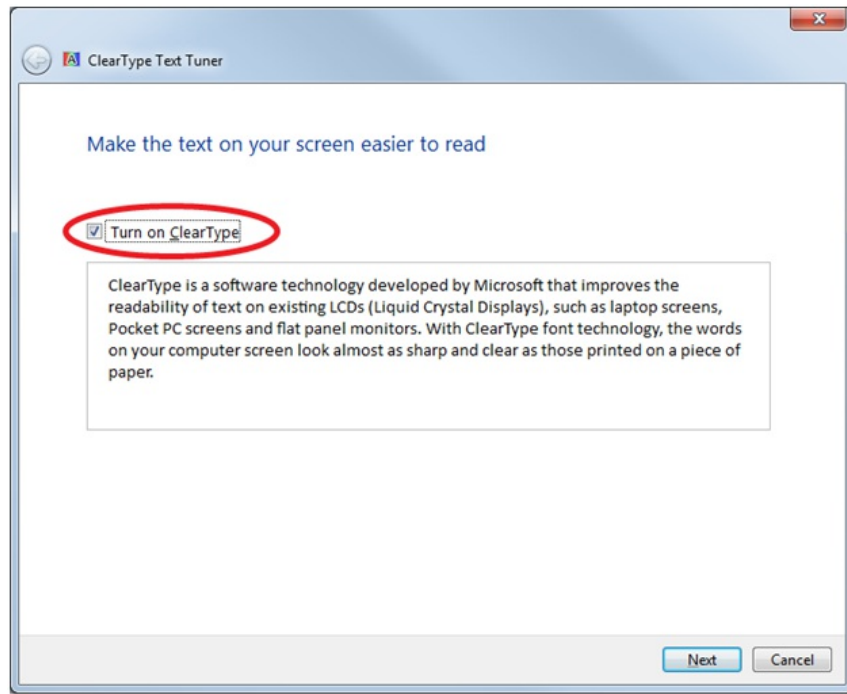


The option of no font smoothing as the default value is considered to be an uncommon setting, so it is a little more difficult to find than other settings. If the user prefers to change the default font rendering selection to the Windows grayscale anti-aliasing technique described earlier, in Windows 7 that is now done through the ClearType Tuner.

## ClearType Tuner

The quality of the ClearType text can be optimized for you and your monitor. The ClearType Tuner is a new control panel component for Windows 7. Because there are differences in monitor characteristics and differences between readers' eyes, there are font rendering options that can only be optimized by a reader looking at text on their monitor. The ClearType Tuner uses various samples of ClearType, presented in the form of an eye-test, to make fine grained adjustments to the ClearType algorithms. Each wizard page tunes a parameter such as monitor gamma (relationship between voltage and brightness), your sensitivity to color artifacts, and your preference for letter heaviness.

In order to switch between ClearType and grayscale, the setting "Turn on ClearType" on the opening page of the ClearType Tuner can be toggled.



Either way, the user is taken through the rest of the ClearType Tuning wizard for two reasons; if an application explicitly enables ClearType rendering, it is useful for that experience to be tuned, and some graphics platforms have more fine tuning of the rendering for both gray rendering and ClearType.

## Font Design and Font Rendering

The availability of higher resolution font rendering techniques like ClearType has had a significant impact on the design of fonts for onscreen reading. From the early days of the printing press, as new technologies and printing styles were developed, typefaces were redesigned to take advantage of those technologies. For example, many typefaces still in use today incorporate "ink traps" into the design so that ink would not clog up key features of a glyph. This is an aspect of making specific design choices in the font in order to work the best with the technology. In traditional typeface design, the term font refers to a typeface at a given size. So a 10 point *Times New Roman* would be a different font from a 24 point *Times New Roman*. In the days of metal cast typography, each of these sizes were designed by a punch cutter to be optimized for the medium for which they were to be used, often with changes in stem contrast, x-height, or character spacing for a given size. The advent of photo typesetting in the mid-twentieth century was a step backwards in this regard, as it used one size as a type master, and then used optics to scale that master size to any other presentation size.

Microsoft Windows has taken the more traditional approach to digital outline fonts, and through a combination of font hinting and new typeface design we attempt to optimize each size for the medium for which they were intended. With Microsoft's initial release of TrueType for Windows 3.1, the traditional typefaces *Times New Roman*, *Arial*, and *Courier New* were used as core fonts. In the creation of these fonts, one master size was chosen for the outline data, usually something around 10 or 12 point, and, similar to the technique used in photo-typesetting, the outlines could be scaled to any requested size for a given display resolution. But, going back to the more traditional ways, each size was carefully examined and changes were made to the basic design through font hinting—including changes to critical features like stem contrast, x-height, or glyph spacing. As earlier mentioned, hinting fonts to be tuned for a low-resolution medium like full pixels on a 96 PPI screen was very time consuming. To help in this process for Microsoft Windows, we commissioned or designed in-house new outline typeface designs that were optimized for the world of 96 PPI bi-level rendering. These custom fonts include *Tahoma*, *Verdana*, *Georgia*, *Trebuchet MS*, and even *Comic Sans MS*. These fonts still needed to be hinted to tune the individual sizes, but because the typeface was designed with the medium in mind, it was a more straightforward

process and less time consuming.

Even with typefaces tuned to the display medium, 96 PPI pixels on a screen are still larger than many of the features we'd like to show in a typeface—and that is where ClearType helps us. Therefore, with ClearType, it made sense to commission a new set of fonts that were optimized for this new medium. Now the existing fonts for Windows still work well with the technology, but this project was an attempt to get the very best design for onscreen reading using ClearType. This led to a new set of fonts that shipped and were tuned for Windows Vista. The [ClearType Collection fonts](#) of *Calibri*, *Cambria*, *Consolas*, *Corbel*, *Candara*, *Constantia*, the new user interface font *Segoe UI*, and the Japanese font *Meiryō* were designed for this medium. As part of the engineering work on these font projects along with the default setting of ClearType, we decided in the hinting process to do the fine, size-specific hinting only for ClearType, and not for bi-level rendering. This allowed us to focus our efforts on the fine levels of detail and quality for the vast majority of customers.

## ClearType Fonts in Windows 7

A reasonable question for us to ask ourselves is what is the experience like in Windows 7 when bi-level or hybrid font smoothing is chosen as the default?

As mentioned earlier, not all applications will choose to render with the default settings. Microsoft Office and Internet Explorer will default in some cases to using ClearType rendering. Some applications that use fonts tuned for ClearType and not bi-level rendering may choose ClearType rendering to maintain the benefits of the font designs. Some applications need higher precision glyph widths like sub-pixel positioning or “natural width ClearType,” and would reflow if they were changed to bi-level or grayscale rendering. Other applications like Adobe Reader have their own built-in text rendering engine that is independent of the Windows graphics platforms. Likewise, platforms like Java on Windows also use their own rendering techniques.

In some situations with the Windows 7 Explorer, ClearType rendering will remain on so that *Segoe UI* will keep its optimal design. Changing the system font from *Segoe UI* to some other font could be problematic, leading to issues like reflowing dialog box entries, missing text due to wrapping, unlabeled buttons, etc. We know many would value global changes to the fonts used by Windows, but to maintain to reliably across resolutions, DPI, and languages to name a few issues means we cannot have total flexibility on the system font settings at this time.

Given the challenges of turning off ClearType, there are a few mitigations in the fonts to handle some scenarios where ClearType is not available. In the ClearType font *Calibri*, since it is the default font for Microsoft Office, an unusual technique was used to attempt to improve the quality of the font rendering when font smoothing grayscale was selected. In this case, as opposed to the normal situation where font smoothing was disabled at lower text sizes to remove the blur, at these lower sizes the font enabled grayscale in order to improve the character shape. Also, at a few key sizes, the *Calibri* font had some bitmap fonts embedded in the outline file. These bitmaps kick in when bi-level rendering is requested. These bitmaps were intended to handle the case where *Calibri* was being used in a Remote Terminal situation and the default for Remote Terminal was not set to ClearType for performance reasons.

## ClearType Research on Performance

As mentioned earlier, one of the goals behind ClearType is to improve the performance of reading text on computer screens. We have supported several areas of research looking into measuring this work. The research is done at universities and published in peer-reviewed journals. We have another [Microsoft blog](#), that among other things related to fonts, also describes some of the research work on reading performance. Since those blog entries give more detail and background, we'll just describe some of the performance highlights.

- [We've measured](#) an improvement in word recognition accuracy of 17% using ClearType over bi-level rendering.
- [We've found](#) a 5% speed improvement in reading speed and a 2% improvement in comprehension (this is remarkable) using ClearType over bi-level rendering. A 5% reading speed improvement may sound small, but the cumulative effects can be huge given the amount of time people spend reading.
- [We've found](#) the reading speed improvements of 5% continue over longer spans of text, and we've found that non-traditional reading tasks like document scanning are about 8% faster with ClearType over bi-level rendering.
- [We've found](#) that reading sub-optimal text causes eye fatigue by increasing squinting and decreasing the blink rate. (This may seem obvious, but prior to this work there was no understanding of the physiological mechanisms of eye fatigue.)

## ClearType Research on Rendering Preferences

Another research question we've asked ourselves is why do some people prefer bi-level rendering over ClearType? Is it due to hardware issues or is there some other attribute that we don't understand about visual systems that is playing a role. This is an issue that has piqued our curiosity for some time. Our first attempt at looking further into this involved doing an informal and small-scale preference study in a community center near Microsoft. This was done with two identical laptops, one with ClearType and one with bi-level rendering. They were placed side by side and participants were asked which version they preferred. This was done with three different samples. Here were the results:

	Prefer ClearType	Prefer Bi-Level	No Preference

<b>Sample 1</b>	33	1	1
<b>Sample 2</b>	33	2	0
<b>Sample 3</b>	33	2	0
<b>Average %</b>	94%	5%	1%

Comments:

1. 35 participants.
2. Comments for bi-level rendering:  
Washed out; jiggly, sketchy; if this were a printer, I'd say it needed a new cartridge; fading out – esp. the numbers, I have to squint to read this, is it my glasses or it is me?; I can't focus on this; broken up; have to strain to read; jointed.
3. Comments for ClearType:  
More defined, Looks bold (several times), looks darker, clearer (4 times), looks like it's a better computer screen (user suggested he'd pay \$500 more for the better screen on a \$2000 laptop), sort of more blue, solid, much easier to read (3 times), clean, crisp, I like it, shows up better, and my favorite: from an elderly woman who was rather put out that the question wasn't harder: this seems so obvious (said with a sneer.)

Two other additional preference tests were performed with 28 of 30 participants preferring ClearType to bi-level rendering in one study and another with 52 of 55 participants preferring ClearType. Combining these three tests, we get 113 of 120 participants preferring ClearType rendering over bi-level rendering. It is important to note that in a forced preference test like this, just because someone preferred ClearType, it does not mean that they also don't like bi-level rendering. It is just a preference towards ClearType.

Further examination of those who prefer bi-level rendering is of great interest to us and we continue to research this topic and to work with university researchers as well. We expect to see published papers on this topic in the future.

**Future Research**

Going forward, much of our research is in finding ways to make the highest quality text rendering more accessible to everyone. Each visual system has its own characteristics, and just as the ClearType tuner allows us to tune the algorithm for display characteristics, it would also be nice to tune for visual system characteristics. For example, in the United States 7% of the male population is color blind. We believe that we can improve the ClearType algorithm so that text for a colorblind reader is even better than for a reader without colorblindness. Researching ways to improve text rendering for those with high color sensitivity and lower visual acuity would be just as important for us.

**Conclusion**

Making the screen the best possible place to read is an exciting opportunity for us. It blends the engineering challenges of working with many display technologies and human visual systems with the artistic challenge of creating a beautiful set of glyphs, where every subtle typographic nuance is important. In doing this, we need to keep in mind how the science of reading must guide us in making the experience optimal for us—humans. Each rendering technology has advantages and disadvantages for different people; depending on the application in use there are tradeoffs involved. Many of these issues go beyond the ability for people to easily discern choices. Our job is to work hard to provide a great platform for developers as well as tools that people can use to make choices and control how they use their technology. Our goal should be that the out-of-box experience just works. We think that, most of the time, we've accomplished this and we also recognize this area is complex and there is a wide spectrum of feedback.

The team at Microsoft working on these problems has been together since 1990, developing fonts and font-rendering solutions, and working to get a better understanding of the science of reading. The team is made up of engineers, type designers/artists, and psychologists and we work with many other experts throughout Microsoft in attempting to tackle this tough, yet vitally important task. You spend over 80% of the time at the computer reading, so it should be as pleasant an experience as possible. The following article from IEEE Spectrum describes some of the issues we deal with related to [the technology, art, and science of text](#).

--Greg

# Engineering Windows 7 for a Global Market

Steven Sinofsky | [2009-07-07T03:00:00+00:00](#)

*Microsoft has been a global software company for a long time and has always put a lot of effort into engineering our products for a global customer base. It is also an area where the engineering is complex—probably a lot more complex than many might think—and one where we are always trying to learn and improve. Building global software is a responsibility for everyone on the team. We also have feature teams dedicated to developing both global and market specific features—whether it is font handling or doing East Asian language input as two examples. We of course have a significant engineering effort that goes into localizing (“translating” is not quite accurate) Windows into nearly 100 languages. Julie Bennett represents the global development and localization teams and she and John McConnell on her team collaborated across the team to author this post that provides an overview of engineering for a global market. -- Steven*

Many of the readers of the e7 blog are located outside of the United States or speak a language other than English, so we thought it would be useful to share the international and multi-lingual improvements in Windows 7. Our goal for Windows 7 is to deliver exciting features that benefit users worldwide as well as features that make Windows feel local to every user. Like Windows 7's focus to improve the fundamental scenarios of performance and reliability, we improved our processes to allow us to deliver a great customer experience in every language and every country we serve, including delivery of Windows 7 as close to simultaneously as possible worldwide. This blog entry discusses some of the new features and improved processes that we believe make Windows 7 a great worldwide release.

## Features

The international features of Windows 7 are pervasive across the system, from such low-level aspects as the supported characters in NTFS file names (now upgraded to match Unicode 5.1) to such high-level aspects as the selection of backgrounds and themes (now including locally-relevant photos). But there are certain features which are intrinsically critical for proper support of the world's many languages and cultures, and we will describe some of those here.

## Fonts

Language and writing are at the heart of any culture and thus support for fonts is essential to supporting international users. Windows 7 significantly increases both the range and quality of fonts. We have added fifty new fonts:

New Fonts			
Aparajita	Vijaya	Khmer UI Bold	Mangal Bold
Aparajita Bold	Vijaya Bold	Ebrima	Raavi Bold
Aparajita Bold Italic	Vrinda Bold	Ebrima Bold	Shonar Bangla
Aparajita Italic	Lao UI	Microsoft New Tai Lue	Shonar Bangla Bold
Gautami Bold	Lao UI Bold	Microsoft New Tai Lue Bold	Shruti Bold
Iskoola Pota Bold	Segoe UI Light	Microsoft PhagsPa	Tunga Bold
Kalinga Bold	Segoe UI SemiBold	Microsoft PhagsPa Bold	Utsaah
Kartika Bold	Segoe UI Symbol	Microsoft Tai Le Bold	Utsaah Bold
Kokila	Meiryo UI	Segoe Pseudo	Utsaah Bold Italic
Kokila Bold	Meiryo UI Italic	Sakkal Majalla	Utsaah Italic
Kokila Bold Italic	Meiryo UI Bold	Sakkal Majalla Bold	Vani
Kokila Italic	Meiryo UI Bold Italic	Gabriola	Vani Bold
Latha Bold	Khmer UI		

As you might guess from the font names in the above table, many of the new fonts are for non-Latin scripts. In fact, Windows 7 will be the first version of Windows to ship with more fonts for non-Latin scripts than for Latin-based scripts. One major area of improvement is for the languages of India. To the nine (9) fonts for Indian languages that shipped in Vista, Windows 7 adds forty (40) more. Windows 7 will now include multiple fonts (often in multiple weights) for each of the official languages of India.

संकेत चिह्न निर्माण  
संकेत चिह्न निर्माण  
संकेत चिह्न निर्माण  
संकेत चिह्न निर्माण

**Aparajita: A New Devanagari Font in Regular, Bold, Italic and Bold-Italic**

Besides new fonts, we have also improved many of the existing fonts. For example, we have added over two thousand (2,000) glyphs to Consolas, Calibri, Cambria Bold, and Cambria Math. But the most dramatic improvements have been to some of the non-Latin scripts. For example, Windows 7 does a much better job rendering the common Lam-Alef ligature in Arabic (see the illustration below) and in the placement of vowel marks.

وَاللَّامِي ۚ وَاللَّامِي ۚ

**Left: Lam-Alef Ligature in Vista Right: Lam-Alef Ligature in Windows 7**

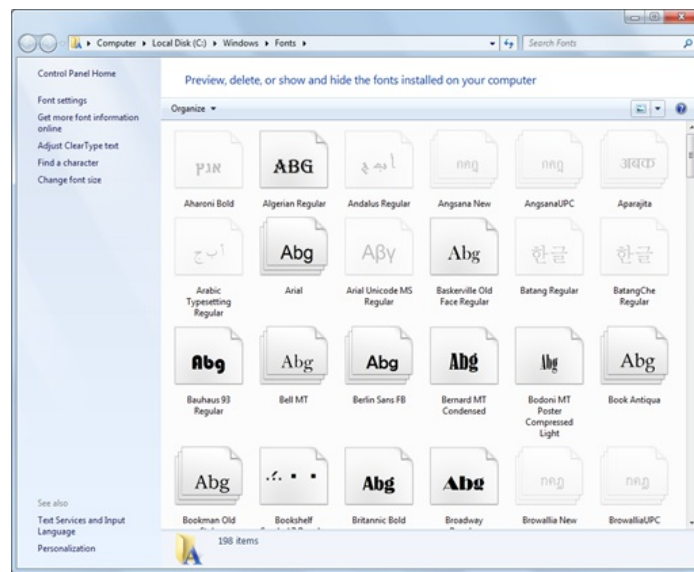
Changes to fonts (even clear improvements) are always tricky because of backwards compatibility issues. For example, if a character changes width or position, it may cause existing documents to reflow (repaginate), which is unacceptable. Therefore, whenever we change a font, we must run extensive verification tests against the changes to ensure the font metrics and other tables are unchanged. In the case of the Lam-Alef fix shown

above, we discovered that there were existing applications that relied on the (undocumented) order of the glyphs within the old font. These applications would break if we simply replaced the glyphs. The font team worked closely with the international application compatibility team to ensure that changes we made did not affect the order of glyphs within the font, thus providing backward compatibility.

## Font Control Panel

With so many new and expanded fonts for Windows 7, we also wanted to help users manage their fonts more easily. For the first time in years, we have done a complete overhaul of the font control panel.

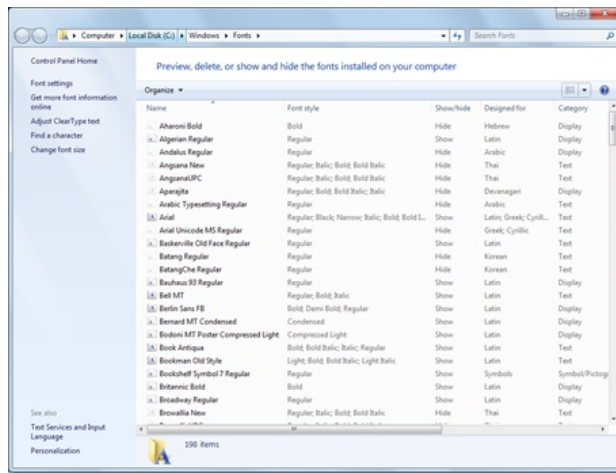
The first picture below shows the font control panel with the large icon view. The most obvious change is that the font icons now convey much more information about the appearance of the font. The content of the icon gives a hint as to the glyph repertoire of the font. The style of the icon matches the style of the font. Non-Latin fonts show typical glyphs from the script for the font to see how it is designed. A more subtle change is that some font icons are faded to indicate fonts that are installed, but hidden. Hidden fonts will not show by default in the ribbon and font dialogs. Users can now use the font control panel to tune the fonts that they regularly use. By hiding fonts they never use, users can simplify choosing the correct font within applications. By default, only fonts supporting languages that can be written with the users installed input locales (keyboard layout plus language) will be shown. For example, users with English and French input locales will see only the Latin fonts, whereas users with the Japanese input method installed will see only the Japanese fonts. Users can override these defaults by right-clicking on any of the fonts in the control panel. Hidden fonts are still installed so an existing application that uses a hidden font will behave identically.



Font Control Panel with Large Icon View

The next picture below shows the font control panel with the detailed view. Now users can see much more information about the font. For example, the user can sort fonts by style, whether they are hidden, and information about the creator of the font. Font files generally contain information only in the design language of the font (e.g. a Japanese font might contain only information in Japanese). In Windows 7, we needed a solution that would work for all languages and for all fonts, so we created a hybrid approach that combines information from the font itself with metadata (an XML file that provides the information about the fonts on the system).

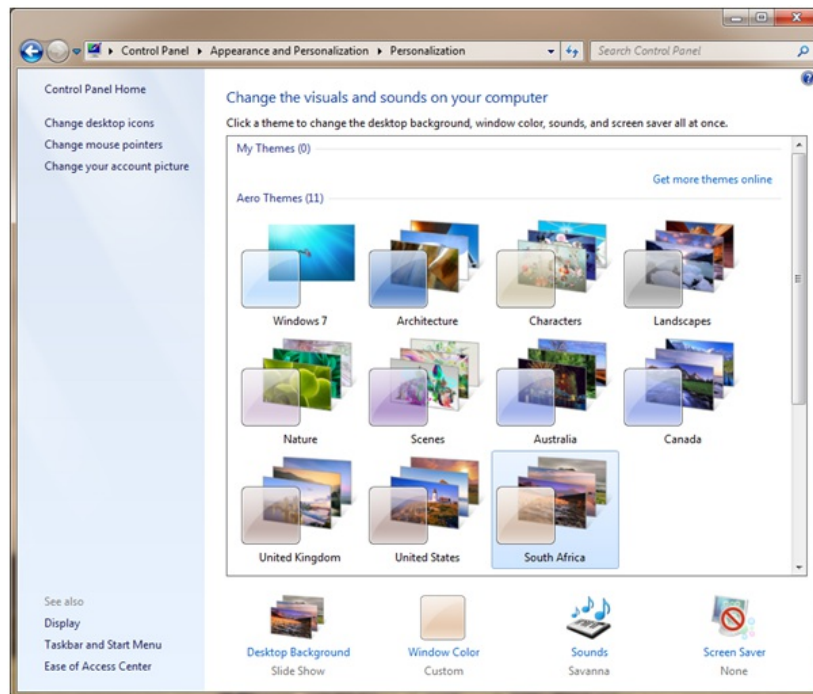




**Font Control Panel with Detail View**

## Local Packs

Windows 7 has increased [opportunities for personalization](#). New themes, backgrounds, and sounds make it easy to customize Windows 7 to match your personality. To the extent that our preferences are influenced by our language and location, Windows 7 reflects this with the introduction of Local Packs. Local Packs provide customized Windows 7 visual themes for a specific region. These visual themes contain locally relevant wallpaper images, custom aero glass colors, and regional sound schemes. Windows® Internet Explorer® Favorites and RSS feeds may also be updated when the Local Pack is activated on an end user's computer. For example, adding and enabling the Local Pack for France will add a market-customized theme for France to the end user's Personalization control panel and a number of links to useful French Public Sector websites and RSS Feeds to the user's profile.



### Customized Themes in the Personalization control panel

The Local Pack content provides users with seamless local experiences right out of the box. Users are never exposed to Local Packs per se, they just select their Location as normal during Windows Welcome, and appropriate local content is exposed to them based on that setting.

Users looking for visual themes for other countries, or indeed any other areas of interest, can find them on the Windows Online Gallery, which is accessible via the [“Get more themes online”](#) link in the Personalization control panel.

### Other Features

Other new features include five (5) new locales (bringing the total number of locales supported to two hundred and ten (210)), twelve (12) new input locales, and improvements to sorting for traditional Chinese characters. Also, we have generally updated our system databases to the latest version of the Unicode Standard (5.1). There are also interface improvements that should allow developers to create better globalized applications. Extended Linguistic Services (ELS) is a cool new feature we describe below in the International Timeliness and Quality section.

Perhaps one of the most important improvements outside the core international features has been in Search, which now recognizes more languages. For example, Windows 7 desktop search now recognizes Russian morphology (the rules for single and plural, tenses, and case). This means that searches for a particular word in Russian will now match not only that exact word, but also the common variations of the word, yielding significantly better results.

### International Timeliness and Quality

In previous versions of Windows, final delivery of every language to every market took several months. For Windows 7, we changed how we worked on international releases to significantly shorten this delta so that all users worldwide can enjoy Windows as simultaneously as possible.

This goal had far reaching implications on how we perform our work as engineers and on how we interact with partners and customers during our public testing phases.

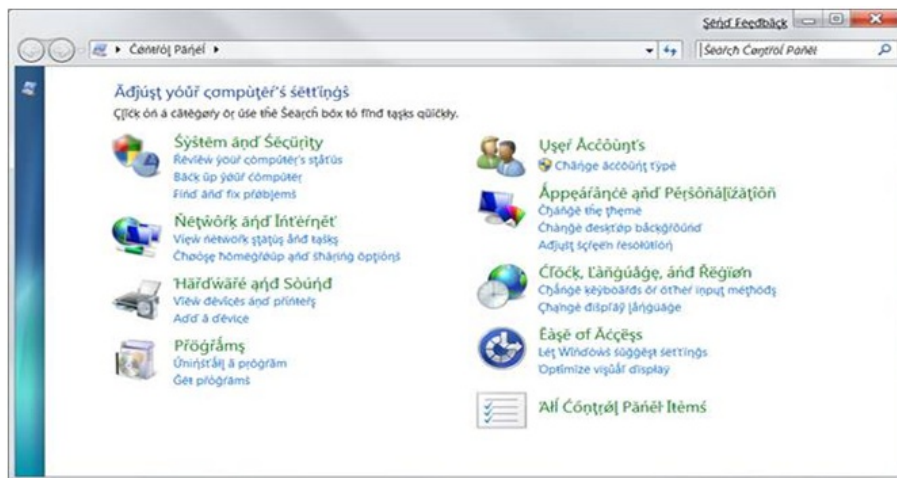
To understand our approach, we should first explain two important concepts: localization and globalization.

Localization is the process of adapting the user experience into another language. Beyond the translation of strings, it can also include activities such as resizing dialogs and mirroring icons for right-to-left languages, such as Hebrew and Arabic. Localization bugs, such as the mistranslation of a menu item, are defects introduced during this process.

Globalization, on the other hand, is the process of producing a product that works well in every country no matter the user interface language setting. A globalization bug may be as simple as showing a UI element in the wrong language and as complex as not properly handling right-to-left scripts. Globalization bugs are inherently more serious than localization bugs as they usually affect many or all languages and often require re-thinking the technical design. In past Windows releases, repairing globalization bugs contributed to the necessity of the long release deltas. For Windows 7 we worked to prevent, find, and fix globalization bugs as early in the development process as possible.

### *Pseudo-Localization*

To prevent common globalization bugs, pseudo-localized builds were created. Pseudo-localization is a process that creates a localized product in an artificial language. That language is identical to English except that each character is written with a different character that visually resembles the English character. Except for being entirely machine generated, we create the pseudo-localized builds exactly the same way as we create the localized builds. Because even monolingual US software developers can read pseudo-localized text, it has proven to be an excellent way to find globalization problems early in the development cycle. In the Windows 7 beta, some UI elements were still in their pseudo-localized form, causing some interesting theories about what the meaning might be. We hope we have solved the mystery with this blog post. ??



Control Panel Dialog in Pseudo-localized Windows 7

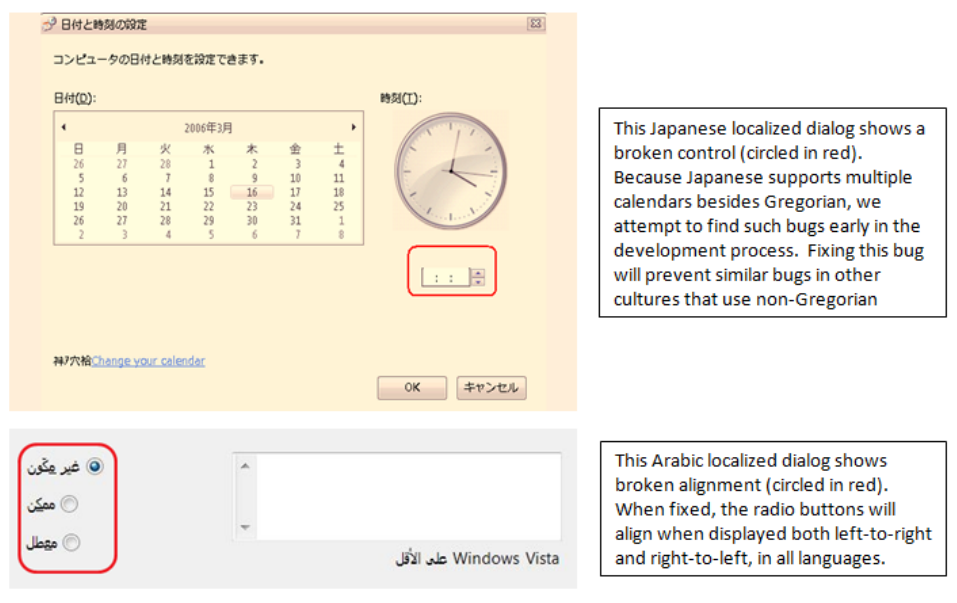
### **Pilot Languages**

Beta is always an exciting time for us as it is our first real chance to hear from you about our efforts. We are thrilled that people from over one hundred and thirteen (113) countries downloaded the Windows 7 Beta. With such a large and diverse beta program, we must have highly scalable processes to gather and incorporate your feedback. In Windows 7, we are very excited about some new approaches we took here.

In the past, localization languages for Windows beta releases were selected for a mix of pragmatic reasons. While this ad hoc approach had benefits, too often we found that serious globalization defects were not reported because they did not manifest in the chosen languages. For the Windows 7 Beta, our priority was to find globalization bugs and therefore we have concentrated on four languages (plus English) that experience has shown are most likely to find specific types of defects:

- **German** - Because it contains some very long words, German can reveal dialog size and alignment defects better than other languages.
- **Japanese** - With tens of thousands of characters, multiple non-Latin scripts, alternative input method engines, and an especially complex orthography, Japanese is a great way to find defects that affect many East Asian languages.
- **Arabic** - Written right-to-left and with contextual shaping (character shape depends on adjacent characters), including this language in the Beta helped us test code paths not exercised by German and Japanese.
- **Hindi** - Windows 95 and Windows 98 never supported Hindi and support for this language relies entirely on Unicode. Testing Hindi helps find legacy (non-Unicode) defects that affect all such languages.

By concentrating on these four languages during Beta, we maximized our chances to find and fix the globalization bugs that affect many languages. This in turn gave us more time to improve the localization of all languages before we release the actual product. The pictures below show two bugs found during Beta that illustrate the advantages of focusing on these pilot languages.



### Globalization Defects Found During Windows 7 Beta

In addition to our goal of finding globalization bugs via these languages, we also asked some of our OEM customers to provide feedback on the language aspects within their manufacturing processes. Since many of the OEMs are located in East Asia, we also localized Windows 7 Beta for Simplified Chinese, Traditional Chinese, and Korean.

## RC Language Packs

In part because of the engineering process improvements described above, we were able to deliver more language packs for Windows 7 RC than we have ever been able to do in the past for Windows. For those of you running the Ultimate version of Windows 7 RC, you will have noticed the following thirty-two (32) Language Packs available for download on Windows Update:

Arabic	German	Portuguese (Portugal)
Bulgarian	Greek	Romanian
Chinese (Simplified)	Hebrew	Russian
Chinese (Traditional)	Italian	Serbian (Latin)
Czech	Japanese	Slovak
Danish	Korean	Spanish
Dutch	Latvian	Swedish
English	Lithuanian	Thai
Estonian	Norwegian	Turkish
Finnish	Polish	Ukrainian
French	Portuguese (Brazil)	

### 32 Windows 7 RC Language Packs on Windows Update

One thing we will do differently in the future is to ensure that all languages available at Beta are also available at RC (e.g. not including Hindi for Windows 7 RC). We will correct this for future versions.

## Understanding feedback from around the world

With Windows 7 beta localized into five languages and globally enabled for hundreds more, we received beta bugs from customers all over the world. We rely on these bug reports to help us improve Windows 7, so we devote much time to reading customer bug reports to determine product issues. Because bugs come from worldwide customers *in many languages*, we look for ways not only to understand their feedback, but also to address it as quickly as possible. The faster we can understand the issue, the better chance we have of addressing the feedback. As we receive bug reports in all the many languages that our customers speak, this has sometimes posed quite a challenge.

In the past, we have handled multilingual bug reports using manual processes, where individual bugs were examined and then manually translated one-by-one for appropriate follow-up by the feature team that owned the affected component. This is a time-consuming and error-prone exercise that scales poorly to a program as large and diverse as the Windows 7 beta. In the worst case, valuable international feedback has missed the window to affect the final product, and thus slipped to a Service Pack or subsequent release.

In Windows 7, by using the language detection API in the new Extended Linguistic Services (ELS), we have been able to automatically detect the language of customer bugs as they are reported. ELS functionality is new for Windows 7 and available to any developer who wants to leverage advanced linguistic functionality in the operating system. Beginning in Windows 7, developers may use ELS to provide language and script detection of any Unicode text, as well as transliteration to map text between writing systems. To use these Windows 7 services and all further services that we will add in subsequent releases, developers need only to learn one simple and unified interface. The ability to detect over one hundred (100) languages is available for all Windows 7 application developers, and we are happy to be able to apply this functionality to triage and handle beta feedback you send us from around the world. We use our own international developer functionality to improve our ability to respond to customer issues globally.

Once we have detected the language, we take the resulting text and use the machine translation support that is available online from [Live Translator](#). This allows us to translate the text to English to get a sense of your feedback. Our engineers can then search our feedback database for specific features or areas of functionality. This also helps us in our efforts to ensure international application compatibility, as we can learn about potentially problematic international application experiences as soon as customers report them. Machine translation does not provide a perfect translation, but it does allow us to determine which issues might require further investigation. This in turn allows us to hear and respond to customer

issues with a much faster turnaround time than we have had in previous releases, which means better quality in Windows 7 when we release it to the world.

By the end of Windows 7 Beta, we had used this process to translate 35,408 issues and comments submitted using the Feedback tool.

## Putting it all together

The end result of the work to improve globalization and localization quality is reflected in the announcement that all fully localized releases of Windows 7 will be available within two weeks of the initial release wave with all languages available in October. We hope (and believe!) end users will find the overall quality of these releases to be the best ever.

Arabic	French	Portuguese (Brazil)
Bulgarian	German	Portuguese (Portugal)
Chinese (Hong Kong S.A.R.)	Greek	Romanian
Chinese (Simplified)	Hebrew	Russian
Chinese (Traditional)	Hungarian	Serbian (Latin)
Croatian	Italian	Slovak
Czech	Japanese	Slovenian
Danish	Korean	Spanish
Dutch	Latvian	Swedish
English	Lithuanian	Thai
Estonian	Norwegian	Turkish
Finnish	Polish	Ukrainian

### 36 Windows 7 language releases available in October 2009

In addition to the 36 languages that will be released in October, there will be additional languages available for download as Language Interface Packs (LIPs) onto any Windows 7 edition as part of the Local Language Program (LLP). The LLP is a partnership with governments, universities, and language experts from around the world. (You can find more information on the LLP at <http://www.microsoft.com/unlimitedpotential/programs/lip.mspx>.) Work on a LIP starts at RTM and continues for many months based on the schedules of our partners. Two (2) LIPs will be available for download when Windows 7 is available in October – Catalan and Hindi. Additional LIPs will become available for download over the following months based on the schedules of our partners. We are happy to have improved the delivery time of the first 38 languages (36 + 2 LIPs) and recognize that future releases are an opportunity to improve further. Creating a track record of dependable release schedules on our part will help everyone around the world plan better for a more unified release timeline.

More information about Extended Linguistic Services (ELS) and other cool new features of Windows 7 are available on-line on MSDN. In particular, you can download the Windows SDK for Windows 7 and read about what is new in the 'International' section. Also, the new [Go Global Developer Center](#) on MSDN has a wealth of information about international technologies.

If you want to send us feedback, please comment on this blog entry or use the Feedback button in Windows 7. We love to hear from you (in any language).

-- Windows International Team

# Our Next Engineering Milestone: RTM

Steven Sinofsky | [2009-07-22T16:40:00+00:00](#)

---

Today marks an important milestone in the Windows 7 project. The Windows 7 team is proud to share with you that a short while ago we have started to release Windows 7 to PC OEM and manufacturing partners. This means our next major milestone will be the availability of PCs loaded with Windows 7 and store shelves stocked with Windows 7 on October 22, 2009.

This is a milestone we could not have achieved without the broad participation across the PC Ecosystem we have talked so much about on this blog. Windows 7 is a product not just of Microsoft, but of a whole industry of partners of all kinds. Throughout the development of Windows 7 we've seen an incredible engagement from so many people that have contributed to making the Windows 7 engineering project one we, collectively, feel good about. The feedback and collaboration throughout the development of Windows 7 has been outstanding and valuable beyond measure. This work has created the kind of experience so many of you have talked about in this blog—the ability to use a broad range of PC hardware and peripherals with a great setup and out of box experience. On behalf of the Windows team and all of the successful installations and device connections, please let me extend an incredible “thank you” to all of our hardware partners who have done such excellent work.

Windows 7 has also been one of the most broadly and deeply tested releases of software we have ever had. Starting with a pre-beta in October of 2008 with a few thousand developers using Windows 7 at the earliest stages, through the Beta, and then the Release Candidate in May when we have had millions of people successfully running the product (and many on multiple PCs). As we have discussed in this forum, the ongoing depth usage of Windows 7 along with the breadth and variety of hardware and software configurations has provided (and will continue to provide) the key tools to make sure we continue to deliver ever-improving Operating System quality.

In developing Windows 7 we also set out to have a great dialog with you, perhaps our strongest critics and our biggest supporters. We know you expect a lot from Windows 7 and you demand a lot from the team that builds your OS. This blog has helped to bring significant issues and important decisions to light and we have debated them—here and elsewhere. Along the way we have definitely learned a lot about working together and also about many specific issues that are important to you. We have worked hard to find the right balance across many diverse points of view and we hope you share our feeling that we've done a good job at being open, honest, and transparent in how we have approached engineering Windows 7. The conversations we had on this blog have been a memorable part of developing Windows 7 and in our hallways, in Redmond and around the world, we've spent collectively thousands of hours discussing and acting on the feedback you have provided here.

While we have reached our RTM Milestone, no software project is ever really “done”. We will continue to monitor and act on the real world experience with Windows 7—we've used the Beta and RC process to test out our servicing and we have every intent of doing a great job on this important aspect 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 7's new APIs and features. Software developers will have lots of new software to show off as well. All of this is yet to come and is very exciting.

Software projects on the scale of Windows are pretty rare and our team has a lot of pride, and as we have said so many times, is humbled by the responsibility. We are going to continue to learn and continue to improve how we engineer our products, with the aim of being the very best engineers we can be and delivering the very best OS for the world's varied customers. Being an engineer is about learning and that learning comes from the experience gained in designing and delivering each release. Together we've learned and together we've engineered a wonderful product.

We know there are lots of questions about how to get Windows 7 and when, and of course more questions to come about exploring and using the full set of Windows 7 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 7 there.

The final few minutes before RTM are a sign-off process where each and every team that contributed to Windows formally commits to having successfully executed the work necessary for the product to be in the release process. We gather one last time (for Windows 7) in the “Ship Room” and a representative from each team signs (literally) and signifies their team's readiness for manufacturing. We thought we'd share this moment with you here today.



On behalf of the Windows 7 engineering team we want to thank you very much for your contributions throughout development and your contributions yet to come to Windows 7. THANK YOU!

Next stop, October 22, 2009!

--The Windows 7 team



# The Windows Feedback Program

Steven Sinofsky | [2008-09-10T03:00:00+00:00](#)

*Introducing Christina Storm who is a program manager on the Windows Customer Engineering feature team working on telemetry.*

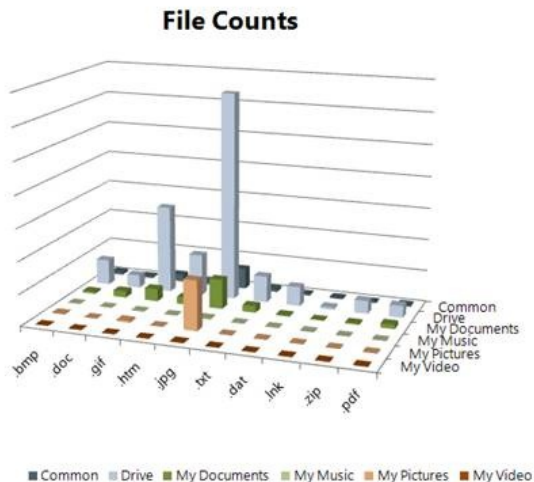
In a previous article Steven has introduced the Windows 7 Feature Teams. I am a [program manager](#) working on telemetry on the Windows Customer Engineering team. Our team delivers the Windows Feedback Program, one of several feedback programs in place today that allow us to work directly with customers and make them part of our engineering process.

The Windows Feedback Program (WFP) has been active for several years during the Windows XP and Windows Vista product cycles, and we are currently ramping up to get all aspects of this program ready for Windows 7. At the core of this program is a large research panel of customers who sign up via our website <http://wfp.microsoft.com> during open enrollment. Customers choose to be part of a survey program, an automated feedback program or both. They then complete a 20-minute profiling survey, which later allows us to look at their feedback based on their profile. We have customers spanning a wide spectrum of computer knowledge in our program, and we are constantly working on balancing the panel to staff up underrepresented groups. The majority of customers who are spontaneously willing to participate in a feedback program like ours are generally enthusiastic about technology. They are early adopters of consumer electronics, digital devices and new versions of software. In contrast, customers who see the PC as a tool to get a job done tend to be a bit more reluctant to join. And we also need more female participants!

Customers who participate in the automated feedback program install a data collection tool developed by the Windows Telemetry Team. The [privacy agreement](#) of this program describes the data collections our tool performs. Here are a few examples:

- Windows usage behavior including installed and used applications.
- File and folder structures on your computer, including number of types of files in folders, such as number of jpg files in the Pictures folder.
- System-specific information, such as hardware, devices, drivers, and settings installed on your computer.
- [Windows Customer Experience Improvement Program \(CEIP\)](#) data.

From the collected data we create reports that are consumed by a large number of Windows feature teams as well as planners and user researchers. This chart below shows the answer to the following question: What is the most common file type that customers who participate in our program store on their PCs and what are the most popular storage locations?



The results help us both with planning for handling the volumes of data customers store on their PCs as well as mimicking real-life scenarios in our test labs by setting up PCs with similar file numbers and file sizes and distribution of files on the PCs.

These data collections furthermore allow us to create reports based on profiled panelists. The above chart may look different if we created it based on data delivered only by developers and then compare it to data delivered only by gamers, just to name a couple of different profiles that participate in our program. The Windows knowledge level sometimes makes a difference, too. Therefore it is very important to us that customers participate in this program who consider themselves Windows experts as well as customers who don't enjoy spending too much time with the PC, who just see the PC as a tool to get things done. Based on the data, we may decide to optimize certain functionality for a particular profile.

In general, we utilize this data to better understand what to improve in the next version of Windows. Let's take a look at how the representative sample has their monitors configured. First what resolutions do customers run with on their PCs? The following table lists typical resolutions and the distribution based on the Windows Customer Experience Improvement Program, which samples all opt-in PCs (Note the numbers do not add to 100% because not every single resolution is included):

Monitor Default Resolution	% of Total Sample
1280X1024	26%
1400X1050	6%
1600X1200	20%
1680X1050	8%
1920X1080	1%
1920X1200	9%

*Approximately 10% of customers have HD or higher resolution.*

One thing you might notice is that about 10% of consumers are running with HD or greater resolution. In some of the comments, people were asking if our data represented the "top" or "power users". Given this sample size and the number of folks with industry leading resolution I think it is reasonable to conclude that we adequately represent this and all segments. This sample is a large sample (those that opt-in) of an extremely large dataset (all Windows customers) so is statistically relevant for segmented studies,

We have found that a large percentage of our program participants lower their display resolution from the highest usable for their display. Looking at the data coming from the Windows Customer Experience Improvement Program to compare to, and noticed a similar trend: over 50% of customers with 1600x1200 screen resolution displays are **adjusting** their resolution down to 1024x768, likely because they find it uncomfortable to read the tiny text on high resolution displays. The negative effect of this resolution change is the loss of fidelity to the point where reading text in editors and web browsers is difficult. High definition video content also won't be able to render properly.

Here is the data just for customers with displays capable of 1600x1200:

chosen resolution	%
640X480	1%
800X600	7%
1024X768	57%
1280X1024	3%
1600X1200	32%
Total	100.00%

*68% of customers with a 1600x1200 display lower their resolution*

In a future blog post, one of the program managers from the Windows Desktop Graphics team is going to describe what we have done with that information to improve display quality and reading comfort in Windows 7.

We also frequently use our data to select appropriate participants for a survey. A researcher may be interested in sending out an online survey only to active users of virtual machine applications. We would first determine that group of users by looking at our "application usage" data and then create the list of participants for the researcher. Sometimes we combine automatically collected data with survey responses to analyze the relationship between a customer's overall satisfaction and their PC configuration.

At the current point in time, 50% of our panel participants are using Windows XP and 50% are using Windows Vista. We are currently not offering open enrollment. If you are interested in being invited to this program, please send an email to [winpanel@microsoft.com](mailto:winpanel@microsoft.com) indicating "Notify me for enrollment" in the subject line. If you'd like to add a bit of information about yourself, including your Windows knowledge level, that would be much appreciated! We will add you to our request queue and make our best effort to invite you when we have capacity.

When we release the Windows 7 beta we will also be collecting feedback from this panel and asking for participation from a set of Windows 7 beta users. Our current plans call for signing up for the beta to happen in the standard Microsoft manner on <http://connect.microsoft.com>. Stay tuned!

-- Christina Storm

# What we do with a bug report?

Steven Sinofsky | [2009-08-10T03:00:00+00:00](#)

---

This has been a busy couple of days for a few of us on the team as we had a report of a bug in Windows 7. The specifics of the issue are probably not as important as a discussion over how we will manage these types of situations down the road and so it seems like a good time to provide some context and illustrate our process, using this recent example.

This week a report on a blog described a crashing issue in Windows 7. The steps to reproduce the crash were pretty easy (1) run `chkdsk /r` on a non-system drive then crash after consuming system memory. Because it was easy to “reproduce”, the reports of this issue spread quickly. Subsequent posts and the comments across the posts indicated that the issue seemed to have been reproduced by others—that is the two characteristics of the report were seen (a) consumption of lots of memory and (b) crashing.

Pretty quickly, I started getting a lot of mail personally on the report. Like many of you, the first thing I did was try it out. And as you might imagine I did not reproduce both issues, though I did see the memory usage. I tried it on another machine and saw the same behavior. In both cases the machine functioned normally during and after the `chkdsk`. As I frequently do, I answered most of the mail I receive and started asking people for steps to reproduce the crash and to share system dump files. The memory usage did not worry me quite as much as the crash. I began having a number of interesting mail threads, but we didn’t have any leads on a repro case nor did we have a crash dump to work with.

Of course I was not the first Microsoft person to see this. The file system team immediately began to look into the issue. They too were unable to reproduce the crash and from their perspective the memory usage was by design and was a specific Windows 7 change for this scenario (the `/r` flag grabs an exclusive lock and repairs a disk and so our assumption is you’d really like the disk to be fixed before you do more stuff on the machine, an assumption validated by several subsequent third party blog posts on this topic). We cast the net further and continued looking for crash dumps and reports. As described below we have quite a few tools at our disposal.

While we continued to investigate, the mail I was getting was escalating in tone and more importantly one of the people I responded to mentioned our email exchange in a blog post. So in my effort to have a normal email dialog I ended up in the thick of the discussion. As I have done quite routinely during the development of Windows 7, I added a comment on the original blog (and the blog where this particular email friend was commenting) outlining the steps we are taking and the information we knew to date. Interestingly (though not unfortunately) just posting the comment drew even more attention to the issue being raised. I personally love being a member of the broader community and enjoy being a casual contributor even when it seems to cause a bit of a stir.

It is worth just describing the internal process that goes on when we receive a report of a crashing issue. Years ago we had one of two reactions. Either we would just throw up our arms and surrender as we had no hope of finding the bug, or we would drop everything and start putting people on airplanes with terminal debuggers in the hopes of finding a reproducible case. Neither of these is particularly effective and the latter, while very heroic sounding, does not yield results commensurate with effort. Most importantly while there might be a crash, we had no idea if that was the only instance or if lots more people were seeing or would see the crash. We were working without any data to inform our decisions.

With the internet and telemetry built into our products (not just Windows 7) we now have a much clearer view of the overall health of the software. So when we first hear a report of a crash we check to see if we’re seeing the crash happen on the millions of machines that are out there. This helps us in aggregate, but of course does not help us if a crash is one specific configuration. However, a crash that is one specific configuration will still show up if there is any statistically relevant sampling of machines and given the size of the user base this is almost certain to be the case. We’re able to, for example, query the call stacks of all crashes reported to see if a particular program is on the stack.

We have a number of tools at our disposal if we are seeing a crash in telemetry. You might have even seen these at work if you crash. We can increase (with consent) the amount of data asked for. We can put up a knowledge base article as a response to a crash (and you are notified in the Windows 7 Action Center). We can even say “hey call us”. As crazy as that one might sound, sometimes that is what can help. If a crashing issue in an already shipping product suddenly appears then something changed—a new hardware device, new device driver, or other software likely caused the crash to appear far more frequently. Often a simple confirmation of what changed helps us to diagnose the issue. I remember one of the first times we saw this was when one day unexpectedly Word started crashing for people. We hadn’t changed anything. It turned out a new version of a popular add-in released and the crash was happening in the add-in, but of course end-users only saw Word crashing. We quickly put up instructions to remove the add-in while in parallel working with the ISV to push out a fix. This ability to see the changing landscape, diagnose, and respond to a problem has radically changed how we think of issues in the product.

We are constantly investigating both new and frequently occurring issues (including crashes, hangs, device not found, setup failures, potential security issues, and so on). In fact we probably look into on the order of hundreds of issues in any given month as we work with our enterprise and OEM customers (and therefore hardware partners, ISVs, etc.). Often we find that issues are resolved by code changes outside core Windows code (such as with drivers, firmware, or ISV code). This isn't about dodging responsibility but helping to fix things at the root cause. And we also make many code changes in Windows, which are seen as monthly updates, hotfixes, and then service pack rollups. The vast majority of things we fix are not applicable broadly and hence not released with immediate urgency—if something is ever broadly applicable we will make the call to release it broadly. It is very important for everyone to understand how seriously we take the responsibility of making sure there are no critical issues impacting a broad set of customers, while also balancing the volume of changes we push out broadly.

To be specific about the investigation around the `chkdsk` utility, let's look at how we dove into this over the past couple of days. We first looked through our crash telemetry (both at the user level and "blue screen" level) and found no reported crashes of `chkdsk`. We of course look through our existing reports of issues that came up during the development of Windows 7, but we didn't see anything at all there. We queried the call stacks of existing reported crashes (of all kinds, since this was reported) and we did not find *any* crashes with `chkdsk.exe` running while crashing. We then began automated test runs on a broad set of machines—these ran overnight and continued for 2 days. We also saw reports related to a specific hardware configuration, so we set up over 40 machines based on variants of that chipset, driver, and firmware and ran those tests. We were not hitting any crashes (as mentioned, the memory usage was already understood). Because some were saying the machines were non-responsive we also looked for that in manual tests and didn't see anything. We also broadened this to request globally to Microsoft folks to try things out (we have quite a few unique configs when you think of all of our offices around the world) and so we had several hundred more test runs going. We also had reports of the crash happening when running *without* a pagefile—that could be the case, but that would not be an issue with this utility as any program that requests more memory than physically available would cause things to tip over and this configuration is not recommended for general purpose use (and this appears to be the common thread on the small number of non-reproducible crashes). Folks interested might read Mark's [blog](#) on the topic of pagefiles in general. While we did not identify anything of note, that does not rule out the possibility of a problem but at this point the chances of any broad issue are extremely small.

In the meantime, we continue to look through external blogs, forums and other reports of crashes to see if we can identify any reproducible cases of this. While we don't contact everyone, we do contact people if the forum and report indicate this has a good chance of yield. In all fairness, it probably doesn't help us when there's a lot of "smoke" while we're trying to find the fire. We had a lot of "showstopper" comments piling on but not a lot of additional data including a lack of a reproducible case or a crash dump.

This type of work will continue until we have satisfied ourselves that we have systematically ruled out a crash or defined the circumstances where a crash can happen. Because this is a hardware/software related issue we will also invite input from various IHVs on the topic. In this case, because it is disk related we can't rule out the possibility that in fact the disk was either failing or about to fail and the excessive use of the disk during a /r repair would in fact generate a failure. And while the code is designed to handle these failures (as you can imagine) there is the possibility that the specific failure is itself not handled well. In fact, in our lab (running tests continuously for a few days) we had one failure in this regard and the crash was in the firmware of the controller for the disk. Obviously we'll continue to investigate this particular issue.

I did want folks to know just how seriously we take these issues. Sometimes blogs and comments get very excited. When I see something like "showstopper" it gets my attention, but it also doesn't help us to have a constructive and rational investigation. Large software projects are by nature extremely complex. They often have issues that are dependent on the environment and configuration. And as we know, often as deterministic as software is supposed to be sometimes issues don't reproduce. We have a pretty clear process on how we investigate reports and we focus on making sure Windows remains healthy even in the face of a changing landscape. With this post, I wanted to offer a view into some specifics but also into the general issue of sounding alarms.

It is always cool to find a bug in software. Whether it is an ATM, movie ticket machine, or Windows we all feel a certain sense of pride in identifying something that doesn't work like we think it should. Windows is a product of a lot of people, not just those of us at Microsoft. When something isn't as it should be we work with a broad set of partners to make sure we can effectively work through the issue. I hope folks recognize how serious we take this responsibility as we all know we're going to keep looking at issues and we will have issues in the future that will require us to change the code to maintain the level of quality we know everyone expects of Windows.

--Steven

# Windows 7 Battery Notification Messages

Steven Sinofsky | [2010-02-08T12:15:00+00:00](#)

---

Over the past week we have seen a little bit of blogosphere activity regarding Windows 7 and batteries, specifically the new Windows 7 message “Considering replacing your battery?”. Since this is related to the engineering of Windows 7 we’re going to use this blog to provide an update to people. As we have talked about many times, we have a relentless focus on the quality of Windows 7 and we take seriously any reports we receive that indicate a potential problem that could result in a significant failure of the OS. In a [previous post](#) we talked about the steps we take when we receive a bug report, in particular when we start to see several reports that appear to be the same. For the past week or so we have been diligently working through these steps and more to see if there is anything in Windows 7 we need to address regarding this issue. At this time we have no reason to believe there is any issue related to Windows 7 in this context.

Several press articles this past week have drawn attention to blog and forum postings by users claiming Windows 7 is warning them to “consider replacing your battery” in systems which appeared to be operating satisfactorily before upgrading to Windows 7. These articles described posts in the support forums indicating that Windows 7 is not just warning users of failing batteries – as we designed Windows 7 to do this – but also implying Windows 7 is falsely reporting this situation or even worse, causing these batteries to fail. To the very best of the collective ecosystem knowledge, Windows 7 is correctly warning batteries that are in fact failing and Windows 7 is neither incorrectly reporting on battery status nor in any way whatsoever causing batteries to reach this state. In every case we have been able to identify the battery being reported on was in fact in need of recommended replacement.

Using all the tools at our disposal including contacting customers reporting this issue on forums, customer service communications, partnerships with our PC makers, and of course the telemetry in Windows 7, we have been monitoring reports and discussions regarding this new feature, trying to separate reports of the designed behavior from those that might indicate an issue with Windows 7. In the latter cases we are trying to understand the scope of applicability and obtain hardware on which to reproduce a faulty behavior. To date all such steps indicate that we do have customers seeing reports of battery health issues and in all cases we have investigated Windows 7 has simply accurately detected a failing battery. Before I go into our status on this particular issue, we should review the details behind this new feature.

One of the most obvious components of PC battery life (the runtime you get on battery power) is the battery itself. PC batteries inherently degrade in their ability to hold a charge and provide power (as is the case for all rechargeable batteries). The cause of this is complex and includes irreversible changes in battery chemistry, and increased internal resistance among other things and those in turn are dependent on the design and manufacturing of the battery. This degradation translates into less battery life for the user over the life of the battery in the PC. Ultimately, batteries must be replaced to restore an acceptable battery life. A quick check of mainstream laptops will show that batteries usually have a warranty of 12 months, which is about the length of time when statistically we expect to see noticeable degradation (meaning that you start to notice the need to charge more frequently). Those of us that have owned the same laptop (or mobile phone, or music player, or anything else with rechargeable batteries) for a couple of years and taken it through regular charge cycles have no doubt “felt” the decline in battery life though we might have attributed to any number of factors since we did not have any information available to us otherwise.

Windows 7 makes use of a feature of modern laptop batteries which have circuitry and firmware that can report to Windows the overall health of the battery. This is reported in absolute terms as Watt-hours (W-hr) power capacity. Windows 7 then does a simple calculation to determine a percentage of degradation from the original design capacity. In Windows 7 we set a threshold of 60% degradation (that is the battery is performing at 40% of its designed capacity) and in reading this Windows 7 reports the status to you. At this point, for example, a battery that originally delivered 5 hours of charge now delivers, on average, approximately 2 hours of charge. The Windows 7 notification is a battery meter icon and notification with a message “Consider replacing your battery?”. This notification is **new to Windows 7 and not available in Windows Vista or Windows XP.**



Battery meter showing a notification that a battery might be going bad

PC batteries expose information about battery capacity and health through the system firmware (or BIOS). There is a detailed specification for the firmware interface (ACPI), but at the most basic level, the hardware platform and firmware provide a number of **read-only** fields that describe the battery and its status. The firmware provides information on the battery including *manufacturer*, *serial number*, *design capacity* and *last full charge capacity*. The last two pieces of information—design capacity and last full charge capacity—are the information Windows 7 uses to determine how much the battery has naturally degraded. This information is read-only and there is no way for Windows 7 or any other OS to write, set or configure battery status information. In fact all of the battery actions of charging and discharging are completely controlled by the battery hardware. Windows only reports the battery information it reads from the system firmware. Some reports erroneously claimed Windows was modifying this information, which is definitely not possible.

As mentioned, every single indication we have regarding the reports we've seen are simply Windows 7 reporting the state of the battery using this new feature and we're simply seeing batteries that are not performing above the designated threshold. Below we'll talk about the data we have to support this point of view. It should stand to reason that some customers would be surprised to see this warning after upgrading a PC that was previously operating fine. Essentially the battery was degrading but it was not evident to the customer until Windows 7 made this information available. We recognize that this has the appearance of Windows 7 "causing" the change in performance, but in reality all Windows 7 did was report what was already the case.

The following data points contributed to our understanding of the reports we are seeing. Please keep in mind that all the telemetry we see is opt-in, anonymous, and respects our privacy policy.

- We have seen no reproducible reports of this notification on new hardware or newly purchased PCs. While we've seen the reports of new PCs receiving this notification, in all cases we have established that the battery was in a degraded state.
- Our OEM partners have utilized their telemetry (call center, support forums, etc.) and have let us know that they are seeing no activity beyond what they expect. It is worth noting that PC manufacturers work through battery issues with customers and have a clear view of what is to be expected both in general and with respect to specific models, timelines, and batteries.
- We've gone through all the major online support and self-help forums and when appropriate have worked to follow up with any reports of this notification being presented in error. Through this we have identified no reproducible cases where the battery or PC was new and have only learned of batteries that were degraded in capacity.
- In our telemetry from RTM code customers, only a very small percentage of users are receiving the "Consider replacing your battery" notification, and as expected, we are seeing systems older than ~1.5 years. We're seeing relatively fewer notifications compared to pre-release software as the average age of the system decreases.
- Microsoft has received 12 customer service incidents in addition to pulling 8 additional incidents from various forums. To date (for a total of 20 incidents), none of these have shown anything other than degraded batteries.
- Microsoft has been using the technet community moderators to assist in further contacting customers reporting on this notification and we've

assigned additional customer service personnel to be ready. However, of the 30 or so contacts we have received we have not learned of any new facts or conditions with respect to this notice.

- During pre-release testing of Windows 7 we saw almost precisely this same experience with customers in terms of the display of the notification. In fact, in looking at the hardware distribution of pre-release testing we saw an ever so slightly higher number of systems receiving this notice. This follows from the fact that a large set of customers are buying Windows 7 with new PCs or using the upgrade provided with a recent Windows Vista PC.
- When looking at the telemetry reports for the machines that have reported displaying this notification we have seen nothing in additional reliability data that indicates any other system anomalies.
- While the information regarding battery status is provided read-only to the operating system through ACPI, we performed a thorough code-review and verified that there exists no code that is capable of modifying battery status information.

This data would confirm our point of view that we are seeing nothing more than the normal course of battery degradation over time. The transparency provided in this new Windows 7 feature produced a notice that previously was not available to customers and did so shortly after upgrade. This is the root cause of the urgency with which we've seen postings, but does not change the reality of the condition of the battery. We have no confirmed cases of new machines with the as-purchased batteries.

As we always say with regards to any reports on the quality of Windows 7, we are going to continue to be diligent and use all the tools at our disposal to get to the bottom of a report that has the potential to require a code change we would distribute to customers. We are as certain as we can be that we have addressed the root cause and concerns of this report, but we will continue to monitor the situation. In particular, we will continue to have focused communication with our OEM partners as they monitor their customers and PCs over time.

Finally, if you believe you are receiving this error and your battery is new or believed to be in great shape we would encourage you to report this to us or your original PC maker. You are welcome to send me mail through the contact form on this page, use the [TechNet forum](#), the [Microsoft Answers forum](#), or visit [support.microsoft.com](http://support.microsoft.com) where you can get additional information about how to contact Microsoft assisted support in your region.

Thanks,

Steven