CLI326: WinFS - File System Integration and Security

Tim Sneath 29 Oct 2003 3:36 PM

6

WinFS is a marriage with NTFS! It's a file system that co-exists with and leverages the best of NTFS. There are areas where NTFS will not scale well in the future, not because as a file system it is inadequate but because the new requirements people will have in a world of digital data requires a more relational, query-orientated file system. WinFS is the solution to this problem.

Looking back at the evolution of Windows file systems, the jump between FAT and NTFS was huge - everything from file layout to the algorithms for storage were completely revamped. WinFS does not represent such a huge shift, however - it is based on the same paradigms but offers stronger and deeper metadata support and a new querying model.

Key WinFS Concepts

The store is a top-level container for WinFS items. There is more than one store per NTFS volume, and stores for the system volume are created by default. WinFS effectively carves out a chunk of the C: drive (called the DefaultStore at present) as a file; WinFS then exposes as individual files. When you install WinFS, a second store called the CatalogStore is also created - this is a meta-store which contains information about the other stores on your system. You can also create your own stores on-demand against new or existing volumes. Manipulation of stores will be possible through the standard disk tools such as Disk Management, Disk Fragmenter, as well as new tools to do WinFS-specific operations such as consistency checking. WinFS will also integrate into the Plug'n'Play infrastructure, so that as volumes are plugged in, their stores are surfaced in the WinFS namespace. In summary, the ideal situation is that WinFS is transparent to the end-user from an operating perspective.

Shares are a view that are carried forward into the WinFS world. WinFS shares can be created to any folder within WinFS. Again, by default, Longhorn creates a share called **\\localhost\defaultstore**, but you can create your own shares deeper in the store (for example, **\\localhost\sharedpictures**). You can then drill down deeper into those shares, e.g. **\\localhost\defaultstore\contacts\tims**.

What is the default behaviour of WinFS when you install Longhorn? WinFS is not a NTFS replacement - as mentioned already, it is hosted within an NTFS volume. A fresh install will host the Documents, Pictures, Music and Videos folders in the DefaultStore. For upgrades, Longhorn will migrate this same content into the DefaultStore. For non-standard directories, a tool will be provided to migrate this content into the store.

Developing for WinFS

The ItemContext class can be used as a scoping object into the store. You can use a ItemSearcher to query against an ItemContext, optionally using OPath filters. Once you have found a document, you can create a standard StreamWriter object as a streaming interface into the file.

The traditional Win32 API calls are also supported natively in WinFS - for example CreateFile, GetSecurity and so on. The same UNC namespace described above is common to both the managed WinFS API and the Win32 API. One caveat is that non-file backed items are not exposed via Win32.

Every WinFS application starts with an XML schema which defines the types and relationships. Within this schema, a PromotionData section (containing PropertyMapping elements) allows Longhorn to *promote* custom properties from OLE Document, coupled with DocumentMapping elements to define which document extensions are to be included within the schema. This allows applications like Word to be able to build and expose WinFS metadata without having to have any knowledge of Longhorn or WinFS.

Demotion is the opposite of promotion. Demotion is the process of taking metadata from WinFS items and moving it back into files so that legacy applications which only understand files can continue to get at this information. For example, if you edit a Word document custom property via the shell, it automatically gets updated in the underlying file itself as a result of demotion.

Security
A WinFS item is the lowest level of granularity for security control. Every item has a security descriptor, containing a discretionary ACL and a security ACL. As with NTFS, there is an inheritance mechanism that allows objects or containers to base their ACLs on a parent object. Inheritance can be blocked so that it does not propagate through the hierarchy.

WinFS Event Handlers
Every update to WinFS can be processed by an event handler before updating the store. In concept these closely resemble a SQL trigger. You can register your own event handlers (for example, an anti-virus software vendor might scan a file before it is updated). Event handlers can reject, accept, add to or replace an operation, and can accept just a subset of operations rather than having to monitor every item. It wasn't clear from the session how the order of event handlers is managed - will every extension attempt to make itself the last event handler? There are however mechanisms to prevent a virus itself from registering itself as the final event handler.

Comments

Steve Cholerton
29 Oct 2003 3:43 PM

From a standard users perspective I would argue that WinFS is a bigger leap than from FAT32 to NTFS, although technologically FAT to NTFS was a big change, did it *really* change the way the majority of users used the file system? WinFS promises to do just that with the ability to view information in the way that people actually want to work, I think this is a *major* enhancement and I am very excited about it.

Tim Sneath
29 Oct 2003 3:48 PM

You've definitely got a point, Steve. The difference between NTFS and WinFS is huge in terms of the potential it unleashes. I suppose the point the speaker was making was that from a purely technical perspective, this is still NTFS but just with extra extensions rather than a completely new file system. Thanks for distinguishing between these two things! Tim

Andrew Cappon
29 Oct 2003 8:41 PM

Having WinFS “stores” on an NTFS volume could lead to performance issues depending on how they get used in the real world. If it’s true that applications can create and use stores other than the default store it becomes possible for the container files to be fragmented. I haven’t seen the details yet but I assume fragmentation is also a possibility in the internal structure of the store. Will Longhorn include a defragger for NTFS volumes that can move portions of WinFS stores? What about a defragger for inside the store? Sorry if this isn’t the right place for this kind of inquiry or if it was covered in the session, which I regrettably was unable to attend. I’m just an interloper with an interest in high performance file streaming.

Tim Sneath
29 Oct 2003 9:02 PM

Andrew, thanks for the question! Within NTFS, you can consider a store as just another file - in particular, imagine it in the same way as a SQL Server MDB file which contains BLOBs. Keeping stores defragmented is obviously a good thing, and perhaps Longhorn will set aside an area of NTFS for stores to reduce fragmentation. Within a WinFS store, the speaker indicated that there would indeed be defragmentation and consistency checking tools. Hope this helps a little. Tim

Tim Sneath
25 Mar 2004 12:20 PM

dianying xia zai
26 Jul 2004 8:13 AM
dianying xia zai:**http://www.kamun.com/**
movie down:**http://movie.kamun.com/**
mp3 xia zai:**http://music.kamun.com/**
engage:**http://club.kamun.com/**