The Old New Thing

If you cancel an operation while it's in progress, then it's not surprising that it's only half-done

18 Feb 2014 7:00 AM | 65

A customer (via their customer liaison) started by asking why they were seeing an unexpected access control entry in the security descriptor of an object.

> The ACEs on the parent grant access to *Administrators*, *CREATOR OWNER*, *SYSTEM*, and *Users*, but the ACEs on the child object (which should simply have been inherited from the parent) include an extra entry for *Bob*. How did Bob get access to the child object? When we view the details of the ACEs, it lists the Bob entry as *Inherited from parent*. But there is no Bob entry in the parent!

I observed, "Probably because Bob is the *CREATOR OWNER*."

> Thanks for the explanation, but even if Bob is the *CREATOR OWNER*, how can we explain that the permission is inherited from the parent?

The permission is inherited from the parent because the parent has specified the rights of the *CREATOR OWNER*, and Bob is the creator/owner. As part of the inheritance process, the rights of the *CREATOR OWNER* get assigned to Bob.

Remember that *CREATOR OWNER* is not a real person. It is a placeholder that gets replaced with the actual creator/owner when the object is created. If Bob created the child object, then the permissions of *CREATOR OWNER* will be given to Bob on the child object.

The *CREATOR OWNER* is not a live entry that dynamically updates to match the current creator/owner. It is a static entry that is assigned at the point of creation. Changes to the owner in the future have no effect because the identity has already been snapshotted. (I think a less confusing name would have been simply *OBJECT CREATOR*, since creation happens only once.)

(Note that there is a little extra weirdness here: If the creator is a member of the Administrators group, then the *CREATOR OWNER* rights get assigned to the entire Administrators group instead of the specific user who created it. You can change this behavior by tweaking the *Default owner for objects created by members of the Administrators group* policy.)

The customer liaison conferred with the customer, and determined that, at least in one of the cases they were studying, Bob was not the original creator.

What actually happened was that at some point, Bob was granted access to the parent object and all its sub-objects. Later, somebody went back to the parent object and told it to revoke Bob's access to the parent object and all its sub-objects. But "If we cancel the process fast enough, then we get the strange behavior as originally described."

Well, duh!

You asked for Bob's access to the parent object and all its sub-objects to be revoked, so the tool you used started a recursive tree walk from the parent object looking for any objects that Bob has access to and removing them. But if you cancel the operation partway through, then that tool didn't get a chance to finish the job, and you obviously are left in a situation where Bob's access was only partially revoked.

The customer liaison confirmed,

> Yes, that's what happened.

It's nice of the customer liaison to confirm the diagnosis, but it still baffles me that they were confused by this in the first place.

To me this is one of those *So what did you expect* type of situations. You start an operation, then partway through, you cancel it, and then you're surprised that the operation did not run to completion.

Blog - Comment List MSDN TechNet

Comments

Joshua
18 Feb 2014 7:03 AM
#

People often expect one UI operation = one transaction until they have to pay the disk cost.

alegr1
18 Feb 2014 7:23 AM
#

It used to be that the object's owner was getting unlimited access by default, but in Windows 7+ (or in Vista already?) it's possible to limit the creator/owner rights.

Chris Crowther
18 Feb 2014 7:26 AM
#

They may have assumed the operation to be atomic. Although given there's no suggestion that it is, it would be a silly assumption.

alegr1
18 Feb 2014 7:55 AM
#

New line of toys for MS kids: "Bob the CREATOR OWNER"

Cancel
18 Feb 2014 8:00 AM
#

A lot of people assume cancel will undo everything they have done since logging into their machine. I know some software I use can undo to what you were doing last week if you left it running. I also know a lot of software where the cancel button does nothing. You hit cancel, and it hangs the program and the only thing you can do is kill the process. It is all about expectations.

Chris Shouts
18 Feb 2014 8:13 AM
#

It's logical to expect that cancelling the operation will be side-effect free. My general advice to someone assigning permissions would be "Never use the Cancel button. It doesn't do what you think it does."

In this instance I assert that the source of the problem is a defective software design.

dirk gently
18 Feb 2014 8:14 AM
#

Well, in this scenario "Cancel" does not actually cancel the operation, but rather the portion of it that was still not done at the point when the cancel button was clicked.

I can understand that the customer would expect this to be an atomic operation, especially since:

1. it's a security feature we are talking about

2. there is no easy procedure to manually go back to a consistent state.

The MAZZTer
18 Feb 2014 8:23 AM
#

Perhaps a better name would be "Abort" as "Cancel" implies you've changed your mind (and would want it rolled back and undone).

And I think we can at least agree that a halfway done operation is good for noone.

However I expect it is the way it is because a power user will typically hit cancel because they realized they need to make more adjustments, then they will apply the operation again to fix the inconsistent state.

Matt

18 Feb 2014 8:25 AM
#

Maybe they should rename the button from "Cancel" (which implies the action will be reversed) and change it to "stop" which implies only that the action will cease, or even "abort" which implies that the action will cease and there may be unexpected mess that needs to be cleaned up afterwards.

Anon
18 Feb 2014 8:26 AM
#

Agreed with all the "Cancel isn't actually 'Cancelling' anything" posts.

In the case of the security permission assignment, that's an "Abort" button. It bails out of the operation, without any regard for issues that might cause. It has NOTHING to do with "Cancelling" the operation.

Imagine if online ordering worked like this:

"Why was my credit card charged? I never received the product!"

"You cancelled the order, sir."

"But I never got the product!"

"Cancelling in the middle of the order may result in undesired behaviour, including charging your card without sending the product."

"Well, refund the money to my card!"

"Unless you backed up the state of the entire universe prior to your order, we can't do that."

Maurits [MSFT]
18 Feb 2014 8:49 AM
#

Agreed. "Cancel" implies an atomicity that "obviously" does not actually hold for this operation.

Kemp
18 Feb 2014 9:01 AM
#

There seems to be some confusion in the comments. People appear to think that the customer's problem was that the tool didn't undo the changes it had already performed. What I get from the article is that the customer's problem was that the tool didn't finish the job after being cancelled (something that can't be solved by renaming the cancel button).

Joker_vD
18 Feb 2014 9:10 AM
#

Yeah, being able to perform operations of arbitrary complexity atomically in constant time would be extremely handy. Unfortunately, it's not quite possible. Locking has high cost on others; saving rollback information has high cost on you.

Also, Windows security system is kinda complicated... never got down to actually studying it, sadly.

Jim Mischel
18 Feb 2014 9:10 AM
#

I think the "Cancel" comments above are absurd. If you subscribe to a magazine for three years and then cancel your subscription, you don't expect your money to be refunded and all the magazines you received for the last three years to somehow be magically transported back to the publisher and the knowledge you gained from reading them be erased from your memory.

In software, the meaning of "Cancel" is context-dependent. It can either mean, "I've changed my mind, don't do anything." Or it can mean, "Stop what you're doing!" In the latter case there is no expectation that Cancel also implies "Undo".

That said, I agree that "Abort" might be a more reasonable label for that button. But the truth is that in general people shouldn't be fiddling with these security settings if they don't know what they're doing. And they certainly shouldn't be pressing the Cancel button without understanding the ramifications. Perhaps the Cancel button should pop up a confirmation box: "Warning: If you cancel this operation, things will be in an inconsistent state."

There's a continual tension between flexibility and usability. With power comes responsibility. Users are given the ability to make potentially disastrous changes to their systems. As software developers, there are only so many ways we can protect them from their own ignorance or impatience.

Joker_vD
18 Feb 2014 9:18 AM
#

@Jim Mischel: Well, a warning about "things being in an inconsistent state" would be technically incorrect. It depends on what you consider consistent state! If you are changing access rights of several objects and cancel it mid-way, your filesystem doesn't end up damaged. All files are there, and any of them either had its access changed successfuly, or not changed at all. That's pretty consistent.

Billy O'Neal

18 Feb 2014 9:43 AM
#

While I agree that one shouldn't assume that the operation isn't completed completely, the documentation for the INHERITED bit flag seems to indicate that this flag is computed dynamically on the part of the security subsystem; and that if INHERITED it set that one can assume that the ACE was actually inherited. But this isn't the case; and it would be nice for the documentation in question to describe that this is a static, persisted flag, not a calculated value.

Similarly, for CREATOR OWNER, the documentation is misleading because in "gets replaced with the actual creator/owner when the object is created," "the object" is referring to the *children* of the object on which the CREATOR OWNER ACE is being set. Most would expect "the object" to be the object on which the ACE is actually set.

In both cases, this seems like a case of whomever owns the inheritance algorithm describing how the flags work, rather than an API designer describing how they work.

Jan Ringoš
18 Feb 2014 10:07 AM
#

Too bad there is no SetNamedSecurityInfoTransacted.

John Doe
18 Feb 2014 11:03 AM
#

My two cents on this is.

"Cancel" usually means to discard, decline, erase, regret, mostly when not in the middle of work being done. An example is a preferences modal dialog, it usually has a Cancel button that just closes the dialog without applying any preference, and usually they have OK and Apply buttons too.

"Abort" usually means stop no matter what for something already happening. An example is copying or moving files, or error dialogs when performing these operations, which usually also have a Retry button.

By the dictionary, this is my interpretation of how it should be. There's reason to be baffled why Raymond got baffled with the customer's confusion.

I'm another one supporting that the button ought to be "Abort".

metafonzie
18 Feb 2014 11:21 AM
#

When applying ACL changes, there is always a progress bar shown. I think the meaning of a Cancel button, when under some kind of progress bar, is always, "Try and cancel the current operation, but consistency is not guaranteed."

The other 'Cancel' context is obvious because its is usually presented under a question.

If I'm deleting a bunch of files and I cancel out the operation, I don't expect files to be magically undeleted.

Antonio 'Grijan'
18 Feb 2014 11:32 AM
#

Of course, "Cancel" button in progress dialogs always means "Abort": it stops the operation, but does not undo the changes it has already made (which can be impossible, for example, in the case of copy/move with overwrite). The problem here is that the button is too tied to how the computer works, and not to how the user works. Its logic is solid viewer with programmer-colored glasses, but the users don't work that way. As pointed above, the user expects all operations to be atomic, and thus, expects "Cancel" to undo whatever has been done. S/he doesn't understand of times, latencies or overhead by transactions.

Labeling process dialog buttons as "Abort" or "Stop" may help with some users that take some time to try to understand. But most users won't notice the difference. The only sensible things to do, from an usability perspective, is either make a full transactional system (unfeasible because of the resource limitations cited above), or to entirely remove the Cancel button (unfeasible, too, because the user must remain in control at all times). It is one of those situations where you just can't win.

Anon
18 Feb 2014 12:07 PM
#

@Jim

What?

When you cancel a magazine subscription, they continue to send the magazines through the end of your subscription period. They do not simply stop delivering magazines.

Anon
18 Feb 2014 12:15 PM
#

@Antonio

"Cancel" almost NEVER means "Abort," except in Windows Explorer.

Prime Example: In every web browser, that "Cancel" box on your file download progress isn't a "Bail out" button, it stops the download and then removes the partial file.

j b

18 Feb 2014 12:29 PM
#

In the early 90s, a highly touted coding principles was to make any function have 3 phases:

Verification: Check parameters / external conditions, ensure that all preconditions for phase 2/3 are met. Otherwise, error return, with no side effects whatsoever.

Work: Do whatever the function is intended to do. Ontermediate operations are made on local variables; still no side effects are allowed. If work cannot be completed successfully, error return.

Results ("commit", if you prefer): Temporary results in local variables copied to var parameters, pointed-to locations or file. This is the only phase when side effects are allowed. With preconditions checked in phase 1, and heavy work in phase 2 succeeded (otherwise, it would error return), the risk of a phase 3 failis minimal.

This principle obviously is an ideal. Sometimes you have to break the rule. Yet, my colleague teaching intro programming inn the 90s, held it to tje students as a law of nature. That paid back! Her students wrote more diciplined, well structured code than I have ever seen. (Only problem: When later encountered code from other sources (e.g. in my courses), they were bewildered: "it isn't as it should be")

We were helped by the roughly simultaneous appearance of Windows, which promoting similar ideas, in particular for dialog boxes. So, our students learned this as "the modern way of doing high quality code".

Since then, in my programming I have followed those principles as far as possible. When I "must" break the rules, it make me feel somewhat uneasy. Certainly, following the rules doesn't always guarantee that a large and complex operation can be canceled with no side effects, but it sure makes it easier to come closer to the ideal. You COULD insist on structuring code so that every function, at every level, during the commit either modifies the local environment of the caller only, or supply a "commit callback" for the caller to execute in his commit phase (which is another commit callback function). This requires all values to be committed to be heap allocated and every function's commit phase to be split off as a separate function. You may be unwilling to take that effort. (Besides, it might cost several microseconds execution time.) But IF you do, you could get quite close to any operation being fully cancelable.

I would never push these principles if the only benefit was cancelable operations. It turns out that the three-phase model forces (or at least stimulates) you to be far more aware of checking preconditions, which is a good thing. Once preconditions are checked, the work phase code may be simplified - shorter, more readable - being free of checks and verifications; they are already done. And, doing all the commits together may make more efficient use of hardware due to temporal locality.

I am not insisting on everyone following slavishly the three-phase function code model, but I honestly think that keeping it as an ideal will help improving your code quality.

voo
18 Feb 2014 12:41 PM
#

@j b: Your "three-phase function" is basically a part of design by contract. Basically you have preconditions, postconditions and invariants - your approach has the first two covered.

j b
18 Feb 2014 1:28 PM
#

voo,

Sure, invariants is essential for a contract. But lots of contracts are coded NOT following a 3-phase model. Preconditions, postconditions and invariants are essentially _external_ requirements - black-box behaviour. They do not dictate the implemnetation strategy. Consider the 3-phase implementation model a disciplined way to realize the concepts of preconditions, postconditions and invariants.

helmet
18 Feb 2014 2:17 PM
#

I like the idea of distinguishing between a "proper" cancel that leaves you as if you hadn't done the operation at all, and a different name ("abort") for the "just drop everything" behavior that ideally should never happen but in practice sometimes does.

However, please, PLEASE, don't call that "abort". Because that term already has a very specific meaning in the computer science circles I frequent, and that meaning is it leaves you as if you hadn't done the operation at all.

For the "bad" cancel, I suggest that instead we label the button "stop". This still suggests that we leave everything "as it is", but without stepping on the toes of the database terminology.

j b
18 Feb 2014 2:41 PM
#

Reminds me of a discssion I had with my BIL many years ago, about an HP operating system that used the term "abort" to terminate a process, and another one (I think it was a Univac) calling it to "kill" the process, and he insisted, "Well, that's about the same thing, isn't it?"

John
18 Feb 2014 3:07 PM
#

To echo a lot of comments here, people too often think of Cancel as a ROLLBACK of sorts (as if you were in a BEGIN TRANSACTION). I agree with a lot of the comments here that any assumption of that sort without an explicit contract saying so is foolish.

@Windows Explorer ACL Comments

I want to say that Windows 7+ (and maybe even Vista) alert you when you cancel an operation to set security permissions via the explorer dialog. Something to the effect of "your security permissions may now be in an inconsistent state" or some such.

John Doe
18 Feb 2014 3:19 PM
#

@helmet, really, it's the DB terminology that got it wrong, so please, PLEASE, don't impose that use of abort on anything else, and allow the proper meaning to take place where needed.

Please, pick your dictionary of choice and see the definitions of abort and cancel.

helmet
18 Feb 2014 3:43 PM
#

@John Doe - I understand how you feel about "abort", and I agree that it would be great if it weren't for those pesky other people who will understand it "wrong". All I was doing was pointing out that there are people have specific expectations for that word (and they're not reading this forum). Sometimes being "right" doesn't matter when the goal is to come up with a term that everyone understands: if sufficiently many people will misunderstand the word, then perhaps it is worthwhile to look for another term.

You said nothing about my proposal of "stop". Don't you like it?

meh
18 Feb 2014 3:44 PM
#

I agree with metafonzie's file deletion analogy and thus would expect Cancel to do the same sort of thing for other operations in Explorer, though the blog post doesn't actually say what the tool was that was cancelled... maybe they used a command line and pressed Ctrl+C at some point. Anyways, the bafflement probably isn't from any disagreement about what the best name for a button should be or what the ideal design behaviour should be, but instead with the idea that the customer liaison had to ask what the actual cancel behaviour is for the tool and what the end result to the file system objects would be. (Why the question was asked also isn't specified... maybe he/she is a real smart person but there is some sort of process that must be followed.)

Off topic: my favourite percent-done bars are the ones that visually unwind themselves when you cancel - that lets me know that things were intended to be restored to a previous state.

Antonio 'Grijan'
18 Feb 2014 3:51 PM

#

> "Cancel" almost NEVER means "Abort," except in Windows Explorer.

> Prime Example: In every web browser, that "Cancel" box on your file download progress isn't a "Bail out" button, it stops the download and then removes the partial file.

Precisely, in many browser's download dialog, "Cancel" stops the download, but tends to leave the part already downloaded. And in the case of download managers with a list of several files, the already completed files are never deleted. This happens, for example, with the DownThemAll extension on Firefox on Linux - no Microsoft code involved at all. AFAIK, Gnome's file copy dialog works that way, too. And VirtualDub's video conversion dialog. Just to name a few.

What do they all have in common with Windows Explorer? They are examples of progress dialogs, too.

cheong00
18 Feb 2014 5:09 PM
#

Btw, since TxF supports transactional operations, is it possible for the shell to make the process of these kind of recursive update transactional?

If that's what had been implemented, that'd probably match what most user expected.

[*Too bad there is no SetNamedSecurityInfoTransacted. -Jan Ringoš*]

Gibwar
18 Feb 2014 6:15 PM
#

The people complaining about the Cancel button astound me. Off the top of my head, I can't think of any progress bar in Explorer where the Cancel button will completely undo an operation. As another example, pressing Cancel on Windows Update doesn't uninstall the patches it may have already completed. A lot of installers are that way too, if you hit Cancel after the actual installation process has done. They'll leave files behind and not properly roll back a half completed installation.

For those wanting to call it Stop or Abort I think has the same connotations as Cancel, as it's not any clearer what state the system will be left in. The only thing I can think of that can clearly demonstrate the action would be Terminate or, for a more unix-y feel, Kill.

voo
18 Feb 2014 7:00 PM
#

@Gibwar: No idea what you think, but the dictionary (oxford) is pretty unambiguous there:

cancel: decide or announce that (a planned event) will not take place; annul or revoke

abort: bring to a premature end because of a problem or fault

Clearly rather different annotations (heck the dictionary explicitly lists "annul" for cancel and I think we don't have to argue the definition of that one; or is that just because I'm at least officially catholic?) and "abort" fits the description much better - nobody expects something to be rolled back if they "abort" an operation, but there's tons of software out there where "cancel" does indeed roll back changes.

About your astonishment: It's not about whether someone who is used to the idiosyncrasies of a system can understand and use it - by that definition filenames where only the first 8 characters are significant are perfectly fine too, but that doesn't change the fact that they are still a bad user experience. Inconsistencies for no good reason are a bad thing, surprising your users is just as bad.

"Precisely, in many browser's download dialog, "Cancel" stops the download, but tends to leave the part already downloaded"

Huh? You must be talking about Safari, because the other 3 (Chrome, FF, IE) all delete half downloaded files. Even then *one* browser is not "many"..

alegr1
18 Feb 2014 7:11 PM
#

[Too bad there is no SetNamedSecurityInfoTransacted. -Jan Ringoš]

But there is CreateFileTransacted+SetSecurityInfo

Although the TFS is being phased out.

[*SetSecurityInfo is not a transacted operation. -Raymond*]

Drak
18 Feb 2014 10:48 PM
#

Well, if it rolls back changes it should be called 'Roll back', 'Restore' or 'Cancel and roll back'. 'Cancel' in and of itself does not imply a roll back.

For people thinking of real-world 'cancel' situations like magazines: you are canceling something that will take place in the future. You don't actually cancel a subscription, you cancel the renewal of the subscription.

Just like you can't cancel a hotel booking you are in the middle of because the hotel can't retroactively give your room to anyone else, can they?

Just my feelings on this.

cheong00

18 Feb 2014 10:48 PM
#

I read that "Security Change" is also logged by filesystem journal, so I thought there should be corresponding functions.

Maybe it's reserved for future improvements then.

Virtual8086
18 Feb 2014 11:24 PM
#

Going on with the discussion of a better name for the button... I'm not a native English speaker but how do you feel about the word "Break"? Although it seems to be uncommon to use it in such cases, at least it implies that the things being changed will stay broken.

Neil
19 Feb 2014 3:02 AM
#

Well, that's my new piece of information for today; I'd always assumed "CREATOR OWNER" means "apply these permissions to whomsoever is the current owner of the object" rather than "grant these permissions to the creator when creating an object".

Neil
19 Feb 2014 3:05 AM
#

Oh, and of course the next problem is how to remove orphaned permissions...

Gechurch
19 Feb 2014 3:45 AM
#

@The MAZZTer

"However I expect it is the way it is because a power user will typically hit cancel because they realized they need to make more adjustments, then they will apply the operation again to fix the inconsistent state."

+1. I've often done exactly that.

@Kemp

"There seems to be some confusion in the comments. People appear to think that the customer's problem was that the tool didn't undo the changes it had already performed. What I get from the article is that the customer's problem was that the tool didn't finish the job after being cancelled (something that can't be solved by renaming the cancel button)."

Exactly. I'm surprised by how many people had this incorrect reading of the article, given the technical nature of the readership. If the 'Cancel' button had reverted the changes, Bob would still have access to all the files - something the person asking the question didn't want. To get the behaviour the person wanted (Bob having no access), Windows would have had to ignore the 'Cancel' button altogether and continued on removing Bob's access.

@Joker_vD

"@Jim Mischel: Well, a warning about "things being in an inconsistent state" would be technically incorrect. It depends on what you consider consistent state! If you are changing access rights of several objects and cancel it mid-way, your filesystem doesn't end up damaged. All files are there, and any of them either had its access changed successfuly, or not changed at all. That's pretty consistent."

In your mind 'consistent state' (of file permissions) == 'no damage to the filesystem'?? That's insane!

My 2c:

Would 'Abort' be a better button label than 'Cancel'? Yes

If I hit 'Apply' and watched a whole bunch of permission changes go through, then hit the 'Cancel' button, would I be surprised if my permissions ended up inconsistent? Of course not, that would be daft.

GWO
19 Feb 2014 4:30 AM
#

Interesting to see how people's expectations vary with a cancel. I always consider these in the context of Abrahams' exception guarantees for C++; the strong guarantee is basically the transact/rollback semantics that many non-technical users may expect. The basic guarantee is the "Abort" semantics they usually get.

One would certainly like the strong guarantee wherever possible - for external errors more than for manual intervention. If I move a directory to another filesystem, which runs out of space during the process, I really don't want to have to clear up the partial move manually.

Deduplicator
19 Feb 2014 4:42 AM
#

Re Consistent State:

1 Sure, changing the security descriptors and aborting the operation halfway through doesn't leave the filesystem in an inconsistent state in regard to unusable.

2 But anyway, the permissions are in an inconsistent state, which is quite bad enough, thank you very much.

3 All software SHOULD be able to at rationally if 2 prevails, because this situation can arise through different means on NTFS: Junctions anyone? That does not mean any user who has no idea about the underlying system ever consciously encountered and/or expects it.

4 Cancel/Abort: Nobody is surprised by cancel meaning abort, iff they know the system and/or the program putting up that dialog. It's still a bad UI which doesn't speak about the operations performed in the same granularity as the user ordered them, instead of the myriad steps the program uses to implement them. Especially if there is no compelling case for doing so, and nobody to date gave one... Twice so if both cancel and abort are used appropriately elsewhere in the system (meaning all OS+programs).

Engywuck
19 Feb 2014 5:14 AM
#

another surprising detail: when admin A has "change permissions" allowed on a directory but not on a subdirectory he can change the permissions on the parent, but the "inherited" permissions on the subdirectory won't change (with a scary message ond no way to rollback). Was somewhat counter-intuitive when I first stumbled upon it. This can happen by having non-inheriting permissions for the parent ("this folder only").

Lev
19 Feb 2014 5:45 AM
#

I think it's obvious that if an action took time then the "Cancel" button won't magically complete it immediately, and it's almost as obvious that it can't be rolled back immediately, either.

GWO
19 Feb 2014 6:20 AM
#

@Lev: "I think it's obvious that if an action took time then the "Cancel" button won't magically complete it immediately, and it's almost as obvious that it can't be rolled back immediately, either."

That presupposes a level of technical knowledge - and a problem domain. For anyone whose experience is working on production database systems, your second "obvious fact" is both counter-intuitive and wrong. What you think is "obvious" is only obvious through OS/kernel/filesystem-coloured glasses.

Anon
19 Feb 2014 6:59 AM
#

@GWO

Exactly. A massive problem in tech circles is that no one seems to spend more than a half-second thinking about users who were not present during the design process.

John Doe
19 Feb 2014 7:22 AM
#

@Gechurch, the customer didn't really have an expectation, he/she just observed that a child object still had rights for Bob while the parent didn't. If there's an expectation, it's that the recursion applies the operation first to children and then to the parent.

In this case, it correctly processes the parents before the children. It's not really intuitive until you think about it: parents have a broader scope, so let's apply the new permissions (either give or take) far out first.

So, we're discussing the possibility that "Abort" would be a better button label than "Cancel", when the customer was actually finding strange that parent object got updated before child objects. But it's crucial that the user has the notion that "Cancel" does in fact abort the on-going set of operations.

Evan
19 Feb 2014 7:48 AM
#

@Gechurch: "In your mind 'consistent state' (of file permissions) == 'no damage to the filesystem'?? That's insane!"

I mostly disagree. When talking about file systems, "consistent state" to me means that the invariants of the file system are held: e.g. there aren't unused HD blocks that don't appear on the free list, no block is part of two files (for nitpickers: outside of what the FS allows for COW and such yadda yadda yadda), etc. If you stop with a list of ACLs half applied, all of the ACLs are still legal ACLs even if the ACLs for two files disagree. You haven't screwed up any of the internal structure of the file system.

Imagine you have a buggy program that you're using, you hit "save", and it corrupts your save file by writing garbage. Whoops. Well that sucks, but the file system is still in a consistent state.

DWalker
19 Feb 2014 8:34 AM
#

Not that anyone is going to re-word dialog boxes after all this time, but "Stop" is a better term for "don't do anything else and don't undo" than "Cancel". And "Abort" is simply too harsh of a word.

helmet
19 Feb 2014 10:29 AM

#

@DWalker, thanks for agreeing. And @Virtual8086, I like your idea! Since the "cancel" button can (apparently) leave things in an inconsistent state, calling it "break" sounds appropriate! And it's more family-friendly than an other option that starts with "f" and ends with "up".

Kidding aside, it's interesting to see that there are many representatives for both the camp of "the cancel button should leave things as they were before we started" (nice in many ways, though perhaps not practical) and the camp of "cancel just stops and leaves things however they were at the time" (with differing opinions on whether the final state even needs to be consistent).

Deduplicator
19 Feb 2014 10:58 AM
#

Why is 'Abort' too harsh a word? I don't know a better word to state concisely: The operation/configuration is halted, it cannot be resumed, and there probably is detritous in the form of a partially completed operation left, which might require manual cleanup.

Cancel OTOH denotes: The operation/configuration is halted, it may be resumable and the canceled operation does not have any impact.

There are few places both are equally right and appropriate.

Stop now: a) a longrunning operation/subtask is halted b) it can be resumed, though there's probably no way to compensate for the downtime.

Is anyone here phobic to the word "abort"?

Deduplicator
19 Feb 2014 11:10 AM
#

Why is 'Abort' too harsh a word? I don't know a better word to state concisely: The operation/configuration is halted, it cannot be resumed, and there probably is detritous in the form of a partially completed operation left, which might require manual cleanup.

Cancel OTOH denotes: The operation/configuration is halted, it may be resumable and the canceled operation does not have any impact.

There are few places both are equally right and appropriate.

Stop now: a) a longrunning operation/subtask is halted b) it can be resumed, though there's probably no way to compensate for the downtime.

Is anyone here phobic to the word "abort"?

Regarding "Break": Break what? The operation? That's a bit unnatural. Break it in what way? Perhaps break it up so it finishes faster in parallel? Or did you mean taking a break? "Pause" is more appropriate there. Break away? Where to? Break off? Well, you really have to add the "off" then...

As far as i see, the trouble is simply that numerous buttons which should say 'abort' read 'cancel' instead.

The decision not to support rollback under every circumstance in favor of more speed and easier design, esp. for advanced operations, seems to be accepted here.

alegr1
19 Feb 2014 12:18 PM
#

[SetSecurityInfo is not a transacted operation. -Raymond]

It's as much a transacted operation, as SetFileAttributesByHandle, SetEndOfFile, SetFileShortName, etc.

The file security info is part of attributes/metadata stream. Because of that, I expect any changes to it be be isolated as part of the transaction. A change in the security info outside of the transaction should not affect the security info visible inside a transaction, and vice versa.

The dedicated Xx*Transacted functions are only provided for the functions that take a file name as input. Those functions that operate on a file handle (such as ReadFile, WriteFile, etc) do not have special *Transacted form. Since SetSecurityInfo operates on a handle, it doesn't require a special Transacted variant.

[*It's not listed in the table of "functions whose behavior changes if used on a transacted handle." SetNamedSecurityInfo also is not listed as supporting transactions. -Raymond*]

Lev
19 Feb 2014 9:34 PM
#

@GWO

I'd rather say that the ability to quickly undo something that has been slowly done is specific to databases, so most people wouldn't suppose that.

Deduplicator
20 Feb 2014 6:13 AM
#

What about doing your setup using transactional filesystem?

You can bailout without changing anything until you are forced to use the first non-transacted method.

I think the only one you need is CommitTransaction...

And even Explorer has a setup phase for copy/delete/move where nothing is committed to storage yet.

Maurits [MSFT]
20 Feb 2014 6:38 PM
#

> Why is 'Abort' too harsh a word?

It's an emotional trigger for some users, like "illegal operation", "master/slave", or "collaboration."

Gabe
20 Feb 2014 7:56 PM
#

Deduplicator: The setup phase is just collecting information about what files need to be operated on. The problem is that the "committing" phase takes long enough that you can stop it before it finishes. Of course you could do that phase transacted too, but it could have serious repercussions.

There are times when I need to change permissions (say, add a new user) on a whole filesystem. This could take several minutes (or longer). I don't really want the whole filesystem to get locked, file-by-file, until the transaction completes.

GWO
21 Feb 2014 2:22 AM
#

@LEV: "I'd rather say that the ability to quickly undo something that has been slowly done is specific to databases, so most people wouldn't suppose that."

Again, that presupposes so much, its barely worth commenting on.

It's been at least 20 years since people using computers should be expected to have *any* understanding of how a computer works, let alone the distinction between how databases and filesystems typically behave. A naive users expectations are based on what they would like to happen - not what happens to be the most efficient at a filesystem level. They have literally no interest in what those words mean. As soon as you argue from that stand point, > 99% of your users glaze over.

If I cancel a financial transaction it gets undone.

If I cancel a holiday, I may suffer some financial penalty, but everything - flights, hire car, hotels - all gets cancelled.

If I cancel a meeting, I don't want half the people turning up depending on who got invited already.

If I cancel a "Move Folder" I get the contents randomly scattered between source and destination. Ummm, no. To a naive user, that breaks the principle of least surprise. I want my computer to operate on the principles as the rest of my life, not what is most for a computer. That might fly in the 80s, but not anymore.

Gechurch
21 Feb 2014 4:29 AM
#

@John Doe

"@Gechurch, the customer didn't really have an expectation..."

Yes they did. Bob had access to a folder. At some point the customer revoked Bob's access but said "If we cancel the process fast enough, then we get the strange behavior as originally described". Their expectation was that if you cancel an operation before it completes, then that operation would somehow complete anyway.

To those suggesting that when you cancel something that has taken a finite amount of time that you have to be some sort of IT guru to realise that the operation hasn't been rolled back, I say stop being daft. I don't know the first thing about brain surgery. But if a surgeon has been operating on a patient for 30 minutes then stops part-way through the surgery and walks out, I would be happy to bet a large sum of money that the patient isn't left in the same state as they were before the operation began. That's just common sense.

Gechurch
21 Feb 2014 4:34 AM
#

@Evan

"I mostly disagree. When talking about file systems... ...You haven't screwed up any of the internal structure of the file system."

This is a discussion about permissions. And if you cancel part-way through applying a change to those permissions, they'll end up in an inconsistent state. File systems have nothing to do with it.

Gechurch
21 Feb 2014 4:38 AM
#

@Deduplicator

I love that someone with your moniker double-posted!

Gechurch
21 Feb 2014 5:05 AM
#

@GWO

"If I cancel a financial transaction it gets undone.

If I cancel a holiday, I may suffer some financial penalty, but everything - flights, hire car, hotels - all gets cancelled.

If I cancel a meeting, I don't want half the people turning up depending on who got invited already."

You've hit on the crux of the problem - we're overloading the term 'cancel'. There's is a big difference between cancelling something that hasn't happened yet, and cancelling something that is already part-way through.

"If I cancel a "Move Folder" I get the contents randomly scattered between source and destination. Ummm, no. To a naive user, that breaks the principle of least surprise."

I disagree. Replace the word 'Folder' with 'Bricks' and you'll see what I mean.

Deduplicator
21 Feb 2014 8:45 AM
#

The problem is not that there is a difference between canceling something long-running or fast-completed, as that would only effect efficiency of the operation. The problem instead is using "cancel" where "abort" is the proper verb. And there's no excuse for doing so, for two main reasons:

- The described behavior is reasonable and often achievable with little overhead using transactions.

- The actual behavior can be described as succinctly and unambiguously.

@Gechurch: If that surgeon aborts midway through surgery, he'll loose his approbation and be thrown in jail.

If he cancels surgery instead, depending on circumstances it either gets rescheduled, someone else gets tasked to do it at the original time and location or maybe everything gets replanned.

See the difference between abort and cancel?

BTW: I don't object to current behaviour of explorer, i only object to them using the wrong wording for their dialog and my options, promising goods they do not deliver (but which _might_ be reasonably possible most times using e.g. transactional NTFS).

Regarding 'Principle of least surprise' GWO mentioned: That's certainly needlessly violated there, as ui and mechanics are both reasonable, but do not correspond. Either would make a good choice, maybe with fallover from transacted to non-transacted if cancelling gets expensive/impossible. Maybe even offer both cancel for stop+rollback and abort for stop+partial_commit where it makes sense for ubergeeks...