# Buck Hodges

VSTS - Team Foundation Server - Team Build - Web Access - Version Control - Admin & Ops

## Keyword expansion in TFS

Periodically, the topic of keyword expansion comes up, which TFS (at least through 2008) does not support. At one point during the v1 product cycle, it was a planned feature and was partially implemented. However, there are lots of challenges to getting it right in TFS version control, and it wasn't worth the cost to finish the feature. As a result, we ripped it out, and TFS does not support keyword expansion.

Since it's not supported in the product and not likely to be supported any time soon, folks gravitate toward the idea of using checkin policies to implement keyword expansion. The idea is appealing since the checkin policy will be called prior to checkin, of course, which would seem to provide the perfect opportunity to do keyword expansion.

Personally, I’m not fond of trying to do keyword expansion as a checkin policy. There are a number of issues related to checkin policies to deal with immediately, because any checkin policy that performs keyword expansion is going to modify file contents.

- Checkin policies get called repeatedly. Every time the user clicks on the checkin policy channel in the pending changes tool window in Visual Studio, for instance, the checkin policies are evaluated.
- Whatever a policy does must be done really quickly. Otherwise, you are going to make the VS painful to use. The checkin policy evaluation isn't done on a background thread, and it wouldn't really help anyway since you wouldn't want to have to wait for some long policy evaluation before the checkin process started.
- Checkin policies can be evaluated at any time. The user may or may not actually be checking in at the point that the checkin policies are evaluated. You even have the option of evaluating checkin policies prior to shelving.
- For applications using the version control API, checkin policies are only evaluated if the application chooses to evaluate them (see How to validate check-in policies, evaluate check-in notes, and check for conflicts). Some folks may read this and this it's a hole in checkin policy enforcement. However, since checkin policy evaluation is done on the client, you can't rely on it being done (i.e., clients can lie and call the web service directly anyway). The other reason has to do with performance. For an application like Visual Studio, it controls when checkin policies are evaluated, and by the time that it calls the version control API to checkin, there's no need to evaluate them yet again. Some day there may be server-side checkins, but they don't exist yet (as of TFS 2008).
- You've got to get your checkin policy onto all of the client computers that are used to check in. The deployment story for checkin policies is probably the single biggest hole in the checkin policy feature in the product (the second biggest hole is the lack of built-in support for scoping the policy to something less than an entire team project, though there is a power tool checkin wrapper policy to do that now). Any computer without the checkin policy assembly on it and properly listed in the registry is not going to do keyword expansion.

If you read that and still want to do it, you would need to pend an edit on each file that does not already have the edit bit set (for example, non-edit branch, merge, rename, and undelete) and is not a delete (can’t edit a pending delete). I’m pretty sure that VS and the command line will have problems with changes being pended during a checkin policy evaluation, because they’ve already queried for the pending changes and won’t re-query after the pending checkin policy evaluation.  This would result in edits not being uploaded.  This pretty much makes pending edits on files via the checkin policy impractical.

Alternatively, you could do keyword expansion only for changes where the edit bit is already set in the pending change.  That's sort of the "least evil solution."  You would just use the checkin policy for keyword expansion in files that already have pending edits (i.e., check to see that the Edit bit is set in the pending change's ChangeType).

Some of the files with pending edits may not have actually been changed (e.g., you pended edits on all of the files in a directory as a convenience because you knew you would be editing at least half of them via a script).  When the server detects that a file that's being checked in with only a pending edit hasn't been modified, it doesn't commit a change for that file (i.e., create a new version of that file).  You can read a bit about that in the post, VC API: CheckIn() may return 0.  To detect for yourself whether this is the case, you can compute the MD5 hash of the file content and compare that to the HashValue property of the PendingChange class.  If the two are equal, then the file didn't change.  For those of you doing government work, you'll want to watch out for FIPS enforcement.  When that's turned on in Windows, MD5 hashes are unavailable because the MD5CryptoServiceProvider class in .NET throws when you try to create one.  In that environment, the hash values are empty arrays.

But wait, there's more!  You would also have to make sure that you read and write the file in the correct encoding (e.g., reading in DBCS as ASCII or Unicode would be bad – for example, Japanese or Chinese DBCS files).  There are probably more encoding issues to contend with.  One thing that's probably on your side, though, is that if you do read in the file in the wrong encoding, you won't likely find the markers indicating that the file needs keyword expansion.  To avoid randomly finding the tags when you know you don't want to, you'd likely want to skip all binary files that your expansion logic doesn't know how to handle (e.g., you could conceivably handle keyword expansion in JPEG file headers, but that doesn't seem too likely).

The other thing to consider is how keyword expansion interacts with branching and merging.  Imagine putting the date in every file in a keyword expansion system.  It's going to be a merge conflict every time your merge branches.  The same is true for log (history) information.  You would need to write a tool to handle the content conflicts; otherwise, merging large branches would be a real bear.

Even with all of that, you are not going to get one of the things that often comes up (and this was true when we were thinking of putting in the product, because it was all going to be done on the client prior to checking in) which is the ability to record the changeset number in the keyword expansion comment.

So, to sum it all up, you are going to need to consider the following.

1. Your checkin policy will get evaluated multiple times and often not when someone is actually checking in.
2. You'll want to only modify the files that already have pending edits, as pending new changes from within a checkin policy may lead to new content changes being uploaded with the rest of the checkin.
3. You'll want to test out how it's going to interact when merging files from one branch to another.  What's it like to deal with the conflicts your keyword expansion introduces?
4. There will be checkins where keyword doesn't happen for whatever reason, so it's not completely reliable.  This is in addition to the fact that changes that do not already involve an edit won't have keyword expansion at all.

If we had done keyword expansion as a feature, what would have been different (other than the fact that you wouldn't have to think about all of this :-) )? We wouldn't have the limitation of 1. Just like in 2, we'd have to think hard about whether every rename, branch, undelete, and merge should have an edit pended also. At best it would have been an option, and it wouldn't have been the default. Regarding the branch merging issue, doing something on the server would be a performance hit that may be excessive with large branches (keep in mind that the server stores the content as compressed and doesn't uncompress it -- the client takes care of compressing and decompressing content), so the client would need to have some logic to help make it bearable (e.g., before performing a three-way merge, collapse all of the expanded keywords).

Our conclusion was that the feature was too expensive to implement and test relative to the value provided and other features could be implemented and tested with a similar effort (e.g., making history better). Folks either strongly agree (can't live without it) or disagree (don't care about it at all) with that conclusion. There's rarely anyone on the fence. The feedback we've received to this point indicates that we've made the right tradeoff for the vast majority of our customers (and potential customers).

Technorati tags: tfs, team foundation, version control, keyword expansion, checkin policies
Published Saturday, July 07, 2007 1:56 PM by buckh
Filed under: Source Control, Team Foundation

### Comment Notification

If you would like to receive an email when updates are made to this post, please register here

Subscribe to this post's comments using RSS

## Comments

### # re: Keyword expansion in TFS

that illustrates one of the problems in the reliance upon code-as-text, doesn't it? because you can imagine that keywords would just be a mix of mandatory (username, date, etc.) and arbitrary (label, free text, sprint number) attached to a changeset, which the client could just substitute when viewing the code. but at the same time, having some alternative visualization of arbitrary meta-data attached to the code (for instance, as additional columns in the repository view, like Vista's explorer windows) would help to eliminate the need for keyword expansion... :)

Saturday, July 07, 2007 5:23 PM by JohnS

### # re: Keyword expansion in TFS

I agree that there are lots of possibilities for associating and displaying metadata related to files, changesets, and more.

However, the folks that want keyword expansion invariably want it because they want to have that information in the files themselves. That way no matter what editor or environement is being used, the information is available.

Buck

Saturday, July 07, 2007 8:41 PM by buckh

### # re: Keyword expansion in TFS

If you are a making case for shipping product to meet deadlines I understand. Though it sounds like you have gone to the next step and decided to remove keyword expansion from source code control architecture.

I like to use keyword expnsion ( mainly $Revision$) for programs that are not compiled (T-SQL, vbs, cmd files, etc..) The history comments we can type in ourselves and do not need keyword expansion. The $Revision$ information ensures some level of confidence when debugging problems.

I do not agree with the vendor trend of using diff tools to act as my source code control. I think it compliments the process.

**Tuesday, August 07, 2007 10:38 AM by TimAgain**

# re: Keyword expansion in TFS

Tim, I did blur the two reasons a bit.  With v1, it was cut for time.  For v2 and beyond, we haven't come to the conclusion that we can have the features that folks using keyword expansion want and weighed against the time to implement and test the feature.  In your case, you want a subset of the traditional keyword expansion features.

Partly I wanted to see the reaction that folks would have to guage interest in the feature, and I appreciate your comments.

Buck

**Tuesday, August 07, 2007 11:11 AM by buckh**

# re: Keyword expansion in TFS

Keyword Expansion is one of those features that doesn't seem valuable on paper.  However, it serves as one more check and balance point.  When you see the version number in the file, it may be obvious that you're working on the wrong source file.

Also, as part of our standard header, we assign the source file name and version number to a private constant.  I can then search for the file name in the DLL or EXE and know exactly which version of the source file that item was built with.  That can save a lot of debug time later.  Often times, I can see exactly why a bug was not fixed or has reappeared.  Search the DLL and sure enough, it was built with an old version of the source file!

**Monday, September 10, 2007 2:13 PM by Matt Janofsky**

# re: Keyword expansion in TFS

I used keyword expansion (mainly $history) in several projects and made it a standard part of our dev process. It's very convenient to look at source code and determine what changes have been made regardless of environment or available tools. If these comments are just in source control, it's one more (inconvenient) step.

I recently migrated from VSS to TFS Source Control and could not believe that it did not support this function. Now something that has been part of our standard process for years has to be modifed for a yet unknown alternative.

**Tuesday, September 11, 2007 2:12 PM by ArchVile**

# re: Keyword expansion in TFS

I also find keyword expansion important for non-compiled text files (sql scripts, config files, etc.)

When you are looking at copies of these files that have been deployed, printed, emailed, or burned on a CD you can tell which revision in source control it was.

I don't understand why it's necessary for the source control system to store the *values* of the keywords to be expanded.  Strip out the existing value of the keyword when checking in and insert the current value when checking out.  Wouldn't that solve the merging problems?  The keyword would always be something like "$MyKeyword:$" stored in the source system and "$MyKeyword: 1234$" when expanded.

**Thursday, September 27, 2007 10:07 AM by thensley**
**# re: Keyword expansion in TFS**

I agree with that last point by thensley.

I am developing some javascript files that will e used by a 3rd party and I need to send periodic updates to a bunch of people that don't have access to TFS. It would be nice to automatically include a revision number so that people can check they have the correct version in the correct environment.

**Monday, July 28, 2008 12:00 PM by Alastair Green**
**# re: Keyword expansion in TFS**

Used to compare deployed versions of stored procedures on the DB server. Also used for logging information by setting up a private constant in the source file and inserted into the error handling.

**Monday, August 18, 2008 1:26 PM by Jeff Clark**
**# re: Keyword expansion in TFS**

Keyword on $Revision$ is incredibly handy. SharePoint development can require a lot of non-compiled files to effectively customize a system. The ability to quick reference a version number on a deployed file is extremely handy.

**Tuesday, September 02, 2008 9:50 PM by Maurice Prather**
**# re: Keyword expansion in TFS**

This just boggles my mind. How can you not have the ability to embed a version string into a binary file? I personally know of several instances where that information saved us countless hours of tracking down a bug that had already been fixed, simply because we could instantly determine that our customer had the wrong version of our code.

**Monday, September 08, 2008 7:54 PM by Jeff Chadwell**
**# re: Keyword expansion in TFS**

I think this is really an unfortunate omission. I would not consider moving over unless the keyword expansion exists, for the reasons others have already provided:

1. When code is compiled into a database, it is nice to see which version it is, and automatic keyword expansion streamlines this process greatly.

2. Injection of version info into error messages in the code - again this streamlines this too.

The keyword expansion is, in our case, never used to determine the version of a file that is under source control, it is used for determining versions of files that have left source control and been distributed.

So...we will continue to host our repository within a system that can handle this quite simple feature.

**Tuesday, November 11, 2008 11:20 AM by Martin**
**# re: Keyword expansion in TFS**

Quick point, and maybe I'm being blinkered, but why does it have to be expanded on checkin?

Surely expansion on checkout is pretty easy...

**Tuesday, November 11, 2008 11:22 AM by Martin**
**# re: Keyword expansion in TFS**

...and reduce the keyword on checkin to prevent conflicts?

(sorry for the multiple messages)

**Tuesday, November 11, 2008 11:24 AM by Martin**
# re: Keyword expansion in TFS

Keyword expansion is very important for development. I have never used source control for a SW product without it, wouldn't consider a version control product real if its missing, and is a required feature for us.

So while TFS has many things that make it superior to VSS (which BTW stinks for large products), it will not be used by any of the product teams I work with until it matures and gets this feature.

While I feel Buck's pain at implementing it, its just too important a source of information for broad based teams and external partners who get source files but won't have access to TFS.

**Wednesday, November 26, 2008 3:18 PM by padamson**
# re: Keyword expansion in TFS

Can't agree more that this is a crucial feature and seriously lessens the usefulness of TFS. And frankly, with all of the logic involved with merging, branching, and check-in policies, to say that adding keywords is too hard sounds like BS. Sorry guys, if VSS could do it...

It's also a little too integrated with Visual Studio, to the point where it's almost useless without it. It's also very poorly integrated with SQL Management Studio. It doesn't seem to work unless I have a project, which I really don't need in order to edit a single stored procedure or view. VSDB edition isn't much better.

I don't know if this is true but I've heard that MS doesn't use TFS for their own source control. Even if this is a rumour, what does that say about it?

**Friday, March 27, 2009 11:16 AM by rvolk**
# re: Keyword expansion in TFS

Keyword expansion remains on our feature backlog, but I don't know when it will make it into the product. It is not in 2010.

We've been using TFS internally for years, and the entire Developer Division has been using it since early last year. Brian periodically posts an update on it: http://blogs.msdn.com/bharry/archive/2009/01/16/microsoft-tfs-adoption-update.aspx. You can see the full breakdown of where it is used. It is true that Windows and Office do not yet use it for version control, but DevDiv, MSIT, and other parts of the company do.

Buck

**Friday, March 27, 2009 12:07 PM by buckh**
# re: Keyword expansion in TFS

At the very least, $Revision$ should be supported. It's practically universal.

I'm saying this after 20+ years of using many source code tracking systems (SCCS, RCS, CVS, PVCS, ClearCase, PerForce, Subversion, VSS, etc.), and more to the point, having to convert source files from one system to another.

I've heard the counter-arguments for not using keywords and why they are bad and all. All of these arguments are quite simply wrong.

**Wednesday, April 01, 2009 3:36 PM by Loadmaster**
# re: Keyword expansion in TFS

This feature is very important to trouble shooting field installations that go bad. I can not believe this feature does not work in TFS. At the very least I would like to see

**$Workfile$**

**$Modtime$**

**$Author$**

**$Revision$, how hard could that be.  Every other revision control software has it.**

**Thursday, April 23, 2009 4:15 PM by rskedel**
**# re: Keyword expansion in TFS**

**+1.**

**If it's too difficult to support complex user-defined keyword substitution, fine, omit that. But don't use that as an excuse to avoid implementing the basic predefined keywords that experienced developers expect to have available, like $Id$, $Revision$, $Author$, $Modtime$ and so forth.**

**Thursday, June 11, 2009 3:08 PM by John Hardin**
**# re: Keyword expansion in TFS**

**I think it should be included.  If it isn't, supply a good alternative that works with text files cross platform, regardless of editor.**

**Monday, June 15, 2009 11:42 AM by adam davis**
**# re: Keyword expansion in TFS**

**I am coming in new to TFS - or was.  Staggered it doesn't support keyword expansion, won't be using it now.**

**Thursday, June 25, 2009 5:31 AM by D Bullen**
**# re: Keyword expansion in TFS**

**ARGH! Started a new job, was excited to finally have a chance to try out TFS "for real." Excitement turned to disbelief mixed with bitter disappointment at finding out that keyword expansion is nonexistent. Srsly?**

**Friday, July 24, 2009 4:52 PM by David A. P.**
**# re: Keyword expansion in TFS**

**Hi!**

**I implemented an Check-In Plicy which does keyword explansion...**

**http://blog.kalmbach-software.de/2009/07/24/tfs-automatically-insert-check-in-comments-into-source-code/**

**It currently only supports $log$, $date$, $author$, $typ$.**

**$revision$ is hard to implement, because the number is only available *after* the check-in is done ;( you can only use "previousrevision" ;=**

**But the main problem in bigger teams is the very bad concept of check-in policies... there is no global storage of the settings; also you do have to deploy it on every dev-PC...**

**Are there any news in TFS2010 regarding Check-In Polcies?**

**Saturday, July 25, 2009 8:46 AM by Jochen Kalmbach**
**# re: Keyword expansion in TFS**

**Jochen, nice work!  I wish I had better news for you on deploying checkin policies in 2010, but unfortunately it is still not automatic (it'll get strong consideration for the next release after 2010, I believe).  There is a little-known feature in the 2008 power tools that will do it, but you then have to make sure all of the clients have the power tools**

**installed. If that's universal, it may be an option for you.**

**Buck**

**Monday, July 27, 2009 6:15 PM by buckh**

**# re: Keyword expansion in TFS**

**Buck, I agree with the previous posters. We just moved to TFS and I also am shocked to find it doesn't do $revision$ expansion. I'm the main database developer and most of the code I write is in scripts (text files). I embed the revision into all the stored objects (packages, procedures, triggers, views, etc) so we know exactly which version a client has installed in their database. PLEASE push for this feature in future releases. I'd drop TFS in a second and go back to VSS if it were up to me.**

**Also, do you have a link to the 2008 power tools that you mentioned?**

**Wednesday, July 29, 2009 10:34 AM by Don McLeish**

**# re: Keyword expansion in TFS**

**Don, I've been reading all of the comments, and I've also forwarded them to the version control PM. It's definitely on the backlog for consideration in the next release. It seems to be a very binary issue where folks either must have it or don't use the feature at all.**

**Here's a link to the power tools: http://msdn.microsoft.com/en-us/teamsystem/bb980963.aspx.**

**Buck**

**Wednesday, July 29, 2009 5:26 PM by buckh**

---