

# When does TC decide to create a clean...

---



[Simone Busoli](#) 201 posts since

Jun 20, 2005

Using version 5.0.2 with TFS server side checkout.

I notice random clean checkouts with message: "Will perform clean checkout. Reason: Checkout directory is empty or doesn't exist", which should not be true since the agent has built the project in the past.

How do I diagnose the reason for clean checkouts?

Tags: tfs, checkout



[Marina Grechko](#) 600 posts since

Oct 12, 2009 **1. Re: When does TC decide to create a clean patch?** Mar 16, 2010 12:18 PM

Simone,

Please refer to the following section in our doc for the automatic clean checkout reasons: <http://confluence.jetbrains.net/display/TCD5/Clean+Checkout#CleanCheckout-AutomaticCleanCheckout>

There are cases when agent can delete the build checkout directory when it expires. It means that checkout directories are automatically deleted from disk if not used (no builds use it as checkout directory) for a specified period of time (8 days by default). Does this reason match your case?



[Simone Busoli](#) 201 posts since

Jun 20, 2005 **2. Re: When does TC decide to create a clean patch?** Mar 17, 2010 12:16 AM

Hi Marina, it helps, thanks. I wasn't aware of all the reasons why TC performs a clean checkout. However I would like to be able to diagnose that, I'm not sure but I think I've seen more clean checkouts than I expected to see. Does the agent log the reason with some explanation anywhere?

When does TC decide to create a clean...

There's one point I have not clear yet and that's how the checkout concept in TeamCity relates to "getting the sources from source control". If you're using agent-side checkout they probably have a 1-1 relation, but in case of server-side checkout I think it's a bit less clear what checkout means, in that the server caches the sources from the VCS and sends it to the agents.

So I have a couple of questions about that:

1. I'm using server-side checkout because I think it reduces the access to the VCS which in my case represents a bottleneck. Is that really true or does TC server cache the sources even when using agent-side checkout? To be clearer, if I have 2 agents and 1 build configuration and I run the build on both agents, how many times are the sources downloaded from the VCS in the two cases (agent/server checkout)? I'd say one for server-side and 2 for agent-side, can you confirm this is the behavior?
2. In case of server-side checkout are the VCS sources cached per build configuration or if several build configurations share the same VCS root then they are cached one time only?
3. Related to the question before, if using server-side checkout and I have several build configurations sharing the same VCS root but with different checkout rules, how many times are the sources cached? Do different checkout rules influence this behavior?
4. When using a same non-default checkout directory for several build configurations it's not clear to me if the consideration below holds: The previous build in this directory was of a build configuration with different VCS settings (can only occur if the same checkout directory is specified for several build configurations with individual VCS settings and VCS Roots)
5. When using server-side checkout and TC decides to perform a clean checkout, is this decision influenced in any way by the agent on which the build configuration is going to run? In other words, the clean checkout is performed based on the sources the server has in that moment of time or based on the copy of the sources the agent has? I noticed that at times the server simply cleans the agent checkout directory and sends it all the cached sources (which then doesn't impact on the VCS), while in other cases the sources are downloaded from scratch from the VCS. What discriminates this behavior?

Thanks a lot. Cheers.



[Yegor Yarko](#) 1,459 posts since

May 5, 2004 3. **Re: When does TC decide to create a clean patch?** Mar 18, 2010 5:15 PM

Simone,

Generally, I do not think you should go into such detail and try to workaround issues with your version control using TeamCity. Moreover, the concrete behavior depends on the version control you use.

When does TC decide to create a clean...

In general, in case of server-side checkout, TeamCity caches the sources on the server per VCS root checkout settings and goes to the version control server for the clean patch only once per day with all the rest builds only retrieving incremental patch from the VCS server.

However, sometimes building incremental patch can be not quicker then building a full one. e.g. AFAIK for TFS there is no full patch cached by TeamCity.

When using a same non-default checkout directory for several build configurations it's not clear to me if the consideration below holds:

The previous build in this directory was of a build configuration with different VCS settings (can only occur if the same checkout directory is specified for several build configurations with individual VCS settings and VCS Roots)

Do you really *need* to use non-default checkout directory? This note relates to the case:

- a build A with certain VCS settings is build in the directory X
- when a build B with *different* VCS settings is going to build in the *same* directory X, TeamCity first deleted the content of the directory and performs clean patch.

When using server-side checkout and TC decides to perform a clean checkout, is this decision influenced in any way by the agent on which the build configuration is going to run? In other words, the clean checkout is performed based on the sources the server has in that moment of time or based on the copy of the sources the agent has?

The version and the VCS settings of the directory on agent is maintained on the server side, thus related clean sources decision is taken on the server. However, agent can also request a clean patch when it discovers inappropriate state of the directory (e.g. empty checkout directory).

When does TC decide to create a clean...

I noticed that at times the server simply cleans the agent checkout directory and sends it all the cached sources (which then doesn't impact on the VCS), while in other cases the sources are downloaded from scratch from the VCS. What discriminates this behavior?

The full sources patch is downloaded from the VCS at least once a day (the first build after server cleanup). And also after manual "clean patch" action. Also, incremental patch since this daily full patch is downloaded for each build.



[Simone Busoli](#) 201 posts since

Jun 20, 2005 4. **Re: When does TC decide to create a clean patch?** Mar 21, 2010 2:28 PM

Generally, I do not think you should go into such detail and try to workaround issues with your version control using TeamCity. Moreover, the concrete behavior depends on the version control you use.

I agree that ideally I should not care what TC does with the VCS, but at the moment we are experiencing major issues with the load TC is exercising on our VCS, and we have to find a way to deal with it. Our code base is pretty big, and checking it out completely might take tens of minutes, which is something we cannot deal with. So any advice on this side is welcome.

In general, in case of server-side checkout, TeamCity caches the sources on the server per VCS root checkout settings and goes to the version control server for the clean patch only once per day with all the rest builds only retrieving incremental patch from the VCS server.

So, as far as I understand, if I have several build configurations sharing the same VCS root but each one with different checkout rules (which we set up with the aim of improving checkout performance), do they share a common clean patch or does each one of them require a standalone patch? In the second case I guess our decision to fine-tune the checkout rules per build configuration is counter productive then, and we better remove them completely, can you confirm? In this case, however, we need to configure build triggering rules, since we don't want to see a build starting because something has been checked in which doesn't have anything to do with this build configuration.

However, sometimes building incremental patch can be not quicker then building a full one. e.g. AFAIK for TFS there is no full patch cached by TeamCity.

Can you explain this?

Do you really *need* to use non-default checkout directory? This note relates to the case:

When does TC decide to create a clean...

I do. Please take into account that certain codebases might be very complex, and while removing the need for a special checkout directory might be very simple in most cases because of well factored code which doesn't need to know where it's running, in other cases, when you have to maintain large chunks of legacy code, this might just not be possible. This is our case, where the code - test code - needs to be running in a specific location on the file system and considering to make it independent of this constraint requires a lot of work.

- a build A with certain VCS settings is build in the directory X

- when a build B with *different* VCS settings is going to build in the *same* directory X, TeamCity first deleted the content of the directory and performs clean patch.

BTW, luckily we are in a situation in which different build configurations use the same checkout directory but with same VCS settings.



[Yegor Yarko](#) 1,459 posts since

May 5, 2004 5. **Re: When does TC decide to create a clean patch?** Mar 24, 2010 12:59 PM

I agree that ideally I should not care what TC does with the VCS, but at the moment we are experiencing major issues with the load TC is exercising on our VCS, and we have to find a way to deal with it.

There are at least two activities that result in TFS server communications: checking for changes and sources checkout. If you experience high load on TFS, it's worth figuring out what is producing the load. Checking for changes happens in background on and on the build start. The frequency of the background operation can be managed with "Checking for changes interval" in a VCS root. Also, teamcity-vcs.log can provide additional details on the operations and how long do they take.

Our code base is pretty big, and checking it out completely might take tens of minutes, which is something we cannot deal with. So any advice on this side is welcome.

I'd advice to try using agent-side checkout: this way the sources transfers are handled by TFS and TeamCity does not introduce any additional steps. To eliminate clean checkout you also should ensure the same directory is used only by build configurations with exactly the same VCS settings (including same VCS roots and checkout rules).

So, as far as I understand, if I have several build configurations sharing the same VCS root but each one with different checkout rules (which we set up with the aim of improving checkout performance), do they share a common clean patch or does each one of them require a standalone patch?

When does TC decide to create a clean...

Currently, a cached patch is stored for VCS root + checkout rules combination, so different checkout rules mean different caches. Actually, this behavior is implementation-specific and may change, so I'd not try to rely on it too much. The main point is that clean patches cache is not designed to cope with VCS loading issues, but to try to speedup things in TeamCity a bit.

In the second case I guess our decision to fine-tune the checkout rules per build configuration is counter productive then, and we better remove them completely, can you confirm? In this case, however, we need to configure build triggering rules, since we don't want to see a build starting because something has been checked in which doesn't have anything to do with this build configuration.

Checkout rules affect both the changes displaying in a build configuration and sources checked out on agent for the build, so general advice is to limit the sources only to those actually affecting the build: this way you will at least save human resources investigating changes/builds correspondence. At the same time using the same VCS settings over more build configurations will probably improve sources checkout performance. But that depends on a number of factors and should be investigated in each case separately. Probably, you can experiment and find the best approach for you.

However, sometimes building incremental patch can be not quicker than building a full one. e.g. AFAIK for TFS there is no full patch cached by TeamCity.

Can you explain this?

Sorry, I was wrong on the TFS caching, TeamCity does cache TFS full patches in case of server-side checkout.

As to incremental vs clean patch - depending on the nature of the changes incremental can be quicker or not. If changes usually affect not many files, then incremental patch is usually quicker.

I do understand the issue with fixed checkout directory. BTW, since TeamCity 5.0 we try to detect same checkout directory used with different VCS settings - please pay attention to warnings near checkout directory setting.



[Simone Busoli](#) 201 posts since

Jun 20, 2005 6. Re: **When does TC decide to create a clean patch?** Mar 24, 2010 10:45 PM

There are at least two activities that result in TFS server communications: checking for changes and sources checkout. If you experience high load on TFS, it's worth figuring out what is producing the load. Checking for changes happens in background on and on the build start. The frequency of the background operation can be managed with "Checking for

When does TC decide to create a clean...

changes interval" in a VCS root. Also, teamcity-vcs.log can provide additional details on the operations and how long do they take.

Our issue is primarily with downloading the sources from the server due to a big amount of code.

I'd advice to try using agent-side checkout: this way the sources transfers are handled by TFS and TeamCity does not introduce any additional steps. To eliminate clean checkout you also should ensure the same directory is used only by build configuraitons with exactly the same VCS settings (including same VCS roots and checkout rules).

I'd advice to try using agent-side checkout: this way the sources transfers are handled by TFS and TeamCity does not introduce any additional steps. To eliminate clean checkout you also should ensure the same directory is used only by build configuraitons with exactly the same VCS settings (including same VCS roots and checkout rules).

Ok but I fear things might get even worse. Since now the load on the VCS is already high, moving to agent-side checkout means multiplying the current load for the number of agents-1 since no cache is kept on the server. Am I missing something? This is improved only by the fact that now I'm using custom checkout rules so there are actually as many cached sources as the number of VCS root/checkout rules combination which would become 1 in case of agent checkout. Is this correct?