
AMD DEVELOPER INSIDE TRACK

AMD CODEANALYST SUPPORTS MICROSOFT® VISUAL STUDIO® 2010



This video features Frank Swehosky, a 10 year AMD CodeAnalyst team veteran, giving an overview of how to integrate AMD CodeAnalyst Performance Analyzer into Microsoft Visual Studio 2010. He also gives some insight into how you can analyze C/C++ native code as well as .NET managed code.

TRANSCRIPTS

Frank Swehosky, Sr. Software Development Engineer on the AMD CodeAnalyst Team

Sharon: AMD Developer Inside Track is here with Frank Swehosky from the CodeAnalyst Team. Frank, why don't you introduce yourself?

Frank: Hi, I'm Frank Swehosky. I joined AMD right out of college in the year 2000. So, I've been here 10 years, been working on the CodeAnalyst project the entire time along with a couple of other projects they tossed my way.

Sharon: Great so you are very well qualified to tell us about CodeAnalyst. And what I wanted to cover today was, we recently added Visual Studio 2010 support. Can you walk us through the integration process of how CodeAnalyst integrates into Visual Studio?

Frank: Sure. Visual Studio makes it pretty easy for various people to make add on's and integration packages. They have a whole SDK that we use. There are classes that we have to subclass and implement various methods and then it just plugs right in. It's great! A little tricky debugging it sometimes. You have to have another browser, or IDE nested within it to debug it, but it works pretty nicely.

Sharon: Great so from a user perspective they turn it on and...

Frank: Right from a user perspective, they've installed Visual Studio 2010. They've maybe done a couple of projects. They install CodeAnalyst and it asks them up front, do you want to integrate with Visual Studio during install? All they have to do is hit "ok". Then, the next time they open Visual Studio there will be an ability to open a CodeAnalyst project for each Visual Studio project they have. There's a pane on the tool bar that comes up, same tool window that the solution explorer is in where they can say, "Hey, create a CodeAnalyst project for this." Or they can use the convenient toolbar and just click that button to get CodeAnalyst going. Once a CodeAnalyst project is created for a Visual Studio project it will automatically pull the current startup project settings. So, when you click debug, Visual Studio it runs an application, if you are working on a DLL it will launch the executable that uses that DLL and CodeAnalyst will pull that information out of the project and use it. So, you don't have to find your executable again. You don't have to worry about command arguments or the working directory.

Sharon: That sounds very convenient. And what if they have CodeAnalyst already installed and then they want to install Visual Studio 2010, is there another different process that they have to go through?

Frank: They'll need to reinstall CodeAnalyst, but fortunately for you we are doing quarterly updates so there should be a new version of CodeAnalyst out with better features.

Sharon: There you go, so you should be upgrading your CodeAnalyst every quarter anyways.

Frank: Why not?

Sharon: Good to know. Ok, what type of code can you profile with CodeAnalyst?

Frank: You can profile any code that can generate PDB information. With Visual Studio, primary users are C++, C, and of course, .net applications. You can use any of those for CodeAnalyst. For the unmanaged code it's easy. You've got the native code, you've got the symbols. It just works like any compiler would. With the managed code you have the "Just In Time" stuff. And CodeAnalyst, we've written a profiling agent that will be activated when your code is compiled on the fly and we'll capture the information, the native code that's generated, so we can give you the source line information for the "just in time" stuff that isn't generated until you actually take the profile.

Sharon: Interesting, so then how do you optimize that?

Frank: Right. You're given the assembly level information if you want it, or you can go down the source line. It's a little trickier with "just in time" or managed stuff because you're not sure how the virtual machine is going to render it. But, if you're trying multiple things, you can use the same function three different ways and see how the virtual machine renders it. See if it's more efficient one way or the other and of course, the data, the way you

structure your data can matter a lot. So, if you have loops, nested loops, accessing data the way the classic example, which is shipped with CodeAnalyst does, you can see how data cache misses will occur. If the Visual Studio 2010 managed code implements that, it will do it the same way. And you will still be able to see, “Hey there was a lot of data cache misses at this source line.” Whether or not it’s been implemented one way or another in assembly with the managed code you’ll still see the misses. So you can say, “I need to re-work this algorithm to make sure that the data is accessed more efficiently”

Sharon: Are there any advantages to using CodeAnalyst within the IDE itself, within Visual Studio? Well, I guess you sort of covered that with not having to find the file.

Frank: Right, the main one is convenience. You are right there within the IDE you don’t have to swap back and forth between different applications. You don’t have to build it and then go to another application to run it and go back and worry about previous versions and such.

Sharon: Is there anything that is limited by using it within the IDE?

Frank: Not that I can think of!

Sharon: Now the other question that we get asked often at Developer Central is, does CodeAnalyst only work on AMD processors?

Frank: No, not at all. On Windows we specifically coded it so any platform with APIC enabled will work and all modern systems have APIC’s. We worried about the APIC’s about when the Athlon™ K7’s were out, but nowadays they’re in everything and that will let timer-based sampling occur which is every so many milliseconds you take a sample. That will work on any system. The vendor specific events or the instruction-based sampling, which we are very proud of, is, unfortunately, limited to only AMD chips right now because that’s the only ones they are implemented on.

Sharon: We have the IP for those, so we can enable them right? Finally, I just wanted to the last question which is what is the CodeAnalyst team working on now? What’s coming up?

Frank: Sure. The June release, we moved to a quarterly release schedule and the June release is going through QA right now. We have fixed a couple of minor things and then implemented NGEN for Visual Studio 2010. The .NET version changes a little each time and therefore some of the tools like NGEN or the output change. So, we have to adapt and move on. NGEN allows you to pre-JIT the managed code for a particular machine. It will generate the native code once so that you don’t have to run the virtual machine on it every time.

Sharon: Ok, so it would be pretty important to download each new version of CodeAnalyst.

Frank: Well, we recommend it, but if it’s not convenient for you all then don’t worry about it. IF there is a particular bug that you find that is bugging you, please let us know! We’ve got the support email. We’ve got the forums and we will try to get that, your specific bug fixed in the next version.

Sharon: Great. Thanks so much for your time Frank. This has been very enlightening.

Frank: Glad to help, thank you.