Debugger Support

Enumerating Threads

Virtual threads are just objects in the heap, there may be millions of them. There is no API support in the debugger API for enumerating all virtual threads.

JDI VirtualThreads.allThreads() (and JDWP VirtualMachine/AllThreads) enumerates all active platform threads, virtual threads are not enumerated.

For thread dumps and troubleshooting purposes the debugger can invoke the com.sun.management.Threads API in the target VM to enumerate threads. This can be used as stop-gap solution until there is better support for structured debugging in the APIs.

Thread groups

Virtual threads are not active members of a thread group.

JDI ThreadGroupReference::threads (and JDWP ThreadGroupReference/Children) enumerate all active platform threads in a group, virtual threads are not enumerated.

ThreadStart/ThreadEnd events

JDI ThreadStartEvent/ThreadDeathEvents are sent for all threads when enabled. This may impact performance if there are a huge number of virtual threads.

JDWP EventRequest/Set defines a new *PlatformThreadsOnly* filter that can be used when requesting THREAD_START and THREAD_END events. This allows these events to be filtered for virtual threads so they are not sent to the front-end/debugger.

JDI ThreadStartRequest/ThreadDeathRequest define a new method to control whether thread start/end events are sent for all threads or only platform threads.

IsVirtual

JDI ThreadReference defines isVirtual() to test if a thread is a virtual thread.

JDWP ThreadReference/IsVirtual is the equivalent.

Testing if target VM supports virtual threads

The following is temporary, to allow debuggers to distinguish JDK 17 EA builds from Loom EA builds.

JDI VirtualMachine defines supportVirtualThreads() to test if the target VM supports virtual threads.

The supportsVirtualThread boolean in the reply to the JDWP CapabilitiesNew command is the equivalent.

Not Supported

The following are not currently supported for virtual threads:

- JDI ThreadReference.stop
- JDI ThreadReference.interrupt
- JDI ThreadReference.popFrame
- JDI ThreadReference.forceEarlyReturn
- JDI StackFrame.setValue

JDWP agent options

As a temporary solution to allow existing debuggers work with virtual threads, the JDWP agent will track virtual threads so they can be enumerated for debuggers that want to enumerate all virtual threads. The options that control this behaviour are:

Option Name and Value	Description	Default
enumeratevthreads=y|n	thread lists include vthreads	n
trackvthreads=some|all	track some or all vthreads	some
fakevthreadstartevent=y|n	send fake start event when needed	y

- enumeratevthreads control whether or not virtual threads are included in list of threads returned by JDWP VirtualThread/AllThreads command.

- trackvthreads controls which virtual threads will be returned when enumeratevthreads=y.
- fakevthreadstartevent is independent of the two others. If y, then before sending an event, send a fake THREAD_START event for if the debugger hasn't already been notified about the virtual thread that the event occurs on. Note currently there is a bug, and if trackvthreads=all, no fake THREAD_START will ever be sent. This is only an issue if the debug agent is started after some virtual threads have been created.

Links

Project Loom Early Access builds

Java Debug Interface (JDI)

JDWP protocol details

JVM Tool Interface

Java Native Interface

Slides from March 24, 2021 meeting with IntelliJ and Eclipse maintainers

Sample application (uses Helidon MP configured to run each service in its own virtual thread)

Eclipse Bug 527000 tracks adding support for virtual threads