JVM Tool Interface (JVM TI)

Implementation in HotSpot

Robert.Field@Sun.com
JVM TI Implementation – Overview

• Implementation issues
• HotSpot implementation layers
• Implementation details
• Q & A
Implementation Issues

• Capabilities
• Early start-up
• Multiple environments
• Retransformation / Redefinition
Implementation Issues: Capabilities

- Agent requests what capabilities it wants
- Allows pay-for-what-you eat
- Allows implementation subsets
- Capabilities change VM configuration:
  > interpreter, compiler, ...
- Dynamic configuration?

```
can_tag_objects
can_generate_field_modification_events
can_get_owned_monitor_info
can_pop_frame
can_redefine_classes
```
Implementation Issues: Early start-up

- JVM TI agents start before VM initialized
- JVM TI events can be sent before main()
- VM in delicate states
- Allows VM configuration
Implementation Issues: Environments

- Each has its own JVM TI environment
- Can be multiple environments in one VM
- Each with its own capabilities, events, state, ...
- Hidden from VM Core
Implementation Issues: Redefinition

- Class redefinition allows an agent to replace the definition of an already loaded class
- Class retransformation allows an agent to transform an already loaded class
- Retransformation added in JDK6
- Used for bytecode instrumentation
Implementation: Layers

• User agent
  > JVM TI
• JVM TI View
  > jvmtiEnv.hpp
• JVM TI Implementation
  > jvmtiExport.hpp
• VM Core
Implementation: Layers – JVM TI View

- Transitions from JVM TI (external C interface)
- To HotSpot implementation of JVM TI
  > HotSpot types
  > C++ calls
- Both interfaces and transition code generated
  > jvmti.h (JVM TI standard interface – C-interface)
  > jvmtiEnter.cpp (transition code)
  > jvmtiEnv.hpp (interface to HotSpot implementation)
Implementation: Layers - jvmtiExport

• Communicate from JVM TI to the VM Core
  > Information that will be needed (capabilities)
    > Thus what to preserve
    > Which events to send
• Send events from the VM Core
• Shield JVM TI internals from VM Core
Function Flow
Event Architecture
Event Flow
Implementation Details

• Threads and Environments
• Event Controller
• Generated Code
• Some interesting source files:
  > jvmtiRedefineClasses.cpp
  > jvmtiClassFileReconstituter.cpp
  > jvmtiTagMap.cpp
Thread / Environment Data

Per Thread
- JvmtiEnvThreadStateException
  - Per thread enablement
  - Frame pop info

All Threads
- JvmtiEnv
  - Global enablement
  - Event handlers
  - Capabilities

Per Environment
- JvmtiThreadStateException
  - Interpret only mode
  - Current stack depth

All Environments
- All other classes/data
  - ...everything else ...
Event Controller

JvmtiEventController

Thread-Start

JvmtiExport

Method enter/exit

User Enable
Set Callbacks
New Env
Set Frame Pop

JvmtiEnv

User Enabled
Computed Enabled
Event Controller

JvmtiEventController
- Computed Enabled
  - User Enabled
  - Set Callbacks
  - New Env
  - Set Frame Pop

JvmtiExport
- Method enter/exit

JvmtiEventEnabled
- Enabled

JvmtiEnv
- User Enabled
  - Computed Enabled

Thread-Start

JvmtiThreadState
- interp_only_mode
- Frame pops
  - User Enabled
  - Computed Enabled

User Enable
Set Callbacks
New Env
Set Frame Pop
Generated Code: Originates from Spec

jvmti.xml
Generated Code: Generated by XSL

```
jvmti.xml

jvmti.xsl

jvmtiH.xsl
jvmtiHpp.xsl
jvmtiEnter.xsl
jvmtiEnv.xsl
```

```
JvmtiLib.xsl
```
Generated Code: Interfaces, Code, Doc

jvmti.html

jvmti.xsl

jvmti.xml

jvmtiH.xsl

jvmtiHpp.xsl

jvmtiEnter.xsl

jvmdiEnter.xsl

jvmtiEnv.xsl

jvmtiLib.xsl

jvmti.h

jvmtiEnv.hpp

jvmtiEnter.cpp

jvmdiEnter.cpp

jvmtiEnvStub.cpp
Generated Code: Stubs filled

- jvmti.html
- jvmti.xsl
- jvmti.xml
- jvmtiH.xsl
- jvmtiHpp.xsl
- jvmtiEnter.xsl
- jvmdEnter.xsl
- jvmtiEnvFill.java
- jvmtiEnvRecommended.cpp
- jvmtiEnvStub.cpp
- jvmtiEnv.cpp
- jvmtiEnv.xsl
- jvmtiEnter.cpp
- jvmtiEnv.cpp
- jvmtiEnvStub.cpp
- JvmtiLib.xsl
jvmtiRedefineClasses.cpp

- Implemented as VM_Operation
- Used by both RedefineClasses and RetransformClasses
- Dan Daugherty owns this now
- Rocket science
- See doc in jvmtiRedefineClasses.hpp
jvmtiClassFileReconstituter.cpp

- Retransformation uses the redefinition code
- But first, HotSpot data structures must be converted to a class file
jvmtiTagMap.cpp

- Implements heap functionality (heap iteration and walks)
- JVM TI heap functionality identifies objects by user supplied tags
- jvmtiTagMap.cpp maps oops to JNI weak ref and tag
jvmtiManageCapabilities.cpp

• Checks availability of requested capabilities
• Merges with other environments
• Converts capabilities to JvmtiExport flags

```cpp
JvmtiExport::set_can_post_exceptions(
    avail.can_generate_exception_events ||
    avail.can_generate_frame_pop_events ||
    avail.can_generate_method_exit_events);
```
JVM TI – Implementation in HotSpot

Q & A
Function Origins

JVMTI (Total: 116)

New JVMTI: 27

From JVMDI: 84
Morphed
(from just JVMDI: 71)

All three: 13

JVMPIDI discarded: 9

From just JVMPIDI: 4
Cloned

To Scale
Event Origins

JVMTI (Total: 36)

From JVMDI: 17
Morphed (from just JVMDI: 11)

New JVMTI: 5

From just JVMPI: 14
Cloned

All three: 6

JVMDI

JVMPI
discarded: 17