

# Class `sun.tools.debug.RemoteThread`

```
java.lang.Object
|
+----sun.tools.debug.RemoteValue
|
+----sun.tools.debug.RemoteObject
|
+----sun.tools.debug.RemoteThread
```

---

```
public class RemoteThread
extends RemoteObject
```

The `RemoteThread` class allows access to a thread in a remote Java interpreter.

## See Also:

[RemoteDebugger](#), [RemoteThreadGroup](#)

## Author:

Thomas Ball

---

## Method Index

- o **[cont\(\)](#)**  
Resume this thread from a breakpoint, unless it previously suspended.
- o **[down\(int\)](#)**  
Change the current stackframe to be one or more frames lower (as in, toward the current program counter).
- o **[dumpStack\(\)](#)**  
Dump the stack.
- o **[getCurrentFrame\(\)](#)**  
Get the current stack frame.
- o **[getCurrentFrameIndex\(\)](#)**  
Return the current stackframe index
- o **[getName\(\)](#)**  
Return the name of the thread.
- o **[getStackVariable\(String\)](#)**  
Return a stack variable from the current stackframe.
- o **[getStackVariables\(\)](#)**  
Return the arguments and local variable from the current stackframe.

- o **getStatus()**  
Return the thread status description
- o **isSuspended()**  
Return whether this thread is suspended.
- o **next()**  
Continue execution of this thread to the next line, but don't step into a method call.
- o **resetCurrentFrameIndex()**  
Reset the current stackframe
- o **resume()**  
Resume execution of this thread.
- o **setCurrentFrameIndex(int)**  
Set the current stackframe index
- o **step(boolean)**  
Continue execution of this thread to the next instruction or line.
- o **stop()**  
Stop the remote thread.
- o **suspend()**  
Suspend execution of this thread.
- o **up(int)**  
Change the current stackframe to be one or more frames higher (as in, away from the current program counter).

## Methods

### o **getName**

```
public String getName() throws Exception
```

Return the name of the thread.

### o **getCurrentFrameIndex**

```
public int getCurrentFrameIndex()
```

Return the current stackframe index

### o **setCurrentFrameIndex**

```
public void setCurrentFrameIndex(int iFrame)
```

Set the current stackframe index

### o **resetCurrentFrameIndex**

```
public void resetCurrentFrameIndex()
```

Reset the current stackframe

## o up

```
public void up(int nFrames) throws Exception
```

Change the current stackframe to be one or more frames higher (as in, away from the current program counter).

**Parameters:**

nFrames – the number of stackframes

**Throws:** IllegalAccessError

when the thread isn't suspended or waiting at a breakpoint

**Throws:** ArrayIndexOutOfBoundsException

when the requested frame is beyond the stack boundary

## o down

```
public void down(int nFrames) throws Exception
```

Change the current stackframe to be one or more frames lower (as in, toward the current program counter).

**Parameters:**

nFrames – the number of stackframes

**Throws:** IllegalAccessError

when the thread isn't suspended or waiting at a breakpoint

**Throws:** ArrayIndexOutOfBoundsException

when the requested frame is beyond the stack boundary

## o getStatus

```
public String getStatus() throws Exception
```

Return the thread status description

## o dumpStack

```
public RemoteStackFrame[] dumpStack() throws Exception
```

Dump the stack.

## o getCurrentFrame

```
public RemoteStackFrame getCurrentFrame() throws Exception
```

Get the current stack frame.

**Throws:** IllegalAccessError

when the thread isn't suspended or waiting at a breakpoint

## o suspend

```
public void suspend() throws Exception
```

Suspend execution of this thread.

**o resume**

```
public void resume() throws Exception
```

Resume execution of this thread.

**o step**

```
public void step(boolean skipLine) throws Exception
```

Continue execution of this thread to the next instruction or line.

**Parameters:**

skipLine – true to execute to next source line, false to next instruction.

**Throws: IllegalAccessError**

when the thread isn't suspended or waiting at a breakpoint

**o next**

```
public void next() throws Exception
```

Continue execution of this thread to the next line, but don't step into a method call. If no line information is available, next() is equivalent to step().

**Throws: IllegalAccessError**

when the thread isn't suspended or waiting at a breakpoint

**o isSuspended**

```
public boolean isSuspended()
```

Return whether this thread is suspended.

**o cont**

```
public void cont() throws Exception
```

Resume this thread from a breakpoint, unless it previously suspended.

**o stop**

```
public void stop() throws Exception
```

Stop the remote thread.

**o getStackVariable**

```
public RemoteStackVariable getStackVariable(String name) throws Exception
```

Return a stack variable from the current stackframe.

**Returns:**

the variable as a RemoteValue, or null if not found.

**o getStackVariables**

```
public RemoteStackVariable[] getStackVariables() throws Exception
```

Return the arguments and local variable from the current stackframe.

**Returns:**

an array of RemoteValues.

---

[All Packages](#)

[This Package](#)

[Previous](#)

[Next](#)