---

package sun.tools.debug

Interface Index

- DebuggerCallback

Class Index

- RemoteArray
- RemoteBoolean
- RemoteByte
- RemoteChar
- RemoteClass
- RemoteDebugger
- RemoteDouble
- RemoteField
- RemoteFloat
- RemoteInt
- RemoteLong
- RemoteObject
- RemoteShort
- RemoteStackFrame
- RemoteStackVariable
- RemoteString
- RemoteThread
- RemoteThreadGroup
- RemoteValue
- StackFrame

Interface sun.tools.debug.DebuggerCallback

public interface **DebuggerCallback**
extends Object

The DebuggerCallback interface is used to communicate asynchronous information from the debugger to its client. This may be the actual client object, or a delegate of its choosing.

Method Index

Methods

printToConsole

```
public abstract void printToConsole(String text) throws Exception
```

Print text to the debugger's console window.

breakpointEvent

```
public abstract void breakpointEvent(RemoteThread t) throws Exception
```

A breakpoint has been hit in the specified thread.

exceptionEvent

```
public abstract void exceptionEvent(RemoteThread t,
                                    String errorText) throws Exception
```

An exception has occurred.

threadDeathEvent

```
public abstract void threadDeathEvent(RemoteThread t) throws Exception
```

A thread has died.

quitEvent

```
public abstract void quitEvent() throws Exception
```

The client interpreter has exited, either by returning from its main thread, or by calling System.exit().

Class sun.tools.debug.RemoteArray

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

---

public class **RemoteArray**
extends RemoteObject

The RemoteArray class allows remote debugging of arrays.

See Also:
RemoteValue, RemoteDebugger

Method Index

Methods

getSize

```
public final int getSize()
```

Return the number of elements in the array.

typeName

```
public String typeName()
```

Return this RemoteValue's type ("array").
Overrides:
typeName in class RemoteObject

arrayTypeName

```
public String arrayTypeName(int type)
```

Return the element type as a string.

getElementType

```
public final int getElementType() throws Exception
```

Return the element type as a "TC_" constant, such as "TC_CHAR".

getElement

```
public final RemoteValue getElement(int index) throws Exception
```

Return an array element.
Parameters:
index – the index of the element
Returns:
the element as a RemoteValue
Throws: ArrayIndexOutOfBoundsException
when the index is greater than the size of the array

getElements

```
public final RemoteValue[] getElements() throws Exception
```

Returns a copy of the array as instances of RemoteValue.

getElements

```
public final RemoteValue[] getElements(int beginIndex,
                                       int endIndex) throws Exception
```

Returns a copy of a portion of the array as instances of RemoteValue.

Parameters:

beginIndex – the beginning array index

endIndex – the final array index

Throws: ArrayIndexOutOfBoundsException

when the index is greater than the size of the array

description

```
public String description()
```

Return a description of the array.

Overrides:

description in class RemoteObject

toString

```
public String toString()
```

Return a string version of the array.

Overrides:

toString in class RemoteObject

Class sun.tools.debug.RemoteBoolean

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

---

public class **RemoteBoolean**
extends RemoteValue

The RemoteBoolean class extends RemoteValue for booleans.

See Also:
RemoteValue, RemoteDebugger

Method Index

Methods

get

```
public boolean get()
```

Return the boolean's value.

typeName

```
public String typeName()
```

Print this RemoteValue's type ("boolean").

Overrides:

typeName in class RemoteValue

toString

```
public String toString()
```

Return the boolean's value as a string.

Overrides:

toString in class Object

Class sun.tools.debug.RemoteByte

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

---

public class **RemoteByte**
extends RemoteValue

The RemoteByte class extends RemoteValue for bytes.

See Also:
RemoteValue, RemoteDebugger

Method Index

- **get**()
  Return the byte's value.
- **toString**()
  Return the byte's value as a string.
- **typeName**()
  Print this RemoteValue's type ("byte").

Methods

get

```
public byte get()
```

Return the byte's value.

typeName

```
public String typeName()
```

Print this RemoteValue's type ("byte").
Overrides:
typeName in class RemoteValue

toString

```
public String toString()
```

Return the byte's value as a string.
Overrides:
toString in class Object

Class sun.tools.debug.RemoteChar

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

---

public class **RemoteChar**
extends RemoteValue

The RemoteChar class extends RemoteValue for chars.

See Also:
RemoteValue, RemoteDebugger

Method Index

Methods

get

```
public char get()
```

Return the char's value.

typeName

```
public String typeName()
```

Print this RemoteValue's type ("char").
Overrides:
typeName in class RemoteValue

toString

```
public String toString()
```

Return the char's value as a string.
Overrides:
toString in class Object

Class sun.tools.debug.RemoteClass

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

---

public class **RemoteClass**
extends RemoteObject

The RemoteClass class allows access to a class in a remote Java interpreter.

See Also:
RemoteDebugger

Method Index

- **catchExceptions**()
  Enter the debugger when an instance of this class is thrown.
- **clearBreakpoint**(int)
  Clear a breakpoint at a specific address in a class.
- **clearBreakpointLine**(int)
  Clear a breakpoint at a specified line.
- **clearBreakpointMethod**(RemoteField)
  Clear a breakpoint at the start of a specified method.
- **description**()
  Return a (somewhat verbose) description.
- **getClassLoader**()
  Return the classloader for this class.
- **getField**(int)
  Return the static field, specified by index.
- **getField**(String)
  Return the static field, specified by name.
- **getFieldValue**(int)
  Return the value of a static field, specified by its index
- **getFieldValue**(String)
  Return the value of a static field, specified by name.

- **getFields**()
  Return all the static fields for this class.
- **getInstanceField**(int)
  Return the instance field, specified by its index.
- **getInstanceFields**()
  Return all the instance fields for this class.
- **getInterfaces**()
  Return the interfaces for this class.
- **getMethod**(String)
  Return the method, specified by name.
- **getMethodNames**()
  Return the names of all methods supported by this class.
- **getMethods**()
  Return the class's methods.
- **getName**()
  Returns the name of the class.
- **getSourceFile**()
  Get the source file referenced by this stackframe.
- **getSourceFileName**()
  Get the name of the source file referenced by this stackframe.
- **getStaticFields**()
  Return all the static fields for this class.
- **getSuperclass**()
  Return the superclass for this class.
- **ignoreExceptions**()
  Don't enter the debugger when an instance of this class is thrown.
- **isInterface**()
  Is this RemoteClass an interface?
- **setBreakpointLine**(int)
  Set a breakpoint at a specified source line number in a class.
- **setBreakpointMethod**(RemoteField)
  Set a breakpoint at the first line of a class method.
- **toString**()
  Return a (somewhat verbose) description.
- **typeName**()
  Returns the name of the class as its type.

getName

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

Returns the name of the class.

typeName

```
public String typeName() throws Exception
```

Returns the name of the class as its type.
Overrides:
typeName in class RemoteObject

isInterface

```
public boolean isInterface() throws Exception
```

Is this RemoteClass an interface?

getSuperclass

```
public RemoteClass getSuperclass() throws Exception
```

Return the superclass for this class.

getClassLoader

```
public RemoteObject getClassLoader() throws Exception
```

Return the classloader for this class.

getInterfaces

```
public RemoteClass[] getInterfaces() throws Exception
```

Return the interfaces for this class.

getSourceFileName

```
public String getSourceFileName()
```

Get the name of the source file referenced by this stackframe.

getSourceFile

```
public InputStream getSourceFile() throws Exception
```

Get the source file referenced by this stackframe.

getFields

```
public RemoteField[] getFields() throws Exception
```

Return all the static fields for this class.
Overrides:
getFields in class RemoteObject

getStaticFields

```
public RemoteField[] getStaticFields() throws Exception
```

Return all the static fields for this class.

getInstanceFields

```
public RemoteField[] getInstanceFields() throws Exception
```

Return all the instance fields for this class. Note: because this is a RemoteClass method, only the name and type methods will be valid, not the data.

getField

```
public RemoteField getField(int n) throws Exception
```

Return the static field, specified by index.
Throws: ArrayIndexOutOfBoundsException
when the index is greater than the number of instance variables
Overrides:
getField in class RemoteObject

getField

```
public RemoteField getField(String name) throws Exception
```

Return the static field, specified by name.
Overrides:
getField in class RemoteObject

getInstanceField

```
public RemoteField getInstanceField(int n) throws Exception
```

Return the instance field, specified by its index. Note: because this is a RemoteClass method, only the name and type information is valid, not the data.
Throws: ArrayIndexOutOfBoundsException
when the index is greater than the number of instance variables

getFieldValue

```
public RemoteValue getFieldValue(int n) throws Exception
```

Return the value of a static field, specified by its index
Overrides:
getFieldValue in class RemoteObject

getFieldValue

```
public RemoteValue getFieldValue(String name) throws Exception
```

Return the value of a static field, specified by name.

Overrides:

getFieldValue in class RemoteObject

getMethod

```
public RemoteField getMethod(String name) throws Exception
```

Return the method, specified by name.

getMethods

```
public RemoteField[] getMethods() throws Exception
```

Return the class's methods.

getMethodNames

```
public String[] getMethodNames() throws Exception
```

Return the names of all methods supported by this class.

setBreakpointLine

```
public String setBreakpointLine(int lineno) throws Exception
```

Set a breakpoint at a specified source line number in a class.

Parameters:

lineno – the line number where the breakpoint is set

Returns:

an empty string if successful, otherwise a description of the error.

setBreakpointMethod

```
public String setBreakpointMethod(RemoteField method) throws Exception
```

Set a breakpoint at the first line of a class method.

Parameters:

method – the method where the breakpoint is set

Returns:

an empty string if successful, otherwise a description of the error.

clearBreakpoint

```
public String clearBreakpoint(int pc) throws Exception
```

Clear a breakpoint at a specific address in a class.
Parameters:
pc – the address of the breakpoint to be cleared
Returns:
an empty string if successful, otherwise a description of the error.

clearBreakpointLine

```
public String clearBreakpointLine(int lineno) throws Exception
```

Clear a breakpoint at a specified line.
Parameters:
lineno – the line number of the breakpoint to be cleared
Returns:
an empty string if successful, otherwise a description of the error.

clearBreakpointMethod

```
public String clearBreakpointMethod(RemoteField method) throws Exception
```

Clear a breakpoint at the start of a specified method.
Parameters:
method – the method of the breakpoint to be cleared
Returns:
an empty string if successful, otherwise a description of the error.

catchExceptions

```
public void catchExceptions() throws Exception
```

Enter the debugger when an instance of this class is thrown.
Throws: ClassCastException
when this class isn't an exception class

ignoreExceptions

```
public void ignoreExceptions() throws Exception
```

Don't enter the debugger when an instance of this class is thrown.
Throws: ClassCastException
when this class isn't an exception class

description

```
public String description()
```

Return a (somewhat verbose) description.
Overrides:

description in class RemoteObject

toString

```
public String toString()
```

Return a (somewhat verbose) description.

Overrides:

toString in class RemoteObject

Class sun.tools.debug.RemoteDebugger

```
java.lang.Object
   |
   +----sun.tools.debug.RemoteDebugger
```

---

public class **RemoteDebugger**
extends Object

The RemoteDebugger class defines a client interface to the Java debugging classes. It is used to instantiate a connection with the Java interpreter being debugged.

Constructor Index

- **RemoteDebugger**(String, String, DebuggerCallback, boolean)
  Create a remote debugger, connecting it with a running Java interpreter.

  To connect to a running interpreter, it must be started with the "–debug" option, whereupon it will print out the password for that debugging session.
- **RemoteDebugger**(String, DebuggerCallback, boolean)
  Create a remote debugger, and connect it to a new client interpreter.

Method Index

- **close**()
  Close the connection to the remote debugging agent.
- **findClass**(String)
  Find a specified class.
- **freeMemory**()
  Report the free memory available to the Java interpreter being debugged.
- **gc**(RemoteObject[])
  Free all objects referenced by the debugger.
- **get**(Integer)
  Get an object from the remote object cache.
- **getExceptionCatchList**()
  Return the list of the exceptions the debugger will stop on.
- **getSourcePath**()
  Return the source file path the Agent is currently using.

- **itrace**(boolean)
  Turn on/off instruction tracing.
- **listBreakpoints**()
  Return a list of the breakpoints which are currently set.
- **listClasses**()
  List the currently known classes.
- **listThreadGroups**(RemoteThreadGroup)
  List threadgroups
- **run**(int, String[])
  Load and run a runnable Java class, with any optional parameters.
- **setSourcePath**(String)
  Specify the list of paths to use when searching for a source file.
- **totalMemory**()
  Report the total memory usage of the Java interpreter being debugged.
- **trace**(boolean)
  Turn on/off method call tracing.

Constructors

RemoteDebugger

```
public RemoteDebugger(String host,
                      String password,
                      DebuggerCallback client,
                      boolean verbose) throws Exception
```

Create a remote debugger, connecting it with a running Java interpreter.

To connect to a running interpreter, it must be started with the "–debug" option, whereupon it will print out the password for that debugging session.

Parameters:

host – the name of the system where a debuggable Java instance is running (default is localhost).
password – the password reported by the debuggable Java instance. This should be null when starting a client interpreter.
client – the object to which notification messages are sent (it must support the DebuggerCallback interface)
verbose – turn on internal debugger message text

RemoteDebugger

```
public RemoteDebugger(String javaArgs,
                      DebuggerCallback client,
                      boolean verbose) throws Exception
```

Create a remote debugger, and connect it to a new client interpreter.

Parameters:

javaArgs – optional java command–line parameters, such as –classpath
client – the object to which notification messages are sent (it must support the DebuggerCallback interface)
verbose – turn on internal debugger message text

Methods

close

```
public void close()
```

Close the connection to the remote debugging agent.

get

```
public RemoteObject get(Integer id)
```

Get an object from the remote object cache.
Parameters:
id – the remote object's id
Returns:
the specified RemoteObject, or null if not cached.

listClasses

```
public RemoteClass[] listClasses() throws Exception
```

List the currently known classes.

findClass

```
public RemoteClass findClass(String name) throws Exception
```

Find a specified class. If the class isn't already known by the remote debugger, the lookup request will be passed to the Java interpreter being debugged. NOTE: Substrings, such as "String" for "java.lang.String" will return with the first match, and will not be successfully found if the request is passed to the remote interpreter.
Parameters:
name – the name (or a substring of the name)of the class
Returns:
the specified (Remote)Class, or null if not found.

listThreadGroups

```
public RemoteThreadGroup[] listThreadGroups(RemoteThreadGroup tg) throws Exception
```

List threadgroups

Parameters:

tg – the threadgroup which hold the groups to be listed, or null for all threadgroups

gc

```
public void gc(RemoteObject save_list[]) throws Exception
```

Free all objects referenced by the debugger. The remote debugger maintains a copy of each object it has examined, so that references won't become invalidated by the garbage collector of the Java interpreter being debugged. The gc() method frees all all of these references, except those specified to save.

Parameters:

save_list – the list of objects to save.

trace

```
public void trace(boolean traceOn) throws Exception
```

Turn on/off method call tracing. When turned on, each method call is reported to the stdout of the Java interpreter being debugged. This output is not captured in any way by the remote debugger.

Parameters:

traceOn – turn tracing on or off

itrace

```
public void itrace(boolean traceOn) throws Exception
```

Turn on/off instruction tracing. When turned on, each Java instruction is reported to the stdout of the Java interpreter being debugged. This output is not captured in any way by the remote debugger.

Parameters:

traceOn – turn tracing on or off

totalMemory

```
public int totalMemory() throws Exception
```

Report the total memory usage of the Java interpreter being debugged.

freeMemory

```
public int freeMemory() throws Exception
```

Report the free memory available to the Java interpreter being debugged.

run

```
public RemoteThreadGroup run(int argc,
                             String argv[]) throws Exception
```

Load and run a runnable Java class, with any optional parameters. The class is started inside a new threadgroup in the Java interpreter being debugged. NOTE: Although it is possible to run multiple runnable classes from the same Java interpreter, there is no guarantee that all applets will work cleanly with each other. For example, two applets may want exclusive access to the same shared resource,such as a specific port.

Parameters:

argc – the number of parameters

argv – the array of parameters: the class to be run is first, followed by any optional parameters used by that class.

Returns:

the new ThreadGroup the class is running in, or null on error

listBreakpoints

```
public String[] listBreakpoints() throws Exception
```

Return a list of the breakpoints which are currently set.

Returns:

an array of Strings of the form "class_name:line_number".

getExceptionCatchList

```
public String[] getExceptionCatchList() throws Exception
```

Return the list of the exceptions the debugger will stop on.

Returns:

an array of exception class names, which may be zero–length.

getSourcePath

```
public String getSourcePath() throws Exception
```

Return the source file path the Agent is currently using.

Returns:

a string consisting of a list of colon–delineated paths.

setSourcePath

```
public void setSourcePath(String pathList) throws Exception
```

Specify the list of paths to use when searching for a source file.

Parameters:

pathList – a string consisting of a list of colon–delineated paths.

Class sun.tools.debug.RemoteDouble

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

---

public class **RemoteDouble**
extends RemoteValue

The RemoteDouble class extends RemoteValue for doubles.

See Also:
RemoteValue, RemoteDebugger

Method Index

Methods

get

```
public double get()
```

Return the double's value.

typeName

```
public String typeName()
```

Print this RemoteValue's type ("double").
Overrides:
typeName in class RemoteValue

toString

```
public String toString()
```

Return the double's value as a string.
Overrides:
toString in class Object

Class sun.tools.debug.RemoteField

```
java.lang.Object
   |
   +----sun.tools.debug.Field
           |
           +----sun.tools.debug.RemoteField
```

---

public class **RemoteField**
extends Field
implements AgentConstants

A RemoteField allows access to a variable or method of an object or class in a remote Java interpreter.

See Also:
RemoteStackVariable

Method Index

- **getModifiers**()
  Returns a string with the field's modifiers, such as "public", "static", "final", etc.
- **getName**()
  Returns the name of the field.
- **getType**()
  Returns a type string describing the field.
- **isStatic**()
  Returns whether the field is static (a class variable or method).
- **toString**()
  Returns a String that represents the value of this Object.

Methods

getName

```
public String getName()
```

Returns the name of the field.

getType

```
public String getType()
```

Returns a type string describing the field.

getModifiers

```
public String getModifiers()
```

Returns a string with the field's modifiers, such as "public", "static", "final", etc. If the field has no modifiers, an empty String is returned.

isStatic

```
public boolean isStatic()
```

Returns whether the field is static (a class variable or method).

toString

```
public String toString()
```

Returns a String that represents the value of this Object.
Overrides:
toString in class Object

Class sun.tools.debug.RemoteFloat

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

---

public class **RemoteFloat**
extends RemoteValue

The RemoteFloat class extends RemoteValue for floats.

See Also:
RemoteValue, RemoteDebugger

Method Index

- **get**()
  Return the float's value.
- **toString**()
  Return the float's value as a string.
- **typeName**()
  Print this RemoteValue's type ("float").

Methods

get

```
public float get()
```

Return the float's value.

typeName

```
public String typeName()
```

Print this RemoteValue's type ("float").
Overrides:
typeName in class RemoteValue

toString

```
public String toString()
```

Return the float's value as a string.
Overrides:
toString in class Object

Class sun.tools.debug.RemoteInt

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

---

public class **RemoteInt**
extends RemoteValue

The RemoteInt class extends RemoteValue for ints.

See Also:
RemoteValue, RemoteDebugger

Constructor Index

- **RemoteInt**(int)

Method Index

- **get**()
  Return the int's value.
- **toString**()
  Return the int's value as a string.
- **typeName**()
  Print this RemoteValue's type ("int").

Constructors

RemoteInt

```
public RemoteInt(int i)
```

Methods

get

```
public int get()
```

Return the int's value.

typeName

```
public String typeName()
```

Print this RemoteValue's type ("int").
Overrides:
typeName in class RemoteValue

toString

```
public String toString()
```

Return the int's value as a string.
Overrides:
toString in class Object

Class sun.tools.debug.RemoteLong

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

---

public class **RemoteLong**
extends RemoteValue

The RemoteLong class extends RemoteValue for longs.

See Also:
RemoteValue, RemoteDebugger

Method Index

Methods

get

```
public long get()
```

Return the long's value.

typeName

```
public String typeName()
```

Print this RemoteValue's type ("long").
Overrides:
typeName in class RemoteValue

toString

```
public String toString()
```

Return the long's value as a string.
Overrides:
toString in class Object

Class sun.tools.debug.RemoteObject

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

---

public class **RemoteObject**
extends RemoteValue

The RemoteObject class allows access to an object in a remote Java interpreter. Remote objects are not created by the local debugger, but are returned by the remote debugging agent when queried for the values of instance or static variables of known objects (such as classes), or from local (stack) variables. Each remote object has a reference cached by the remote Java interpreter, so that the object will not be garbage–collected during examination. The RemoteDebugger's gc() operation frees references to objects that are no longer being examined.

See Also:

RemoteDebugger, RemoteClass, RemoteString, RemoteThread, RemoteThreadGroup

---

Method Index

- **description**()
  Return a description of the object.
- **getClazz**()
  Returns the object's class.
- **getField**(int)
  Return an instance variable, specified by slot number.
- **getField**(String)
  Return an instance variable, specified by name.
- **getFieldValue**(int)
  Returns the value of an object's instance variable.
- **getFieldValue**(String)
  Returns the value of an object's instance variable.
- **getFields**()
  Return the instance (non–static) fields of an object.

- **getId**()
  Returns the id of the object.
- **toString**()
  Return object as a string.
- **typeName**()
  Returns the RemoteValue's type name ("Object").

Methods

typeName

```
public String typeName() throws Exception
```

Returns the RemoteValue's type name ("Object").

Overrides:

typeName in class RemoteValue

getId

```
public final int getId()
```

Returns the id of the object.

getClazz

```
public final RemoteClass getClazz()
```

Returns the object's class.

getFieldValue

```
public RemoteValue getFieldValue(int n) throws Exception
```

Returns the value of an object's instance variable.

Parameters:

n – the slot number of the variable to be returned.

getFieldValue

```
public RemoteValue getFieldValue(String name) throws Exception
```

Returns the value of an object's instance variable.

Parameters:

name – the name of the instance variable

Returns:

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

getFields

```
public RemoteField[] getFields() throws Exception
```

Return the instance (non–static) fields of an object.

getField

```
public RemoteField getField(int n) throws Exception
```

Return an instance variable, specified by slot number.
Parameters:
n – the slot number of the variable to be returned.

getField

```
public RemoteField getField(String name) throws Exception
```

Return an instance variable, specified by name.
Parameters:
name – the name of the instance variable
Returns:
the variable as a RemoteField, or null if name not found.

description

```
public String description()
```

Return a description of the object.
Overrides:
description in class RemoteValue

toString

```
public String toString()
```

Return object as a string.
Overrides:
toString in class Object

Class sun.tools.debug.RemoteShort

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

---

public class **RemoteShort**
extends RemoteValue

The RemoteShort class extends RemoteValue for shorts.

See Also:
RemoteValue, RemoteDebugger

Method Index

- **get**()
  Return the short's value.
- **toString**()
  Return the short's value as a string.
- **typeName**()
  Print this RemoteValue's type ("short").

Methods

get

```
public short get()
```

Return the short's value.

typeName

```
public String typeName()
```

Print this RemoteValue's type ("short").
Overrides:
typeName in class RemoteValue

toString

```
public String toString()
```

Return the short's value as a string.
Overrides:
toString in class Object

Class sun.tools.debug.RemoteStackFrame

```
java.lang.Object
   |
   +----sun.tools.debug.StackFrame
           |
           +----sun.tools.debug.RemoteStackFrame
```

---

public class **RemoteStackFrame**
extends StackFrame

The RemoteStackFrame class provides access to a stackframe of a suspended thread.

See Also:
RemoteDebugger, RemoteThread

Method Index

Methods

getLocalVariable

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

Return a specific (named) stack variable. A slot number of –1 indicates that the variable is not currently in scope.

getLocalVariables

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

Return an array of all valid local variables and method arguments for this stack frame.

getLineNumber

```
public int getLineNumber()
```

Return the source file line number.

getMethodName

```
public String getMethodName()
```

Get the method name referenced by this stackframe.

getPC

```
public int getPC()
```

Get the program counter referenced by this stackframe.

getRemoteClass

```
public RemoteClass getRemoteClass()
```

Get the class this stackframe references.

Class sun.tools.debug.RemoteStackVariable

```
java.lang.Object
   |
   +----sun.tools.debug.LocalVariable
           |
           +----sun.tools.debug.RemoteStackVariable
```

---

public class **RemoteStackVariable**
extends LocalVariable

A RemoteStackVariable represents a method argument or local variable. It is similar to a RemoteField, but is much more transient in nature.

See Also:
RemoteField

Method Index

- **getName**()
  Return the name of a stack variable or argument.
- **getValue**()
  Return the value of a stack variable or argument.
- **inScope**()
  Return whether variable is in scope.

Methods

getName

```
public String getName()
```

Return the name of a stack variable or argument.

getValue

```
public RemoteValue getValue()
```

Return the value of a stack variable or argument.

inScope

```
public boolean inScope()
```

Return whether variable is in scope.

Class sun.tools.debug.RemoteString

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

---

public class **RemoteString**
extends RemoteObject

The RemoteString class allows access to a string in a remote Java interpreter.

See Also:
RemoteDebugger

Method Index

- **description**()
  Return the string value, or "null"
- **toString**()
  Return the string value, or "null"
- **typeName**()
  Print this RemoteValue's type ("String").

Methods

typeName

```
public String typeName()
```

Print this RemoteValue's type ("String").
Overrides:
typeName in class RemoteObject

description

```
public String description()
```

Return the string value, or "null"
Overrides:
description in class RemoteObject

toString

```
public String toString()
```

Return the string value, or "null"
Overrides:
toString in class RemoteObject

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

Method Index

Return whether this thread is suspended.

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

getName

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

Return the name of the thread.

getCurrentFrameIndex

```
public int getCurrentFrameIndex()
```

Return the current stackframe index

setCurrentFrameIndex

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

Set the current stackframe index

resetCurrentFrameIndex

```
public void resetCurrentFrameIndex()
```

Reset the current stackframe

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

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

getStatus

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

Return the thread status description

dumpStack

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

Dump the stack.

getCurrentFrame

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

Get the current stack frame.

Throws: IllegalAccessError

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

suspend

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

Suspend execution of this thread.

resume

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

Resume execution of this thread.

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

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

isSuspended

```
public boolean isSuspended()
```

Return whether this thread is suspended.

cont

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

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

stop

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

Stop the remote thread.

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.

getStackVariables

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

Return the arguments and local variable from the current stackframe.

Returns:

an array of RemoteValues.

Class sun.tools.debug.RemoteThreadGroup

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

---

public class **RemoteThreadGroup**
extends RemoteObject

The RemoteThreadGroup class allows access to a threadgroup in a remote Java interpreter.

See Also:
RemoteDebugger, RemoteThreadGroup

Method Index

Methods

getName

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

Return the threadgroup's name.

stop

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

Stop the remote threadgroup.

listThreads

```
public RemoteThread[] listThreads(boolean recurse) throws Exception
```

List a threadgroup's threads

Class sun.tools.debug.RemoteValue

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

---

public class **RemoteValue**
extends Object
implements AgentConstants

The RemoteValue class allows access to a copy of a value in the remote Java interpreter. This value may be a primitive type, such as a boolean or float, or an object, class, array, etc. Remote values are not created by the local debugger, but are returned by the remote debugging agent when queried for the values of instance or static variables of known objects, or from local (stack) variables.

See Also:

RemoteDebugger, RemoteArray, RemoteBoolean, RemoteByte, RemoteChar, RemoteClass, RemoteDouble, RemoteFloat, RemoteInt, RemoteLong, RemoteObject, RemoteShort, RemoteString, RemoteThread, RemoteThreadGroup

---

Method Index

- **description**()
  Return a description of the RemoteValue.
- **fromHex**(String)
  Convert hexadecimal strings to ints.
- **getType**()
  Returns the RemoteValue's type.
- **isObject**()
  Returns whether the RemoteValue is an Object (as opposed to a primitive type, such as int).
- **toHex**(int)
  Convert an int to a hexadecimal string.
- **typeName**()
  Returns the RemoteValue's type as a string.

Methods

getType

```
public final int getType()
```

Returns the RemoteValue's type.

isObject

```
public final boolean isObject()
```

Returns whether the RemoteValue is an Object (as opposed to a primitive type, such as int).

typeName

```
public abstract String typeName() throws Exception
```

Returns the RemoteValue's type as a string.

description

```
public String description()
```

Return a description of the RemoteValue.

toHex

```
public static String toHex(int n)
```

Convert an int to a hexadecimal string.

fromHex

```
public static int fromHex(String hexStr)
```

Convert hexadecimal strings to ints.

Class sun.tools.debug.StackFrame

```
java.lang.Object
   |
   +----sun.tools.debug.StackFrame
```

---

public class **StackFrame**
extends Object

The StackFrame class represents a stack frame of a suspended thread.

See Also:
RemoteDebugger, RemoteStackFrame, RemoteThread

Constructor Index

- **StackFrame**()

Method Index

- **toString**()
  Returns a String that represents the value of this Object.

Constructors

StackFrame

```
public StackFrame()
```

Methods

toString

```
public String toString()
```

Returns a String that represents the value of this Object.

Overrides:

toString in class Object
