The RMI Proxy

API Reference

Beta 0.3

Esmond Pitt and Bett Koch

Telekinesis Pty Ltd

for D.B.P.

Contents

Contents

Package
com.rmiproxy

Description
Provides the classes needed by RMI clients and servers to implement RMI Proxying.

Class Summary	
Interfaces	
Firewall[xi]	Defines the constants of the RMI application firewall (RMI proxy).
Version[xxxix]	Copyright notice plus automatic revision & date tracker.
Classes	
Configuration[ix]	Holds configuration details for the RMI application firewall.
ProxyNaming[xix]	The main access point to the RMI application firewall.
Exceptions	
FirewallException[xiii]	General firewall exception class.

Package
com.rmiproxy.jndi

Description

Provides a JNDI Service Provider for the RMI Proxy Registry. If the `rmi.proxyHost` property is not set, it is in all respects identical to the JNDI RMI Registry provider.

Class Summary	
Classes	
`RegistryContext`[xxix]	A RegistryContext is a context representing a remote RMI registry.
`RegistryContextFactory`[xxxvii]	A RegistryContextFactory takes an RMI registry reference, and creates the corresponding RMI object or registry context.

Package
com.rmiproxy.registry

Description

Provides the rmiproxy.registry package, for use by RMI clients and servers behind firewalls.
The package is the RMI Proxy version of the java.rmi.registry package.

- The LocateProxyRegistry class supports locating a proxy registry
- The ProxyRegistry interface defines the interface to the proxy registry. It is identical to the java.rmi.Registry interface

Class Summary

Interfaces

ProxyRegistry xxiii	ProxyRegistry defines the remote interface to the proxy server.

Classes

LocateProxyRegistry xv	`LocateRegistry` is used to obtain a reference to a bootstrap remote object registry on a particular host (including the local host), or to create a remote object registry that accepts calls on a specific port.
ProxyRegistry.AbstractServer xxvii	Abstract implementation class, for rmic

com.rmiproxy
Configuration

Syntax

```
public abstract class Configuration

java.lang.Object
  |
  +--com.rmiproxy.Configuration
```

Description

Holds configuration details for the RMI application firewall.

Configuration details are read from a configuration file, which is a standard Java properties file named *rmi-proxy.config*. All hosts involved in the RMI Proxy setup (client, server and all intervening proxies) are required to have a such a file. It is assumed to exist in the current directory (TODO Fix this?), along with any *rmi-proxy.policy* file (required by proxy hosts only).

Currently two values are configurable:

- **rmi.proxyHost**, the RMI URL of the RMI proxy for this host. It must be specified as a URL, e.g. 'rmi://proxyHostName:1099'. For clients and servers, this may be overridden on the command line with '-Drmi.proxyHost'.
- **rmi.nonProxyHosts**, a space-separated list of RMI URL-formatted names. Each item names a host which is locally accessible and does not need to be accessed via the RMI proxy.

Member Summary

Fields

public static final	ConfigFile[ix]
	Name of the configuration file.

Methods

public static Configuration	current()[ix]
public abstract String	getNonProxyHosts()[x]
public abstract String	getProxyHost()[x]
public abstract boolean	isNonProxyHost(String)[x]

ConfigFile

```
public static final java.lang.String ConfigFile
```

Name of the configuration file.

---

current()

```
public static Configuration[ix] current()
```

Returns: The singleton Configuration instance

---

getNonProxyHosts()

```
public abstract java.lang.String getNonProxyHosts()
```

Returns: The non-RMI-proxy hosts as a space-separated list of RMI URL-formatted names, or null if there are none.

RMI requests for these hosts will be forwarded directly and not via the RMI proxy.

---

getProxyHost()

```
public abstract java.lang.String getProxyHost()
```

Returns: The RMI URL (e.g. 'rmi://10.0.1.6') of the proxy host, or null if there is no proxy.

When present, it is always the URL of the RMI Proxy which is closer to the Internet than this host.

---

isNonProxyHost(String)

```
public abstract boolean isNonProxyHost(java.lang.String host)
```

Returns: False iff the specified host is reached via the RMI proxy

com.rmiproxy
Firewall

Syntax
```
public interface Firewall
```

Description
Defines the constants of the RMI application firewall (RMI proxy).

Member Summary	
Fields	
`public static final`	`JRMP_PROXY_PORT`[xi] Port used by the proxy registry.
`public static final`	`NonProxyHostsPropertyName`[xi] Configuration property which names a list of directly accessible hosts.
`public static final`	`PolicyFileName`[xi] Name of the RMI Proxy security policy file.
`public static final`	`ProxyHostPropertyName`[xii] Configuration property which names the RMI proxy host.

JRMP_PROXY_PORT

```
public static final int JRMP_PROXY_PORT
```

Port used by the proxy registry. This is currently hardcoded to be the RMI Registry port.

NonProxyHostsPropertyName

```
public static final java.lang.String NonProxyHostsPropertyName
```

Configuration property which names a list of directly accessible hosts. This is `rmi.nonProxyHosts`.

These hosts will not be accessed via the RMI proxy.

PolicyFileName

```
public static final java.lang.String PolicyFileName
```

Name of the RMI Proxy security policy file.

A security policy file defining access permissions is required for every RMI Proxy. By default, this is `./rmiproxy.policy`, but this can be overridden on the command line using `-Djava.security.policy`.

ProxyHostPropertyName

```
public static final java.lang.String ProxyHostPropertyName
```

Configuration property which names the RMI proxy host. This is `rmi.proxyHost`

com.rmiproxy
FirewallException

Syntax
```
public class FirewallException extends java.rmi.RemoteException

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--java.rmi.RemoteException
                          |
                          +--com.rmiproxy.FirewallException
```

All Implemented Interfaces: java.io.Serializable

Description
General firewall exception class.

Member Summary	
Constructors	
public	FirewallException() xiii Construct a FirewallException
public	FirewallException(String) xiii Construct a FirewallException with the specified message.
public	FirewallException(String, Exception) xiv Construct a FirewallException with the specified message.

FirewallException()

```
public FirewallException()
```

Construct a FirewallException

FirewallException(String)

```
public FirewallException(java.lang.String msg)
```

Construct a FirewallException with the specified message.

Parameters:
msg - Message

FirewallException(String, Exception)

```
public FirewallException(java.lang.String msg, java.lang.Exception detail)
```

Construct a FirewallException with the specified message.

Parameters:

`msg` - Message

`detail` - Nested exception

com.rmiproxy.registry

LocateProxyRegistry

Syntax

```
public final class LocateProxyRegistry

java.lang.Object
  |
  +--com.rmiproxy.registry.LocateProxyRegistry
```

Description

`LocateRegistry` is used to obtain a reference to a bootstrap remote object registry on a particular host (including the local host), or to create a remote object registry that accepts calls on a specific port.

Note that a `getRegistry` call does not actually make a connection to the remote host. It simply creates a local reference to the remote registry and will succeed even if no registry is running on the remote host. Therefore, a subsequent method invocation to a remote registry returned as a result of this method may fail.

Since: JDK1.1

See Also: `Modified EJP 15/9/2000 for ProxyNaming. This does make a connection to the remote`

Member Summary

Methods

`public static Registry`	`createRegistry(int)` [xvi] Creates and exports a `Registry` on the local host that accepts requests on the specified `port`.
`public static Registry`	`createRegistry(int, RMIClientSocketFactory, RMIServerSocketFactory)` [xvi] Creates and exports a `Registry` on the local host that uses custom socket factories for communication with that registry.
`public static ProxyRegistry`	`getRegistry()` [xvi] Returns a reference to the the remote object `Registry` for the local host on the default registry port of 1099.
`public static ProxyRegistry`	`getRegistry(int)` [xvii] Returns a reference to the the remote object `Registry` for the local host on the specified `port`.
`public static ProxyRegistry`	`getRegistry(String)` [xvii] Returns a reference to the remote object `Registry` on the specified `host` on the default registry port of 1099.
`public static ProxyRegistry`	`getRegistry(String, int)` [xvii] Returns a reference to the remote object `Registry` on the specified `host` and `port`.

Member Summary	
public static ProxyRegistry	getRegistry(String, int, RMIClientSocketFactory)[xviii] Returns a locally created remote reference to the remote object Registry on the specified host and port.

createRegistry(int)

```
public static java.rmi.registry.Registry createRegistry(int port)
```

Creates and exports a Registry on the local host that accepts requests on the specified port.

Parameters:
port - the port on which the registry accepts requests

Returns: the registry

Throws:
RemoteException - if the registry could not be exported

Since: JDK1.1

createRegistry(int, RMIClientSocketFactory, RMIServerSocketFactory)

```
public static java.rmi.registry.Registry createRegistry(int port,
            java.rmi.server.RMIClientSocketFactory csf,
            java.rmi.server.RMIServerSocketFactory ssf)
```

Creates and exports a Registry on the local host that uses custom socket factories for communication with that registry. The registry that is created listens for incoming requests on the given port using a ServerSocket created from the supplied RMIServerSocketFactory. A client that receives a reference to this registry will use a Socket created from the supplied RMIClientSocketFactory.

Parameters:
port - port on which the registry accepts requests

csf - client-side Socket factory used to make connections to the registry

ssf - server-side ServerSocket factory used to accept connections to the registry

Returns: the registry

Throws:
RemoteException - if the registry could not be exported

Since: 1.2

getRegistry()

```
public static ProxyRegistry[xxiii] getRegistry()
```

Returns a reference to the the remote object Registry for the local host on the default registry port of 1099.

Returns: reference (a stub) to the remote object registry

Throws:
RemoteException - if the reference could not be created

`NotBoundException`

Since: JDK1.1

---

getRegistry(int)

`public static ProxyRegistry`[xxiii] **`getRegistry`**`(int port)`

Returns a reference to the the remote object `Registry` for the local host on the specified `port`.

Parameters:
`port` - port on which the registry accepts requests

Returns: reference (a stub) to the remote object registry

Throws:
`RemoteException` - if the reference could not be created

`NotBoundException`

Since: JDK1.1

---

getRegistry(String)

`public static ProxyRegistry`[xxiii] **`getRegistry`**`(java.lang.String host)`

Returns a reference to the remote object `Registry` on the specified `host` on the default registry port of 1099. If `host` is `null`, the local host is used.

Parameters:
`host` - host for the remote registry

Returns: reference (a stub) to the remote object registry

Throws:
`RemoteException` - if the reference could not be created

`NotBoundException`

Since: JDK1.1

---

getRegistry(String, int)

`public static ProxyRegistry`[xxiii] **`getRegistry`**`(java.lang.String host, int port)`

Returns a reference to the remote object `Registry` on the specified `host` and `port`. If `host` is `null`, the local host is used.

Parameters:
`host` - host for the remote registry

`port` - port on which the registry accepts requests

Returns: reference (a stub) to the remote object registry

Throws:
`RemoteException` - if the reference could not be created

`NotBoundException`

Since: JDK1.1

getRegistry(String, int, RMIClientSocketFactory)

```
public static ProxyRegistry[xxiii] getRegistry(java.lang.String host, int port,
            java.rmi.server.RMIClientSocketFactory csf)
```

Returns a locally created remote reference to the remote object `Registry` on the specified `host` and `port`. Communication with this remote registry will use the supplied `RMIClientSocketFactory` `csf` to create `Socket` connections to the registry on the remote `host` and `port`.

Parameters:

`host` - host for the remote registry

`port` - port on which the registry accepts requests

`csf` - client-side `Socket` factory used to make connections to the registry. If `csf` is null, then the default client-side `Socket` factory will be used in the registry stub.

Returns: reference (a stub) to the remote registry

Throws:

`RemoteException` - if the reference could not be created

`NotBoundException`

Since: 1.2

com.rmiproxy
ProxyNaming

Syntax

```
public final class ProxyNaming

java.lang.Object
  |
  +--com.rmiproxy.ProxyNaming
```

Description

The main access point to the RMI application firewall. `ProxyNaming` is the only interface clients and servers need to use.

For RMI **servers**, the RMI Proxy provides:

- the ability to **place RMI servers behind firewalls**
- **access control** to the remote service, by
- hostname/IP address
- bound-name
- remote interfaces and methods

For RMI **clients**, the RMI Proxy supercedes the inferior HTTP tunnelling technique, providing:

- the same ability to place clients behind firewalls
- much improved performance and reliability over HTTP tunnelling
- the ability to **access RMI servers behind different firewalls**
- **access control** to the remote service, by
- hostname/IP address
- bound-name
- remote interfaces and methods

The API implemented by `ProxyNaming` is identical to that of `java.rmi.Naming`.

When an RMI **server** behind a firewall is bound to a registry, `ProxyNaming.bind` or `rebind` should be used instead of `java.rmi.Naming.bind` or `rebind`. This propagates the binding to the server JVM's RMI Proxy Registry. This proxy in turn may have another, more outer, proxy, specified by the value of its `rmi.proxyHost` property; if so, the propagation continues all the way to the outermost one at the final firewall (i.e. the one which has no further proxy). As a result, the server binding is now visible at the outermost RMI Proxy (usually, to the Internet), and can be accessed by clients, subject to the access control policies of the RMI Proxies. The `bind/rebind` action itself is also subject to the access control policy of each proxy, so as to control the security of binding to remote registries - this is not supported by `java.rmi.Naming`.

When an RMI **client** behind a firewall looks up an RMI registry URL, it should use `ProxyNaming.lookup` instead of `java.rmi.Naming.lookup`. This action delegates the lookup via all the client-side proxies, specified by the value of their `rmi.proxyHost` property, to the proxy at the final firewall (i.e. the one which has no further proxy - usually, the one at the junction with the Internet). Network administrators willing, this proxy is able to 'see' the server's outermost proxy through both the remaining firewalls, and so a binding for the name can be found.

The `rmi.proxyHost` property is an RMI URL which names the RMI Proxy host and port for the current JVM (similar to `http.proxyHost` and `http.proxyPort`). It is always the RMI Proxy which is closer to the Internet than this host.

The `rmi.nonProxyHosts` property is a list of hostnames that `ProxyNaming` should contact directly, not via the proxy. If the host being contacted appears in `rmi.nonProxyHosts`, or if no value for `rmi.proxyHost` is set, `ProxyNaming` behaves identically to `java.rmi.Naming`.

Notes:

- The RMI URL to be looked up by an RMI client must contain the hostname/port number of the outermost server-side RMI proxy, not the hostname/port number of the server itself.
- If the server is not behind a firewall, it need not use `ProxyNaming`, regardless of where the client is.
- If the client is not behind a firewall, it need not use `ProxyNaming`, regardless of where the server is.
- Any client of an RMI server, whether accessed directly or via an RMI Proxy, must be prepared to deal with a `NoSuchObjectException` arising out of a premature unexport due to a false DGC report. This is not a new issue in RMI design: it can occur whenever "network partitions exist between client and server" (*RMI Specification*, §3), because these make DGC unreliable. It is *likely* to occur in configurations involving RMI Proxies because such configurations almost always contain network partitions.
- For security and administrative reasons, the RMI Proxy runs on a single TCP/IP port, the standard RMI Registry port (1099). This port must be opened by the firewall administrator for access by the RMI Proxy.

Member Summary

Methods

`public static void`	`bind(String, Remote)` xx Binds a name to a remote object.
`public static String`	`list(String)` xxi Lists the contents of a remote registry.
`public static Remote`	`lookup(String)` xxi Looks up a remote item in a registry.
`public static void`	`rebind(String, Remote)` xxi Rebinds a name to a new remote object, overriding any existing binding.
`public static void`	`unbind(String)` xxii Unbinds a name from any association with a remote object.

bind(String, Remote)

```
public static void bind(java.lang.String name, java.rmi.Remote object)
```

Binds a name to a remote object.

Parameters:

`name` - RMI URL of the item: `rmi://host[:port]/item`

`object` - reference to the remote object to bind

Throws:

`RemoteException` - remote error

`AccessException` - permission denied by RMI or firewall

`AlreadyBoundException` - name already bound

`MalformedURLException` - bad item URL

list(String)

```
public static java.lang.String[] list(java.lang.String name)
```

Lists the contents of a remote registry.

Parameters:

`name` - RMI URL of the registry: `rmi://host[:port]`. Note that if the server is behind a firewall, the client must provide the hostname/port number of the outermost server-side RMI proxy, not the hostname/port number of the server itself.

Returns: An array (possibly 0 elements) of URL-formatted names bound in the registry.

Throws:

`RemoteException` - remote error

`AccessException` - permission denied by RMI or firewall

`MalformedURLException` - bad registry URL

lookup(String)

```
public static java.rmi.Remote lookup(java.lang.String name)
```

Looks up a remote item in a registry.

Parameters:

`name` - RMI URL of the item: `rmi://host[:port]/item`. Note that if the server is behind a firewall, the client must provide the hostname/port number of the outermost server-side RMI proxy, not the hostname/port number of the server itself.

Returns: A stub to the remote object bound to `name`

Throws:

`RemoteException` - remote error

`AccessException` - permission denied by RMI or firewall

`NotBoundException` - name not bound

`MalformedURLException` - bad item URL

rebind(String, Remote)

```
public static void rebind(java.lang.String name, java.rmi.Remote object)
```

Rebinds a name to a new remote object, overriding any existing binding.

Parameters:

`name` - RMI URL of the item: `rmi://host[:port]/item`

`object` - reference to the remote object to bind

Throws:

`RemoteException` - remote error

`AccessException` - permission denied by RMI or firewall

`MalformedURLException` - bad item URL

unbind(String)

```
public static void unbind(java.lang.String name)
```

Unbinds a name from any association with a remote object.

Parameters:

`name` - RMI URL of the item: `rmi://host[:port]/item`

Throws:

`RemoteException` - remote error

`AccessException` - permission denied by RMI or firewall

`NotBoundException` - name not bound

`MalformedURLException` - bad item URL

com.rmiproxy.registry

ProxyRegistry

Syntax

```
public interface ProxyRegistry extends java.rmi.registry.Registry
```

All Superinterfaces: java.rmi.registry.Registry,java.rmi.Remote

All Known Implementing Classes: ProxyRegistry.AbstractServer[xxvii]

Description

ProxyRegistry defines the remote interface to the proxy server. The interface is nothing more than the Registry interface. Consider that a client can only obtain an initial RMI reference via a naming service, which may have to be accessed via one or more proxies. Any RMI reference returned by the proxy, including this one, is swizzled into a reference to the proxy. Any RMI reference passed to the proxy by the client is similarly swizzled into a reference to the proxy (for callbacks). Now consider a server; it binds itself to the local Registry. If these are executing behind a firewall there will be one or more proxies intervening, so the bind is exported to the proxy nearest. This goes through all the same processes as above, with the result that the server now appears to be running in the proxy. If the server returns a reference to another remote object, that object also gets proxied, etcetera ad infinitum.

Member Summary

Inner Classes

ProxyRegistry.AbstractServer[xxvii]	Abstract implementation class, for rmic

Methods

public void	bind(String, Remote)[xxiii] Bind
public String	list(String)[xxiv] List the bindings in a registry
public Remote	lookup(String)[xxiv] Lookup a binding in a registry
public void	rebind(String, Remote)[xxiv] Rebind
public void	unbind(String)[xxv] Unbind

---

bind(String, Remote)

```
public void bind(java.lang.String url, java.rmi.Remote object)
```

list(String)

Bind

Overrides: `bind(String, Remote)` xxiii in interface `ProxyRegistry` xxiii

Parameters:

`url` - Name to bind

`object` - Object to bind

Throws:

`AlreadyBoundException` - name is already bound

`AccessException` - no permission to bind

`RemoteException` - on any remote error

---

list(String)

```
public java.lang.String[] list(java.lang.String url)
```

List the bindings in a registry

Parameters:

`url` - URL of registry

Throws:

`MalformedURLException` - bad URL

`RemoteException` - I/O error

`AccessException` - permission denied

---

lookup(String)

```
public java.rmi.Remote lookup(java.lang.String url)
```

Lookup a binding in a registry

Overrides: `lookup(String)` xxiv in interface `ProxyRegistry` xxiii

Parameters:

`url` - URL to look up

Throws:

`NotBoundException` - name not bound

`RemoteException` - I/O error

`AccessException` - permission denied

---

rebind(String, Remote)

```
public void rebind(java.lang.String url, java.rmi.Remote object)
```

Rebind

Overrides: `rebind(String, Remote)` xxiv in interface `ProxyRegistry` xxiii

Parameters:

`url` - Name to bind

`object` - Object to bind

Throws:

`AccessException` - no permission to rebind

`RemoteException` - on any remote error

---

unbind(String)

```
public void unbind(java.lang.String url)
```

Unbind

Overrides: `unbind(String)` xxv in interface `ProxyRegistry` xxiii

Parameters:

`url` - URL to unbind

Throws:

`NotBoundException` - name is not bound

`AccessException` - no permission to unbind

`RemoteException` - on any remote error

unbind(String)

com.rmiproxy.registry

ProxyRegistry.AbstractServer

Syntax

```
public abstract static class ProxyRegistry.AbstractServer extends
    java.rmi.server.UnicastRemoteObject implements ProxyRegistry[xxiii]

java.lang.Object
  |
  +--java.rmi.server.RemoteObject
        |
        +--java.rmi.server.RemoteServer
              |
              +--java.rmi.server.UnicastRemoteObject
                    |
                    +--com.rmiproxy.registry.ProxyRegistry.AbstractServer
```

All Implemented Interfaces: ProxyRegistry[xxiii],java.rmi.registry.Registry,java.rmi.Remote,java.io.Serializable

Description

Abstract implementation class, for rmic

Member Summary	
Constructors	
public	ProxyRegistry.AbstractServer(int)[xxvii]
public	ProxyRegistry.AbstractServer(int, RMIClientSocketFactory, RMIServerSocketFactory)[xxvii]

ProxyRegistry.AbstractServer(int)

```
public ProxyRegistry.AbstractServer(int port)
```

Throws:

RemoteException

ProxyRegistry.AbstractServer(int, RMIClientSocketFactory, RMIServerSocketFactory)

```
public ProxyRegistry.AbstractServer(int port,
            java.rmi.server.RMIClientSocketFactory csf,
            java.rmi.server.RMIServerSocketFactory ssf)
```

Throws:

RemoteException

ProxyRegistry.AbstractServer(int, RMIClientSocketFactory, RMIServerSocketFactory)

com.rmiproxy.jndi

RegistryContext

Syntax

```
public class RegistryContext implements javax.naming.Context,javax.naming.Referenceable

java.lang.Object
  |
  +--com.rmiproxy.jndi.RegistryContext
```

All Implemented Interfaces: javax.naming.Context,javax.naming.Referenceable

Description

A RegistryContext is a context representing a remote RMI registry.

Member Summary

Fields

public static final	SECURITY_MGR[xxxv]

Constructors

public	RegistryContext(String, int, Hashtable)[xxxiv]
	Returns a context for the registry at a given host and port.

Methods

public Object	addToEnvironment(String, Object)[xxx]
public void	bind(Name, Object)[xxx]
	If the object to be bound is both Remote and Referenceable, binds the object itself, not its Reference.
public void	bind(String, Object)[xxx]
public void	close()[xxx]
public Name	composeName(Name, Name)[xxxi]
public String	composeName(String, String)[xxxi]
public Context	createSubcontext(Name)[xxxi]
public Context	createSubcontext(String)[xxxi]
public void	destroySubcontext(Name)[xxxi]
public void	destroySubcontext(String)[xxxi]
protected void	finalize()[xxxii]
public Hashtable	getEnvironment()[xxxii]
public String	getNameInNamespace()[xxxii]
public NameParser	getNameParser(Name)[xxxii]
public NameParser	getNameParser(String)[xxxii]
public Reference	getReference()[xxxii]
	Returns an RMI registry reference for this context.
public NamingEnumeration	list(Name)[xxxiii]
public NamingEnumeration	list(String)[xxxiii]

Member Summary

public NamingEnumeration	listBindings(Name) xxxiii
public NamingEnumeration	listBindings(String) xxxiii
public Object	lookup(Name) xxxiii
public Object	lookup(String) xxxiii
public Object	lookupLink(Name) xxxiv
public Object	lookupLink(String) xxxiv
public void	rebind(Name, Object) xxxiv
public void	rebind(String, Object) xxxiv
public Object	removeFromEnvironment(String) xxxiv
public void	rename(Name, Name) xxxv Rename is implemented by this sequence of operations: lookup, bind, unbind.
public void	rename(String, String) xxxv
public void	unbind(Name) xxxv
public void	unbind(String) xxxv
public static NamingException	wrapRemoteException(RemoteException) xxxv Wrap a RemoteException inside a NamingException.

addToEnvironment(String, Object)

```
public java.lang.Object addToEnvironment(java.lang.String propName,
            java.lang.Object propVal)
```

Specified By: addToEnvironment(String, Object) xxx in interface RegistryContext xxix

Throws:
NamingException

bind(Name, Object)

```
public void bind(javax.naming.Name name, java.lang.Object obj)
```

If the object to be bound is both Remote and Referenceable, binds the object itself, not its Reference.

Specified By: bind(Name, Object) xxx in interface RegistryContext xxix

Throws:
NamingException

bind(String, Object)

```
public void bind(java.lang.String name, java.lang.Object obj)
```

Specified By: bind(String, Object) xxx in interface RegistryContext xxix

Throws:
NamingException

close()

```
public void close()
```

Specified By: close()[xxx] in interface RegistryContext[xxix]

---

composeName(Name, Name)

```
public javax.naming.Name composeName(javax.naming.Name name, javax.naming.Name prefix)
```

Specified By: composeName(Name, Name)[xxxi] in interface RegistryContext[xxix]

Throws:
NamingException

---

composeName(String, String)

```
public java.lang.String composeName(java.lang.String name, java.lang.String prefix)
```

Specified By: composeName(String, String)[xxxi] in interface RegistryContext[xxix]

Throws:
NamingException

---

createSubcontext(Name)

```
public javax.naming.Context createSubcontext(javax.naming.Name name)
```

Specified By: createSubcontext(Name)[xxxi] in interface RegistryContext[xxix]

Throws:
NamingException

---

createSubcontext(String)

```
public javax.naming.Context createSubcontext(java.lang.String name)
```

Specified By: createSubcontext(String)[xxxi] in interface RegistryContext[xxix]

Throws:
NamingException

---

destroySubcontext(Name)

```
public void destroySubcontext(javax.naming.Name name)
```

Specified By: destroySubcontext(Name)[xxxi] in interface RegistryContext[xxix]

Throws:
NamingException

---

destroySubcontext(String)

```
public void destroySubcontext(java.lang.String name)
```

Specified By: destroySubcontext(String)[xxxi] in interface RegistryContext[xxix]

Throws:

finalize()

NamingException

---

finalize()

```
protected void finalize()
```

Overrides: java.lang.Object.finalize() in class java.lang.Object

---

getEnvironment()

```
public java.util.Hashtable getEnvironment()
```

Specified By: getEnvironment()[xxxii] in interface RegistryContext[xxix]

Throws:
NamingException

---

getNameInNamespace()

```
public java.lang.String getNameInNamespace()
```

Specified By: getNameInNamespace()[xxxii] in interface RegistryContext[xxix]

---

getNameParser(Name)

```
public javax.naming.NameParser getNameParser(javax.naming.Name name)
```

Specified By: getNameParser(Name)[xxxii] in interface RegistryContext[xxix]

Throws:
NamingException

---

getNameParser(String)

```
public javax.naming.NameParser getNameParser(java.lang.String name)
```

Specified By: getNameParser(String)[xxxii] in interface RegistryContext[xxix]

Throws:
NamingException

---

getReference()

```
public javax.naming.Reference getReference()
```

Returns an RMI registry reference for this context.

If this context was created from a reference, that reference is returned. Otherwise, an exception is thrown if the registry's host is "localhost" or the default (null). Although this could possibly make for a valid reference, it's far more likely to be an easily made error.

Specified By: getReference()[xxxii] in interface RegistryContext[xxix]

Throws:
NamingException

See Also: RegistryContextFactory[xxxvii]

---

list(Name)

```
public javax.naming.NamingEnumeration list(javax.naming.Name name)
```

Specified By: list(Name)[xxxiii] in interface RegistryContext[xxix]

Throws:
NamingException

---

list(String)

```
public javax.naming.NamingEnumeration list(java.lang.String name)
```

Specified By: list(String)[xxxiii] in interface RegistryContext[xxix]

Throws:
NamingException

---

listBindings(Name)

```
public javax.naming.NamingEnumeration listBindings(javax.naming.Name name)
```

Specified By: listBindings(Name)[xxxiii] in interface RegistryContext[xxix]

Throws:
NamingException

---

listBindings(String)

```
public javax.naming.NamingEnumeration listBindings(java.lang.String name)
```

Specified By: listBindings(String)[xxxiii] in interface RegistryContext[xxix]

Throws:
NamingException

---

lookup(Name)

```
public java.lang.Object lookup(javax.naming.Name name)
```

Specified By: lookup(Name)[xxxiii] in interface RegistryContext[xxix]

Throws:
NamingException

---

lookup(String)

```
public java.lang.Object lookup(java.lang.String name)
```

Specified By: lookup(String)[xxxiii] in interface RegistryContext[xxix]

Throws:
NamingException

lookupLink(Name)

```
public java.lang.Object lookupLink(javax.naming.Name name)
```

Specified By: lookupLink(Name)[xxxiv] in interface RegistryContext[xxix]

Throws:
NamingException

lookupLink(String)

```
public java.lang.Object lookupLink(java.lang.String name)
```

Specified By: lookupLink(String)[xxxiv] in interface RegistryContext[xxix]

Throws:
NamingException

rebind(Name, Object)

```
public void rebind(javax.naming.Name name, java.lang.Object obj)
```

Specified By: rebind(Name, Object)[xxxiv] in interface RegistryContext[xxix]

Throws:
NamingException

rebind(String, Object)

```
public void rebind(java.lang.String name, java.lang.Object obj)
```

Specified By: rebind(String, Object)[xxxiv] in interface RegistryContext[xxix]

Throws:
NamingException

RegistryContext(String, int, Hashtable)

```
public RegistryContext(java.lang.String host, int port, java.util.Hashtable env)
```

Returns a context for the registry at a given host and port. If “host” is null, uses default host. If “port” is non-positive, uses default port. Cloning of “env” is handled by caller; see comments within RegistryContextFactory.getObjectInstance(), for example.

Throws:
NamingException

removeFromEnvironment(String)

```
public java.lang.Object removeFromEnvironment(java.lang.String propName)
```

Specified By: removeFromEnvironment(String)[xxxiv] in interface RegistryContext[xxix]

Throws:
NamingException

rename(Name, Name)

```
public void rename(javax.naming.Name oldName, javax.naming.Name newName)
```

Rename is implemented by this sequence of operations: lookup, bind, unbind. The sequence is not performed atomically.

Specified By: rename(Name, Name)[xxxv] in interface RegistryContext[xxix]

Throws:
NamingException

rename(String, String)

```
public void rename(java.lang.String name, java.lang.String newName)
```

Specified By: rename(String, String)[xxxv] in interface RegistryContext[xxix]

Throws:
NamingException

SECURITY_MGR

```
public static final java.lang.String SECURITY_MGR
```

unbind(Name)

```
public void unbind(javax.naming.Name name)
```

Specified By: unbind(Name)[xxxv] in interface RegistryContext[xxix]

Throws:
NamingException

unbind(String)

```
public void unbind(java.lang.String name)
```

Specified By: unbind(String)[xxxv] in interface RegistryContext[xxix]

Throws:
NamingException

wrapRemoteException(RemoteException)

```
public static javax.naming.NamingException wrapRemoteException(java.rmi.RemoteException
            re)
```

Wrap a RemoteException inside a NamingException.

wrapRemoteException(RemoteException)

com.rmiproxy.jndi

RegistryContextFactory

Syntax

```
public class RegistryContextFactory implements
    javax.naming.spi.ObjectFactory,javax.naming.spi.InitialContextFactory

java.lang.Object
  |
  +--com.rmiproxy.jndi.RegistryContextFactory
```

All Implemented Interfaces: javax.naming.spi.InitialContextFactory,javax.naming.spi.ObjectFactory

Description

A RegistryContextFactory takes an RMI registry reference, and creates the corresponding RMI object or registry context. In addition, it serves as the initial context factory when using an RMI registry as an initial context.

When an initial context is being created, the environment property "java.naming.provider.url" should contain the RMI URL of the appropriate registry. Otherwise, the default URL "rmi:" is used.

An RMI registry reference contains one or more StringRefAddrs of type "URL", each containing a single RMI URL. Other addresses are ignored. Multiple URLs represent alternative addresses for the same logical resource. The order of the addresses is not significant.

Member Summary

Fields

public static final	ADDRESS_TYPE[xxxvii]

The type of each address in an RMI registry reference.

Constructors

public	RegistryContextFactory()[xxxviii]

Methods

public Context	getInitialContext(Hashtable)[xxxvii]
public Object	getObjectInstance(Object, Name, Context, Hashtable)[xxxviii]

ADDRESS_TYPE

```
public static final java.lang.String ADDRESS_TYPE
```

The type of each address in an RMI registry reference.

---

getInitialContext(Hashtable)

```
public javax.naming.Context getInitialContext(java.util.Hashtable env)
```

Specified By: getInitialContext(Hashtable)[xxxvii] in interface RegistryContextFactory[xxxvii]

getObjectInstance(Object, Name, Context, Hashtable)

Throws:
NamingException

---

getObjectInstance(Object, Name, Context, Hashtable)

```
public java.lang.Object getObjectInstance(java.lang.Object ref, javax.naming.Name name,
            javax.naming.Context nameCtx, java.util.Hashtable env)
```

Specified By: getObjectInstance(Object, Name, Context, Hashtable)[xxxviii] in interface RegistryContextFactory[xxxvii]

Throws:
NamingException

RegistryContextFactory()

```
public RegistryContextFactory()
```

com.rmiproxy
Version

Syntax
```
public interface Version
```

Description
Copyright notice plus automatic revision & date tracker.

Member Summary	
Fields	
public static final	Copyright_{xxxix}
public static final	Date_{xxxix}
public static final	Number_{xxxix}
public static final	Release_{xxxix}
public static final	Title_{xxxix}
public static final	Version_{xxxix}

Copyright
```
public static final java.lang.String Copyright
```

Date
```
public static final java.lang.String Date
```

Number
```
public static final java.lang.String Number
```

Release
```
public static final java.lang.String Release
```

Title
```
public static final java.lang.String Title
```

Version
```
public static final java.lang.String Version
```

Version

Almanac

Configuration — **com.rmiproxy**

```
Object
   ➥Configuration
```

✍■	String **ConfigFile**
❑	Configuration **current**()
❍	String **getNonProxyHosts**()
❍	String **getProxyHost**()
❍	boolean **isNonProxyHost**(String host)

Firewall — **com.rmiproxy**

```
Firewall
```

✍■	int **JRMP_PROXY_PORT**
✍■	String **NonProxyHostsPropertyName**
✍■	String **PolicyFileName**
✍■	String **ProxyHostPropertyName**

FirewallException — **com.rmiproxy**

```
Object
   ➥Throwable                    java.io.Serializable
      ➥Exception
         ➥java.io.IOException
            ➥java.rmi.RemoteException
               ➥FirewallException
```

❇	**FirewallException**()
❇	**FirewallException**(String msg)
❇	**FirewallException**(String msg, Exception detail)

LocateProxyRegistry — com.rmiproxy.registry

Object
➥LocateProxyRegistry

1.1	❑	java.rmi.regis- **createRegistry**(int port) try.Registry throws java.rmi.RemoteException
1.2	❑	java.rmi.regis- **createRegistry**(int port, try.Registry java.rmi.server.RMIClientSocketFact ory csf, java.rmi.server.RMIServerSocketFact ory ssf) throws java.rmi.RemoteException
1.1	❑	ProxyRegistry **getRegistry**() throws java.rmi.RemoteException, java.rmi.NotBoundException
1.1	❑	ProxyRegistry **getRegistry**(int port) throws java.rmi.RemoteException, java.rmi.NotBoundException
1.1	❑	ProxyRegistry **getRegistry**(String host) throws java.rmi.RemoteException, java.rmi.NotBoundException
1.1	❑	ProxyRegistry **getRegistry**(String host, int port) throws java.rmi.RemoteException, java.rmi.NotBoundException
1.2	❑	ProxyRegistry **getRegistry**(String host, int port, java.rmi.server.RMIClientSocketFact ory csf) throws java.rmi.RemoteException, java.rmi.NotBoundException

ProxyNaming — com.rmiproxy

Object
➥ProxyNaming

	❑	void **bind**(String name, java.rmi.Remote object) throws java.rmi.RemoteException, java.rmi.AccessException, java.rmi.AlreadyBoundException, java.net.MalformedURLException
	❑	String[] **list**(String name) throws java.rmi.RemoteException, java.rmi.AccessException, java.net.MalformedURLException

```
❑    java.rmi.Remote lookup(String name)
                        throws java.rmi.RemoteException,
                        java.rmi.AccessException,
                        java.rmi.NotBoundException,
                        java.net.MalformedURLException
❑               void rebind(String name,
                        java.rmi.Remote object)
                        throws java.rmi.RemoteException,
                        java.rmi.AccessException,
                        java.net.MalformedURLException
❑               void unbind(String name)
                        throws java.rmi.RemoteException,
                        java.rmi.AccessException,
                        java.rmi.NotBoundException,
                        java.net.MalformedURLException
```

ProxyRegistry — com.rmiproxy.registry

```
ProxyRegistry                       java.rmi.registry.Registry
                void bind(String url,
                        java.rmi.Remote object)
                        throws java.rmi.RemoteException,
                        java.rmi.AlreadyBoundException,
                        java.rmi.AccessException
          String[] list(String url)
                        throws java.rmi.RemoteException,
                        java.net.MalformedURLException,
                        java.rmi.AccessException
    java.rmi.Remote lookup(String url)
                        throws java.rmi.RemoteException,
                        java.rmi.NotBoundException,
                        java.rmi.AccessException
                void rebind(String url,
                        java.rmi.Remote object)
                        throws java.rmi.RemoteException,
                        java.rmi.AccessException
                void unbind(String url)
                        throws java.rmi.RemoteException,
                        java.rmi.NotBoundException,
                        java.rmi.AccessException
```

ProxyRegistry.Abstract-Server — com.rmiproxy.registry

```
Object
   ➥java.rmi.server.RemoteObject java.rmi.Remote, java.io.Serializable
      ➥java.rmi.server.RemoteServer
         ➥java.rmi.server.UnicastRemoteObject
            ➥ProxyRegistry.AbstractServerProxyRegistry
❊                   ProxyRegistry.AbstractServer(int port)
                        throws java.rmi.RemoteException
❊                   ProxyRegistry.AbstractServer(int port,
                        java.rmi.server.RMIClientSocketFact
                        ory csf,
                        java.rmi.server.RMIServerSocketFact
                        ory ssf)
                        throws java.rmi.RemoteException
```

RegistryContext — com.rmiproxy.jndi

Object
➥RegistryContext javax.naming.Context, javax.naming.Referenceable

	Object	**addToEnvironment**(String propName, Object propVal) throws javax.naming.NamingException
	void	**bind**(javax.naming.Name name, Object obj) throws javax.naming.NamingException
	void	**bind**(String name, Object obj) throws javax.naming.NamingException
	void	**close**()
	javax.naming.Name	**composeName**(javax.naming.Name name, javax.naming.Name prefix) throws javax.naming.NamingException
	String	**composeName**(String name, String prefix) throws javax.naming.NamingException
	javax.naming.Context	**createSubcontext**(javax.naming.Name name) throws javax.naming.NamingException
	javax.naming.Context	**createSubcontext**(String name) throws javax.naming.NamingException
	void	**destroySubcontext**(javax.naming.Name name) throws javax.naming.NamingException
	void	**destroySubcontext**(String name) throws javax.naming.NamingException
♦	void	**finalize**()
	java.util.Hashtable	**getEnvironment**() throws javax.naming.NamingException
	String	**getNameInNamespace**()
	javax.naming.NameParser	**getNameParser**(javax.naming.Name name) throws javax.naming.NamingException
	javax.naming.NameParser	**getNameParser**(String name) throws javax.naming.NamingException
	javax.naming.Reference	**getReference**() throws javax.naming.NamingException
	javax.naming.NamingEnumeration	**list**(javax.naming.Name name) throws javax.naming.NamingException
	javax.naming.NamingEnumeration	**list**(String name) throws javax.naming.NamingException
	javax.naming.NamingEnumeration	**listBindings**(javax.naming.Name name) throws javax.naming.NamingException
	javax.naming.NamingEnumeration	**listBindings**(String name) throws javax.naming.NamingException
	Object	**lookup**(javax.naming.Name name) throws javax.naming.NamingException
	Object	**lookup**(String name) throws javax.naming.NamingException
	Object	**lookupLink**(javax.naming.Name name) throws javax.naming.NamingException

		Object **lookupLink**(String name) throws javax.naming.NamingException
		void **rebind**(javax.naming.Name name, Object obj) throws javax.naming.NamingException
		void **rebind**(String name, Object obj) throws javax.naming.NamingException
❇		**RegistryContext**(String host, int port, java.util.Hashtable env) throws javax.naming.NamingException
		Object **removeFromEnvironment**(String propName) throws javax.naming.NamingException
		void **rename**(javax.naming.Name oldName, javax.naming.Name newName) throws javax.naming.NamingException
		void **rename**(String name, String newName) throws javax.naming.NamingException
✍■		String **SECURITY_MGR**
		void **unbind**(javax.naming.Name name) throws javax.naming.NamingException
		void **unbind**(String name) throws javax.naming.NamingException
❑	javax.naming.NamingException	**wrapRemoteException**(java.rmi.RemoteException re)

RegistryContextFactory — com.rmiproxy.jndi

Object
➥RegistryContextFactory — javax.naming.spi.ObjectFactory, javax.naming.spi.InitialContextFactory

✍■		String **ADDRESS_TYPE**
	javax.naming.Context	**getInitialContext**(java.util.Hashtable env) throws javax.naming.NamingException
		Object **getObjectInstance**(Object ref, javax.naming.Name name, javax.naming.Context nameCtx, java.util.Hashtable env) throws javax.naming.NamingException
❇		**RegistryContextFactory**()

Version — com.rmiproxy

Version

✍■		String **Copyright**
✍■		String **Date**
✍■		String **Number**
✍■		String **Release**
✍■		String **Title**
✍■		String **Version**

Index

A

B

C

D

F

G

I

J

L

N

P

R

S

T

U

V

W