

RMI-Syntax:

Server:

```
import java.rmi.RMISecurityManager;
import java.rmi.RemoteException;
import java.rmi.Naming;
import java.rmi.server.UnicastRemoteObject;
import java.net.InetAddress;
import java.net.UnknownHostException;
```

```
public class RMIServer extends UnicastRemoteObject implements RMIInterface
{
```

```
    public HelloServer() throws RemoteException {
        super();
    }
```

```
    public void RemoteMethod () {
        System.out.println("Received request!");
        .....
    }
```

```
    public static void main(String args[]) {
        String registryServer="";
        // use a Security Manager
        if (System.getSecurityManager() == null) {
            System.setSecurityManager(new RMISecurityManager());
        }
        if (args.length == 1) {
            // get registryServer from command line arg
            registryServer = args[0];
        }
        else {
            try {
                // use local host as registryServer
                System.out.println("Using local host as registryServer.");
                registryServer = InetAddress.getLocalHost().getHostName();
            }
            catch (java.net.UnknownHostException ex) {
                System.out.println("Failed to determine name of local host.");
                System.exit(1);
            }
        }
        try {
            // create instance of RMIServer
            RMIServer obj = new RMIServer();
            // rebind obj in RMIRegistry on registryServer under name "RMI-Server"
            Naming.rebind("//" + registryServer + "/" + "RMI-Server", obj);
            System.out.println("RMI-Server bound in registry on " +
                registryServer);
        } catch (Exception e) {
            System.out.println("RMIServer err: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

Client:

```
import java.rmi.RMISecurityManager;
import java.rmi.RemoteException;
import java.rmi.Naming;
import java.net.InetAddress;
import java.net.UnknownHostException;
```

```
public class RMIClient {
```

```
    public static void main(String args[]) {
```

```
        Security Manager genau so laden wie in RMIServer.
```

```
        RegistryServer genau so laden wie in RMIServer.
```

```
        // "obj" will contain a reference to the remote object
        RMIInterface obj = null;
```

```
        try {
```

```
            // perform lookup in rmiregistry on registryServer for remote
```

```
            // object "RMI-Server"
```

```
            System.out.println("lookup RMI-Server on: " + registryServer);
```

```
            // cast reference to remote interface RMIInterface
```

```
            obj = (RMIInterface)Naming.lookup("//" + registryServer + "/" + "RMI-Server");
```

```
            // call RemoteMethod() of remote object
```

```
            obj.RemoteMethod();
```

```
        } catch (Exception e) {
```

```
            System.out.println("RMIApplication exception: " + e.getMessage());
```

```
            e.printStackTrace();
```

```
        }
```

```
        .....RMIMethoden benutzung irgendwie
```

```
    }
```

```
}
```

ClientApplet:

```
import java.applet.*;
import java.awt.*;
import java.rmi.*;
import java.net.*;
```

```
public class RMIClientApplet extends Applet {
```

```
    private String webserver = "";
```

```
    // reference to remote interface RMIInterface
```

```
    private Hello hs = null;
```

```
    //INSERT YOUR PORT NUMBER
```

```
    private String portNumber = "1099";
```

```

public void init() {

    // get name of server where applet has been loaded from
    webserver = getDocumentBase().getHost();

    try {
        hs = (RMIIInterface)Naming.lookup("//" + webserver + ":" + portNumber +
            "/" + "RMI-Server");

        hs.RemoteMehtode();
    }
    catch (Exception e) {
        System.out.println("RMIClientApplet exception: " + e.getMessage());
        e.printStackTrace();
    }
}

public void paint(Graphics g) {

    if (hs == null) {
        g.drawString("Lookup for RMI-Server on " + webserver + " failed.",10,20);
        return;
    }

    g.drawString("Lookup for RMI-Server on " + webserver + "
        succeeded.",10,20);

    .....RMIMethoden benutzung irgendwie
}
}

```

Interface:

```

import java.rmi.Remote;
import java.rmi.RemoteException;

```

```

public interface RMIIInterface extends Remote {
    void RemoteMethod() throws RemoteException;
}

```

Compile

Server:

```

javac *.java
rmic RMIServer

```

```

jar -cvf RMIServer.jar *.class //jar file erzeugen muss nicht sein

```

client:

```

javac *.java

```

Execute

Server:

```

set HOST=hanibal.ifs.univie.ac.at
java -Djava.rmi.server.codebase=http://%HOST%/~a0026039/RMIServer.jar -
Djava.security.policy=policy RMIServer %HOST%:9090
pause

```

Client:

```

set HOST=hanibal.ifs.univie.ac.at java -Djava.security.policy=policy RMIClient
%HOST%:26058

```

RMI Callback - Syntax

RCAServer:

```
package server;
import interfaces.*;

import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.RMISecurityManager;
import java.rmi.server.UnicastRemoteObject;
import java.net.InetAddress;

/**
 * Simple RMI Callback server.
 * Implements remote method connect(), which receives a reference
 * to a remote object cbi of type CallbackInterface, and then calls
 * cbi.Callback().
 */

public class RCAServer extends UnicastRemoteObject implements
RCAServerInterface {

    public RCAServer() throws RemoteException {
        super();
    }

    public String connect(CallbackHandler cbh) {
        System.out.println("Received request");

        try {
            // execute remote method Callback()
            cbh.callback();
        }
        catch (Exception e) {
            System.out.println("Callback failed!");
            System.out.println("RCAServer err: " + e.getMessage());
            e.printStackTrace();
        }
        return "Hi from rca-server! " + count;
    }

    public static void main(String args[]) {
        Security Manager laden so wie in RMIServer

        try {
            RCAServer obj = new RCAServer();
            String webserver = InetAddress.getLocalHost().getHostName();
            Naming.rebind("//" + webserver + "/" + "rca-server", obj);
            System.out.println("rca- server bound in registry on " + webserver);
        }
        catch (Exception e) {
            System.out.println("RCAServer err: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

Interfaces

```
package interfaces;
```

```
import java.rmi.Remote;  
import java.rmi.RemoteException;
```

```
public interface RCAServerInterface extends Remote {  
    String connect(CallbackHandler cbh) throws RemoteException;  
}
```

```
package interfaces;
```

```
import java.rmi.Remote;  
import java.rmi.RemoteException;
```

```
public interface CallbackHandlerInterface extends Remote {  
    void callback() throws RemoteException;  
}
```

ClientApplet:

```
package applet;  
import interfaces.*;
```

```
import java.rmi.*;  
import java.rmi.server.UnicastRemoteObject;  
import java.awt.*;  
import java.applet.*;
```

```
class CallbackHandler extends UnicastRemoteObject  
    implements CallbackHandlerInterface {
```

```
    RMICallbackApplet applet;
```

```
public CallbackHandler(RMICallbackApplet applet) throws RemoteException {  
    super();  
    this.applet = applet;  
}
```

```
public void callback() {  
    applet.setSuccessMsg("Callback received !");  
    System.out.println("Callback received !");  
}  
}
```

```

package applet;
import interfaces.*;

import java.rmi.*;
import java.applet.*;
import java.awt.*;

public class RMICallbackApplet extends Applet {

    private RCAServer rcas = null;
    private CallbackHandlerImpl cbh = null;
    private String connectMsg = "";
    private String successMsg = "Callback failed!";
    private String webserver = "";

    public void setSuccessMsg(String msg) {
        successMsg = msg;
    }

    public String getWebserver() {
        return webserver;
    }

    public void init() {

        try {
            // CallbackHandler-Objekt erzeugen (remote object!)
            cbh = new CallbackHandler(this);
        }
        catch (Exception e) {
            System.out.println("Cannot instantiate CallbackHandler!");
        }

        try {
            // server bestimmen, von dem das Applet geladen wurde
            webserver = getDocumentBase().getHost();
            // lookup
            rcas = (RCAServer)Naming.lookup("//" + webserver + "/" + "rca-server");
        }
        catch (Exception e) {
            System.out.println("RMICallbackApplet exception: " + e.getMessage());
            e.printStackTrace();
        }

        try {
            // connect() am Server aufrufen; Server fuehrt callback() aus
            connectMsg = rcas.connect(cbh);
        }
        catch (Exception e) {
            System.out.println("RMICallbackApplet exception: " + e.getMessage());
            e.printStackTrace();
        }

    }
}

```

```

public void paint(Graphics g) {
    if (rcas == null) {
        g.drawString("Lookup for rca-server on " + webserver + "
                    failed.",10,20);
        return;
    }

    g.drawString("Lookup for rca-server on " + webserver + "
                succeeded.",10,20);
    g.drawString(successMsg,10,40);
    g.drawString("Remote method call on " + webserver + " returned: " +
                connectMsg,10,60);

}
}

```

Compile

Server:

```

javac server\*.java
rmic server.RCAServerImpl

```

Interfaces:

```

javac interfaces\*.java

```

Applet:

```

javac applet\*.java
rmic applet.CallbackHandler

```

Execute

```

java -Djava.rmi.server -Djava.security.policy=policy server.RCAServer

```

Hier brauchen wir keinen host bzw port angeben da der registryServer im Programm festgelegt wird

**RMICallback (CallbackHandlerImpl = CallbackHandler / RCAServerImpl = RCAServer /
bei Interfaces: CallbackHandlerInterface; RCAServerInterface)**

