

Java™ Remote Method Invocation API (Java RMI)

Overview

Java Remote Method Invocation (Java RMI) enables the programmer to create distributed Java technology-based to Java technology-based applications, in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. RMI uses object serialization to marshal and unmarshal parameters and does not truncate types, supporting true object-oriented polymorphism.

API Specification

- [java.rmi Package](#)
- [java.rmi.dgc Package](#)
- [java.rmi.registry Package](#)
- [java.rmi.server Package](#)
- [java.rmi.activation Package](#)

Architecture and Functional Specification

- [Java RMI Specification](#)

Tutorials

- [Getting Started](#)

The Getting Started Tutorial shows you the steps to follow to create a distributed version of the classic Hello World program using Java RMI. The Hello World applet makes a remote method call to the server from which it was downloaded to retrieve the message "Hello World!"

- [Using Custom Socket Factories with Java RMI](#)

The "Using Custom Socket Factories with Java RMI" tutorial shows you how to create a version of the distributed Hello World program in which the Java RMI runtime uses sockets of a type chosen by the programmer. This tutorial also includes a discussion of how Java RMI can be used over SSL sockets.

- [The Activation Tutorials](#)

The Activation Tutorials describe how to use the Java RMI APIs to implement, to register, and to use activatable objects. Each tutorial presents a different way to implement an activatable object. All tutorials use the same parameterized setup program that registers information about an activatable object with the Java RMI Activation System Daemon (rmid).

- [Configuring inetd to Launch rmid](#)

The Internet services daemon `inetd`, supported on the Solaris Operating System (Solaris OS), provides an alternative to starting up services at system boot time. This daemon, a server process for Internet standard services, can be configured to start services on demand.

- [Designing Services to be Launched from inetd](#)

This tutorial describes how to structure a service program (employing a specially exported local registry) so that the service can be started from `inetd` when clients connect to the service's local registry, and how to configure `inetd` to launch the service program.

- [Dynamic code downloading using Java RMI \(Using the `java.rmi.server.codebase` Property\)](#)

One of the most significant capabilities of the Java platform is the ability to dynamically download Java software from any URL to a VM running in a separate process, usually on a different physical system. The result is that a remote system can run a program, for example an applet, which has never been installed on its disk. This tutorial describes the use of dynamic code downloading in a Java system, and how it can be used with Java RMI.

- [The Java RMI trail of The Java Tutorial *Continued*](#)

This trail provides a brief overview of the Java RMI system and then walks through a complete client/server example that uses Java RMI's unique capabilities to load and to execute user-defined tasks at runtime. The server in the example implements a generic compute engine, which the client uses to compute the value of pi.

Enhancements

- [Java RMI Release Notes](#)

The release notes describe enhancements and changes to the Java RMI APIs and implementation, as well to the associated tools, `rmic`, `rmiregistry`, and `rmid`.

More Information

- [The Java RMI Home Page](#)
- [Java RMI Tools \(rmic, rmiregistry, rmid\)](#)
- [The Java RMI and Object Serialization FAQ](#)
- [Search the archives of the RMI-USERS list](#)
- [Subscribe to the RMI-USERS list](#)
- [Useful java.rmi Properties](#)
- [Useful sun.rmi Properties](#)
- [Java RMI Implementation Logging](#)
- [Applying the Factory Pattern to Java RMI](#)
- [Using Java RMI with SSL](#)