Java Platform, Standard Edition

Java Remote Method Invocation API Guide

Release 23

G10091-01

September 2024

Overview

Java Remote Method Invocation (Java RMI) lets you create distributed applications in Java.

RMI allows an object to invoke methods of remote Java objects running on another Java Virtual Machine (JVM), possibly on different hosts. RMI uses object serialization to marshal and unmarshal parameters, and doesn't truncate types, supporting true object-oriented polymorphism.

RMI Security Recommendations

Follow these recommendations to improve the security of your RMI applications.

- See Serialization Filtering and follow the best practices there to protect your applications.
- Follow Secure Coding Guidelines for Java SE.
- Optionally, you can run a security manager when using RMI, either on a client or server.

> **WARNING:**
>
> The Security Manager and APIs related to it have been deprecated and are subject to removal in a future release. There is no replacement for the Security Manager. See JEP 411 for discussion and alternatives.

- Establish a reasonable security policy. For example, grant `SocketPermission` and allow listen, accept, connect, and resolve actions only among hosts communicating with RMI. Don't have the security policy grant `AllPermission`. See Permissions in the Java Development Kit and Default Policy Implementation and Policy File Syntax.
- Restrict the communication to be local if RMI is being used only for communication among JVMs on the local host. To accomplish this task, specify the appropriate

socket permissions in the security policy file. Alternatively, you can use RMI APIs directly to restrict connections only to the local host. See the `RMISocketFactory` class.

- Ensure that the value of the `java.rmi.server.useCodebaseOnly` property is `True`. By default, the `java.rmi.server.useCodebaseOnly` property is set to `True`. If you set this property to `False`, then remote code loading is enabled, which increases the level of security risk to the system.
- Run RMI over Secure Sockets Layer (SSL)/Transport Layer Security (TLS) and request authentication for both server and client. This is possible using custom socket factories. An application can export a remote object to use custom socket factories that create sockets of a desired type (for example, SSL sockets). Using this technique, an application can use SSL socket communication instead of the default socket communication. See the following:
  - `SslRMIClientSocketFactory` class
  - `SslRMIServerSocketFactory` class
  - Java Secure Socket Extension (JSSE) Reference

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Java Platform, Standard Edition Java Remote Method Invocation API Guide, Release 23
G10091-01