TIBCO Rendezvous®

Configuration Tools

Software Release 8.5
December 2019

TIBCO®

Important Information

Contents

Tables

Preface

TIBCO Rendezvous® is a messaging infrastructure product.

TIBCO is proud to announce the latest release of TIBCO Rendezvous software. This release is the latest in a long history of TIBCO products that leverage the power of the Information Bus® technology to enable truly event-driven IT environments. To find out more about how TIBCO Rendezvous software and other TIBCO products are powered by TIB® technology, please visit us at www.tibco.com.

This manual describes tools for configuring daemons and other component processes of TIBCO Rendezvous software. It is part of the documentation set for Rendezvous Software Release 8.5.0.

Parts of the book *TIBCO Rendezvous Administration* describe the configuration of Rendezvous components using a graphical browser administration interface. This book describes a programmer interface and an XML tool for configuring the same parameters.

Topics

Manual Organization

This book begins with an introduction to the tools and the configuration API:

Most of the book details the objects and methods of the API:

One chapter describes a tool that you can use to quickly configure Rendezvous components from a command line. This tool can also apply a configuration stored in an XML file.

TIBCO Documentation and Support Services

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website, mainly in HTML and PDF formats.

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product. To access the latest documentation, visit https://docs.tibco.com.

TIBCO Rendezvous Documentation

TIBCO_HOME is the top-level directory in which TIBCO products are installed.

- On Windows platforms, the default TIBCO_HOME is C:\tibco.
- On UNIX platforms, the default TIBCO_HOME is /opt/tibco.

The following documents form the Rendezvous documentation set:

- *TIBCO Rendezvous Concepts*

  Read this book first. It contains basic information about Rendezvous components, principles of operation, programming constructs and techniques, advisory messages, and a glossary. All other books in the documentation set refer to concepts explained in this book.

- *TIBCO Rendezvous Administration*

  Begins with a checklist of action items for system and network administrators. This book describes the mechanics of Rendezvous licensing, network details, plus a chapter for each component of the Rendezvous software suite. Readers should have *TIBCO Rendezvous Concepts* at hand for reference.

- *TIBCO Rendezvous Installation*

  Includes step-by-step instructions for installing Rendezvous software on various operating system platforms.

- *TIBCO Rendezvous C Reference*

  Detailed descriptions of each datatype and function in the Rendezvous C API. Readers should already be familiar with the C programming language, as well as the material in *TIBCO Rendezvous Concepts*.

- *TIBCO Rendezvous C++ Reference*

  Detailed descriptions of each class and method in the Rendezvous C++ API. The C++ API uses some datatypes and functions from the C API, so we

recommend the *TIBCO Rendezvous C Reference* as an additional resource. Readers should already be familiar with the C++ programming language, as well as the material in *TIBCO Rendezvous Concepts*.

- *TIBCO Rendezvous .NET Reference*

  Detailed descriptions of each class and method in the Rendezvous .NET interface. Readers should already be familiar with either C# or Visual Basic .NET, as well as the material in *TIBCO Rendezvous Concepts*.

- *TIBCO Rendezvous Java Reference*

  Detailed descriptions of each class and method in the Rendezvous Java language interface. Readers should already be familiar with the Java programming language, as well as the material in *TIBCO Rendezvous Concepts*.

- *TIBCO Rendezvous Configuration Tools*

  Detailed descriptions of each Java class and method in the Rendezvous configuration API, plus a command line tool that can generate and apply XML documents representing component configurations. Readers should already be familiar with the Java programming language, as well as the material in *TIBCO Rendezvous Administration*.

- *TIBCO Rendezvous z/OS Installation and Configuration*

  Information about Rendezvous for IBM z/OS systems regarding installation and maintenance. Some information may be also useful for application programmers.

- *TIBCO Rendezvous for z/OS COBOL Reference*

  Detailed descriptions of each datatype and function in the Rendezvous COBOL API. Readers should already be familiar with the COBOL programming language, z/OS, as well as the material in *TIBCO Rendezvous Concepts*.

- *TIBCO Rendezvous Release Notes*

  Lists new features, changes in functionality, deprecated features, migration and compatibility information, closed issues and known issues.

How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit http://www.tibco.com/services/support .

- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support portal at https://support.tibco.com.
- If you already have a valid maintenance or support contract, visit this site:
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to https://support.tibco.com. If you do not have a user name, you can request one by clicking **Register** on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the TIBCO Ideas Portal. For a free registration, go to https://community.tibco.com.

Chapter 1 Overview

This chapter introduces the TIBCO Rendezvous® configuration tools, and presents important information for programmers who use this configuration API.

Topics

Scope of the Tools

The Rendezvous configuration tools set consists of three parts:

- An API for coding configuration programs.
- A general configuration tool.
- An XML syntax for configuration data.

Configuration API

The Rendezvous configuration API consists of a set of Java classes. You can use these classes and their methods to write programs that configure or examine Rendezvous daemons (and other component processes). This book presents the API in depth.

The methods can get parameters, set parameters, and get state information—the same set of operations that you do using the daemon's browser administration interface.

Using Java programs for these operations can speed the distribution of administrative changes to a large number of daemon processes.

Interacting with a daemon too frequently can degrade the daemon's performance (for example, repeatedly polling for statistical information). Use immediate-access methods conservatively.

Command Line Tool

Using the configuration API, we have implemented a command line tool for general use. `tibrvcfg` interacts with a component to execute one configuration command, and outputs the results to `stdio`.

You can use `tibrvcfg` to dump an XML configuration document reflecting the configuration of a Rendezvous component process. Other commands can load an XML document, so a component conforms to the configuration it specifies.

For details, see Command Line Tool—tibrvcfg on page 299.

XML

DTD files define the syntax of XML configuration documents for Rendezvous components. These definitions guide the command line tools to produce and parse correct XML documents.

API Architecture

Table 1 describes the Java classes that compose the configuration API. These classes belong to five categories:

- Manager
- Proxies
- Immediate-Access Objects
- Read-Only Objects
- Exceptions

Table 1 Overview of Configuration Classes (Sheet 1 of 2)

Class	Description
`DaemonManager`	Establish and manage a connection to the browser administration interface of a daemon process.
Proxy interfaces represent daemon component processes within a configuration program.	
`DaemonProxy`	This interface defines methods common to all Rendezvous components. The `DaemonManager` automatically creates a proxy instance, which the program uses as its main interface to the daemon process.

`RvdProxy`
`RvrdProxy`
`SecurityProxy`
`SecureDaemonProxy`
`RvcacheProxy`
	These interfaces define methods specific to each Rendezvous component. For example, `RvrdProxy` defines methods for interacting with `rvrd`.
	
	Each proxy interface represents one aspect of the behavior of the component. Some components incorporate several aspects, and they support the corresponding proxy interfaces. For example, `rvrd` has aspects of communications daemon (`rvd`) behavior, and aspects of routing daemon behavior (`rvrd`)—so it supports both `RvdProxy` and `RvrdProxy`.
	
	Programs cast the proxy instance to the appropriate proxy interfaces in order to call methods specific to the corresponding aspects.

Table 1 Overview of Configuration Classes (Sheet 2 of 2)

Class	Description
Immediate-access data structure classes represent data structures within component processes. Methods interact with the corresponding process.	
ClientTransport Service Host	These classes represent data structures within a communications daemon process (including rvd, rvsd, rvrd and rvsrd).
Router LocalNetworkInterface NeighborInterface ImportSubject	These classes represent data structures within a routing daemon process (including rvrd and rvsrd).
CertificateSlot	This class represents a data structure that can contain an X.509 certificate. These slots occur in processes that permit secure HTTPS connections (including rvd, rvsd, rvrd, rvsrd, and rvcache).
NetworkServicePair User UserCertificate	These classes represent data structures within a secure daemon process (including rvsd and rvsrd).
CachedField CachedSubject	These classes represent data structures within an rvcache process.
Read-only data structure classes contain information retrieved from component processes. Methods do *not* interact with component processes.	
ComponentInformation RvdInformation RvrdInformation RvsdInformation RvsrdInformation RvcacheInformation	These classes structure general information from the various components.
LoggingParams FaultToleranceParams RvcacheNetworkParams	These classes structure parameter information from the various components.
Exception classes	
ConfigurationException FatalConfigurationException	Methods of the configuration API throw these exception classes.

Data Accessors

Immediate Access

Proxy interfaces and immediate-access data structure classes define methods that access data within component processes. These methods access data *immediately* (though indirectly):

- Methods that *get* data from the component always fetch new data with each call (they do not return stored data from earlier calls).
- Methods that *set* values in the component immediately store the new values (they do not hold values while waiting for another call). If the component rejects a value, the method throws an exception.

This immediate-access paradigm ensures that configuration programs and daemon components always remain synchronized throughout their interactions.

Program Credentials

An important exception to this rule is `SecurityProxy.useCredentials()`, which is not a data access method. Instead, this method records an administrator name and password within the program (not within the daemon). A private method of `SecurityProxy` automatically supplies these credentials whenever the daemon requests them. For details see, `SecurityProxy.useCredentials()` on page 169.

Read-Only Objects

Each proxy interface defines a method that gets general information from the component. These methods return an instance of a subclass of `ComponentInformation`. All of these instances are read-only:

- Methods that *get* data from these objects do *not* interact with the component.
- Programs cannot construct these objects; they exist only because `getComponentInformation()` methods return them.
- Programs cannot modify these objects.

Program Structure

Examples

Programming examples are included on the installation media. When you install Rendezvous software, these examples appear in the directory `src/examples/configapi`.

We encourage you to examine these programs before writing your own programs.

Structure

The basic structure of programs follows these steps:

1. Define subclasses of `javax.net.ssl.HostnameVerifier` and `javax.net.ssl.X509TrustManager` to specify the security behavior of your program.
2. Create an instance of `DaemonManager`.

   The constructor automatically creates an instance of `DaemonProxy`, and stores it in the new manager instance.
3. Get the daemon proxy instance from the daemon manager.
4. Cast the daemon proxy instance to one of the specific aspect proxy interfaces, and call methods of that aspect to access and configure the daemon.

   You may subsequently cast the daemon proxy to different aspect proxy interfaces, and call their methods.

Requirements

Java SDK

The Rendezvous configuration API requires that you first install Java SDK runtime environment 1.4 (or later). You can download this software from java.sun.com.

Environment Variables

Variable	Description
CLASSPATH	Rendezvous installation places the Java archive file rvconfig.jar in the lib directory under TIBRV_HOME. If you have placed this file in any other location, the CLASSPATH variable must include the complete pathname. The CLASSPATH variable must also include the complete pathname to the SDK file jsse.jar, which implements TLS.

Chapter 2 Daemon Manager

This chapter describes the top-level objects in Rendezvous configuration programs.

Topics

DaemonManager

Class

Declaration

```
class com.tibco.tibrv.config.DaemonManager
  extends java.lang.Object
```

Purpose Establish and manage a connection to the browser administration interface of a daemon process.

Although it has a similar name, this class is *not* related to the Daemon Manager (RVDM) feature.

Remarks Each instance connects to *one* process instance of a daemon.

Constant	Description
DaemonManager.UNKNOWN	DaemonManager.getDaemonType() returns these integer constants to indicate the type of daemon to which it has connected.
DaemonManager.RVD	
DaemonManager.RVRD	
DaemonManager.RVSD	
DaemonManager.RVSRD	
DaemonManager.RVCACHE	

Method	Description	Page
DaemonManager()	Create a daemon manager and initialize its connection to a daemon process.	11
DaemonManager.getDaemonType()	Return an integer indicating the type of daemon to which the daemon manager has connected.	12
DaemonManager.getDaemonProxy()	Return the proxy instance representing the daemon process.	13

DaemonManager()

Constructor

Declaration

```
DaemonManager(java.lang.String url)
  throws ConfigurationException
```

Purpose

Create a daemon manager and initialize its connection to a daemon process.

Remarks

Each instance connects to *one* process instance of a daemon. The URL parameter specifies the location of the browser administration interface for that process.

This constructor automatically creates a proxy instance (`java.lang.reflect.Proxy`) within the `DaemonManager` object. The proxy serves as the program's command interface to the daemon component. Programs extract the proxy instance using `DaemonManager.getDaemonProxy()`, and cast it to the appropriate proxy interfaces to access the corresponding daemon or component.

Parameter	Description
`url`	Connect to a daemon at this URL. For example: `http://localhost:7580`

DaemonManager.getDaemonType()

Method

Declaration

```
int getDaemonType()
  throws ConfigurationException
```

Purpose Return an integer indicating the type of daemon to which the daemon manager has connected.

Remarks For a list of values that this method can return, see the constants defined at DaemonManager on page 10.

DaemonManager.getDaemonProxy()

Method

Declaration

```
DaemonProxy getDaemonProxy()
  throws ConfigurationException
```

Purpose

Return the proxy instance representing the daemon process.

Remarks

The constructor `DaemonManager()` automatically creates an appropriate proxy instance for the daemon component. Programs use this proxy instance to configure the daemon. A variety of proxy interfaces allow programs to access parameter values of the daemon. Each proxy interface defines a set of methods related to a specific aspect of daemon behavior.

Programs cast the proxy instance to an appropriate proxy interface, then call methods of the interface.

See Also

`DaemonProxy` on page 14
`RvdProxy` on page 22
`RvrdProxy` on page 76
`SecurityProxy` on page 162
`SecureDaemonProxy` on page 180
`RvcacheProxy` on page 230

DaemonProxy

Interface

Declaration

```
interface com.tibco.tibrv.config.DaemonProxy
  extends XmlSerializable
```

Purpose Define methods common to all Rendezvous components.

Method	Description	Page
`DaemonProxy.getComponentName()`	Get the name of the daemon.	15
`DaemonProxy.getComponentInformation()`	Get basic component information from the daemon.	16

Inherited Methods

`XmlSerializable.printXml()`

Components These components support this interface:

```
rvd
rvrd
rvsd
rvsrd
rvcache
```

For information about these components, see *TIBCO Rendezvous Administration*

Subinterfaces

`RvdProxy` on page 22
`RvrdProxy` on page 76
`SecurityProxy` on page 162
`SecureDaemonProxy` on page 180
`RvcacheProxy` on page 230

DaemonProxy.getComponentName()

Method

Declaration

```
java.lang.String getComponentName()
  throws ConfigurationException
```

Purpose Get the name of the daemon.

Remarks This method gets the name from the daemon component and returns it as a printable string.

DaemonProxy.getComponentInformation()

Method

Declaration

```
ComponentInformation getComponentInformation()
  throws ConfigurationException
```

Purpose

Get basic component information from the daemon.

Remarks

This method returns the contents of the General Information or Component Information page from the component's browser administration interface. The information content varies depending on the type of the component, and its release. For details, see *TIBCO Rendezvous Administration*.

This method returns an object from which you can extract the information content. That object is an instance of one of these classes:

`ComponentInformation` on page 270
`RvdInformation` on page 284
`RvrdInformation` on page 288
`RvsdInformation` on page 291
`RvsrdInformation` on page 293
`RvcacheInformation` on page 278

XmlSerializable

Interface

Declaration `interface com.tibco.tibrv.config.XmlSerializable`

Purpose Define methods for producing XML that describes Rendezvous components.

Method	Description	Page
`XmlSerializable.printXml()`	Print the object as an XML document.	18
`XmlSerializable.toXml()`	Format the object as an XML document.	19

XmlSerializable.printXml()

Method

Declaration

```
void printXml(
    java.io.OutputStream  outputStream)

void printXml(
    java.io.Writer  writer)
```

Purpose Print the object as an XML document.

Parameter	Description
`outputStream`	Print to this output stream.
`writer`	Print to this character stream.

XmlSerializable.toXml()

Method

Declaration `java.lang.String` **`toXml`**`()`

Purpose Format the object as an XML document.

Remarks Only configurable objects support this method. For example, proxies for routing daemons do not support it.

Chapter 3 Communications Daemon—rvd

This chapter describes the proxy interface for the Rendezvous communications daemon (`rvd`), and the immediate-access data objects that support it.

Topics

See Also

RvdProxy

Interface

Declaration

```
interface com.tibco.tibrv.config.RvdProxy
  extends DaemonProxy
```

Purpose Define methods for Rendezvous communications daemons.

Method	Description	Page
RvdProxy.getClientTransports()	Get the client transports of the daemon.	23
RvdProxy.getPortMap()	Get the port map from a managed daemon.	24
RvdProxy.getServices()	Get the network services on which the daemon communicates.	25
RvdProxy.getSubectMaps()	Get the subject maps from a managed daemon.	26

Inherited Methods

DaemonProxy.getComponentName()
DaemonProxy.getComponentInformation()

XmlSerializable.printXml()

Components These components support this interface:

rvd
rvrd
rvsd
rvsrd

See Also For information about the parameters that this interface can access, see these sections:

- Rendezvous Daemon (rvd) on page 35 in *TIBCO Rendezvous Administration*
- Browser Administration Interface—rvd on page 51 in *TIBCO Rendezvous Administration*
- General Information on page 54 in *TIBCO Rendezvous Administration*

RvdProxy.getClientTransports()

Method

Declaration

```
ClientTransport[] getClientTransports()
  throws ConfigurationException
```

Purpose Get the client transports of the daemon.

See Also `ClientTransport` on page 27
Clients on page 56 in *TIBCO Rendezvous Administration*

RvdProxy.getPortMap()

Method

Declaration

```
PortMap getPortMap()
  throws ConfigurationException
```

Purpose Get the port map from a managed daemon.

Remarks If the daemon is not a managed daemon, this method throws a `ConfigurationException`.

See Also `PortMap` on page 57
For the corresponding browser page, see Port Map on page 69 in *TIBCO Rendezvous Administration*
For concept information, see Port Map on page 222 in *TIBCO Rendezvous Administration*

RvdProxy.getServices()

Method

Declaration

```
Service[] getServices()
  throws ConfigurationException
```

Purpose Get the network services on which the daemon communicates.

See Also Service on page 43
Services on page 59 in *TIBCO Rendezvous Administration*

RvdProxy.getSubectMaps()

Method

Declaration

```
SubjectMap[] getSubectMaps()
  throws ConfigurationException
```

Purpose Get the subject maps from a managed daemon.

Remarks If the daemon is not a managed daemon, this method throws a `ConfigurationException`.

See Also `SubjectMap` on page 65
For the corresponding browser page, see Subject Map Summary on page 66 in *TIBCO Rendezvous Administration*
For concept information, see Subject Map on page 213 in *TIBCO Rendezvous Administration*

ClientTransport

Class

Declaration

```
class com.tibco.tibrv.config.ClientTransport
  extends java.lang.Object
```

Purpose Represent a client transport of a Rendezvous communications daemon.

Remarks The method `RvdProxy.getClientTransports()` returns an array of objects of this class.

Method	Description	Page
`ClientTransport.getDescription()`	Get the description string of the transport.	28
`ClientTransport.getDetails()`	Get detailed information about the client transport.	29
`ClientTransport.getIdentifier()`	Get the globally unique identifier for the transport object.	30
`ClientTransport.getService()`	Get the UDP service on which the transport communicates.	31
`ClientTransport.getSubscriptions()`	Get the subscriptions that this transport has registered with the daemon.	32
`ClientTransport.getUsername()`	Get the user name of the transport's program.	33
`ClientTransport.toXml()`	Format the client transport information as an XML document.	34

See Also `RvdProxy.getClientTransports()` on page 23

For the corresponding browser page, see Clients on page 56 in *TIBCO Rendezvous Administration*

ClientTransport.getDescription()

Method

Declaration `java.lang.String` **`getDescription`**`()`

Purpose Get the description string of the transport.

Remarks The client program supplies this string to identify the program and transport.

See Also `ClientTransport` on page 27

For the corresponding browser page, see Clients on page 56 in *TIBCO Rendezvous Administration*

ClientTransport.getDetails()

Method

Declaration `java.util.Map` **`getDetails`**`()`

Purpose Get detailed information about the client transport.

See Also `ClientTransport` on page 27

For the corresponding browser page, see Client Information on page 56 in *TIBCO Rendezvous Administration*

ClientTransport.getIdentifier()

Method

Declaration `java.lang.String` **`getIdentifier`**`()`

Purpose Get the globally unique identifier for the transport object.

See Also `ClientTransport` on page 27

For the corresponding browser page, see Clients on page 56 in *TIBCO Rendezvous Administration*

ClientTransport.getService()

Method

Declaration `int getService()`

Purpose Get the UDP service on which the transport communicates.

See Also `ClientTransport` on page 27

For the corresponding browser page, see Clients on page 56 in *TIBCO Rendezvous Administration*

ClientTransport.getSubscriptions()

Method

Declaration `java.lang.String[] getSubscriptions()`

Purpose Get the subscriptions that this transport has registered with the daemon.

Remarks Each string in the table is the subject name of one subscription.

The daemon limits this list to the first page of subscriptions; if the list would span more than one page, the subscriptions on the remaining pages are not available.

See Also `ClientTransport` on page 27

For the corresponding browser page, see Client Information on page 56 in *TIBCO Rendezvous Administration*

ClientTransport.getUsername()

Method

Declaration `java.lang.String` **`getUsername`**`()`

Purpose Get the user name of the transport's program.

See Also `ClientTransport` on page 27

For the corresponding browser page, see Clients on page 56 in *TIBCO Rendezvous Administration*

ClientTransport.toXml()

Method

Declaration `java.lang.String toXml()`

Purpose Format the client transport information as an XML document.

See Also `ClientTransport` on page 27

For the corresponding browser page, see Clients on page 56 in *TIBCO Rendezvous Administration*

Host

Class

Declaration

```
class com.tibco.tibrv.config.Host
  extends java.lang.Object
```

Purpose Represent the host computer of a Rendezvous communications daemon.

Remarks Objects of this class always represent a host computer *other* than the host computer of the daemon. That is, they represent computers with which this daemon communicates.

The method `Service.getHosts()` return objects of this class.

Method	Description	Page
`Host.getHostname()`	Get the hostname of the computer that this object represents.	36
`Host.getHttpAddress()`	Get the address where the host computer listens for HTTP (browser interface) connections.	37
`Host.getIpAddress()`	Get the IP address of the computer.	38
`Host.getSerial()`	Get the serial number of the Rendezvous license ticket.	39
`Host.getUptime()`	Get the elapsed time that the daemon has been using the UDP service.	40
`Host.getVersion()`	Get the version of the Rendezvous daemon running on a host.	41
`Host.toXml()`	Format the host computer information as an XML document.	42

See Also `Service.getHosts()` on page 49

For the corresponding browser page, see Host List on page 64 in *TIBCO Rendezvous Administration*

Host.getHostname()

Method

Declaration `java.lang.String` **`gethostname`**`()`

Purpose Get the hostname of the computer that this object represents.

See Also `Host` on page 35

For the corresponding browser page, see Host List on page 64 in *TIBCO Rendezvous Administration*

Host.getHttpAddress()

Method

Declaration `java.lang.String` **`getHttpAddress`**`()`

Purpose Get the address where the host computer listens for HTTP (browser interface) connections.

Remarks The address follows the template `http://`*host*`:`*port*

- *host* is the hostname of the computer.
- *port* is the port where the daemon accepts HTTP clients.

See Also `Host` on page 35

Host.getIpAddress()

Method

Declaration `java.lang.String` **`getIpAddress`**`()`

Purpose Get the IP address of the computer.

See Also `Host` on page 35

For the corresponding browser page, see Host List on page 64 in *TIBCO Rendezvous Administration*

Host.getSerial()

Method

Declaration `java.lang.String` **`getSerial`**`()`

Purpose Get the serial number of the Rendezvous license ticket.

Remarks Most Rendezvous components require a valid license ticket for operation. This method returns the ticket that validates the component on this host computer.

See Also `Host` on page 35

For the corresponding browser page, see Host List on page 64 in *TIBCO Rendezvous Administration*

Host.getUptime()

Method

Declaration

```
java.lang.String getUptime()
```

Purpose

Get the elapsed time that the daemon has been using the UDP service.

Remarks

The uptime of a remote host on a service represents the cumulative time that the host (that is, the daemon on that host) has been using that service on behalf of one or more client transports.

When a daemon first begins using a service on behalf of a client transport, it starts counting the uptime for that service from zero. The daemon counts uptime separately for each service it uses.

As long as the daemon still has one or more client transports on the particular service, its uptime continues to increase. When no more client transports remain on that service, the daemon stops using the service (after a short delay). If a new transport subsequently begins using the service again, the daemon begins counting uptime for that service at zero.

See Also

`Host` on page 35
`Service.getHosts()` on page 49

For the corresponding browser page, see Host List on page 64 in *TIBCO Rendezvous Administration*

Host.getVersion()

Method

Declaration `java.lang.String` **`getVersion`**`()`

Purpose Get the version of the Rendezvous daemon running on a host.

See Also `Host` on page 35

For the corresponding browser page, see Host List on page 64 in *TIBCO Rendezvous Administration*

Host.toXml()

Method

Declaration `java.lang.String` **`toXml`**`()`

Purpose Format the host computer information as an XML document.

See Also `Host` on page 35

For the corresponding browser page, see Host List on page 64 in *TIBCO Rendezvous Administration*

Service

Class

Declaration

```
class com.tibco.tibrv.config.Service
  extends java.lang.Object
```

Purpose Represent a UDP service of a Rendezvous communications daemon.

Remarks This object represents a communications daemon's activity on a specific network service—that is, a physical network combined with a UDP service.

The method `RvdProxy.getServices()` returns objects of this class.

(Sheet 1 of 2)

Method	Description	Page
`Service.getClientCount()`	Get the number of client transports that use this service.	45
`Service.getClientTransports()`	Get the client transports that use this service.	46
`Service.getDetails()`	Get detailed information about the network service.	47
`Service.getHostCount()`	Get the number of other host computers with daemons that communicate on this network and service.	48
`Service.getHosts()`	Get the other host computers with daemons that communicate on this network and service.	49
`Service.getInboundRates()`	Get recent statistics about inbound data on this service.	50
`Service.getInboundTotals()`	Get cumulative statistics about inbound data.	51
`Service.getNetwork()`	Get the network number.	52
`Service.getOutboundRates()`	Get recent statistics about outbound data on this service.	53
`Service.getOutboundTotals()`	Get cumulative statistics about outbound data.	54
`Service.getPortNumber()`	Get the UDP service number.	55

(Sheet 2 of 2)

Method	Description	Page
`Service.getSubscriptions()`	Get the client subscriptions registered with this daemon on the network service.	56

See Also `RvdProxy.getServices()` on page 25

For the corresponding browser page, see Services on page 59 in *TIBCO Rendezvous Administration*

Service.getClientCount()

Method

Declaration `int getClientCount()`

Purpose Get the number of client transports that use this service.

See Also
`ClientTransport` on page 27
`Service` on page 43
`Service.getClientTransports()` on page 46

For the corresponding browser page, see Services on page 59 in *TIBCO Rendezvous Administration*

Service.getClientTransports()

Method

Declaration `ClientTransport[] getClientTransports()`

Purpose Get the client transports that use this service.

Remarks This method is similar to `RvdProxy.getClientTransports()`, except that it gets only the transports that use this service.

`Service.getClientCount()` on page 45 returns the size of this client array.

See Also
`RvdProxy.getClientTransports()` on page 23
`ClientTransport` on page 27
`Service` on page 43
`Service.getClientCount()` on page 45

For the corresponding browser page, see Service Information on page 60 in *TIBCO Rendezvous Administration*

Service.getDetails()

Method

Declaration `java.util.Map` **`getDetails`**`()`

Purpose Get detailed information about the network service.

See Also `Service` on page 43

For the corresponding browser page, see Service Information on page 60 in *TIBCO Rendezvous Administration*

Service.getHostCount()

Method

Declaration

```
int getHostCount()
```

Purpose

Get the number of *other* host computers with daemons that communicate on this network and service.

See Also

`Host` on page 35
`Service` on page 43
`Service.getHosts()` on page 49

For the Services browser page, see Services on page 59 in *TIBCO Rendezvous Administration*

For the Hosts browser page, see Host List on page 64 in *TIBCO Rendezvous Administration*

Service.getHosts()

Method

Declaration `Host[] getHosts()`

Purpose Get the *other* host computers with daemons that communicate on this network and service.

Remarks `Service.getHostCount()` on page 48 returns the size of this host array.

See Also
`Host` on page 35
`Service` on page 43
`Service.getHostCount()` on page 48

For the Services browser page, see Service Information on page 60 in *TIBCO Rendezvous Administration*

For the Hosts browser page, see Host List on page 64 in *TIBCO Rendezvous Administration*

Service.getInboundRates()

Method

Declaration `java.util.Map` **`getInboundRates`**`()`

Purpose Get recent statistics about inbound data on this service.

Remarks This method returns a set of statistics about inbound data on this network service during the most recent sampling period:

- `msgs`—the rate (per second) at which the daemon received inbound messages
- `bytes`—the rate (per second) at which the daemon received inbound bytes
- `pkts`—the rate (per second) at which the daemon received inbound packets

See Also `Service` on page 43
`Service.getOutboundRates()` on page 53

For the corresponding browser page, see Service Information on page 60 in *TIBCO Rendezvous Administration*

Service.getInboundTotals()

Method

Declaration `java.util.Map` **`getInboundTotals`**`()`

Purpose Get cumulative statistics about inbound data.

Remarks The return value contains a set of running totals for inbound data on this network service, accumulated since the start of the daemon process:

- `msgs`—number of messages
- `bytes`—number of bytes
- `pkts`—number of packets
- `missed`—number of missed packets (detected as a packet sequence gap)
- `lostMc`—number of multicast packets lost because the sending daemon could not retransmit them
- `lostPtp`—number of point-to-point packets lost because the sending daemon could not retransmit them

See Also `Service` on page 43
`Service.getOutboundTotals()` on page 54

For the corresponding browser page, see Service Information on page 60 in *TIBCO Rendezvous Administration*

Service.getNetwork()

Method

Declaration `java.util.String` **`getNetwork`**`()`

Purpose Get the network number.

See Also `Service` on page 43

For the corresponding browser page, see Service Information on page 60 in *TIBCO Rendezvous Administration*

Service.getOutboundRates()

Method

Declaration `java.util.Map` **`getOutboundRates`**`()`

Purpose Get recent statistics about outbound data on this service.

Remarks This method returns a set of statistics about outbound data on this network service during the most recent sampling period:

- `msgs`—the rate (per second) at which the daemon sent outbound messages
- `bytes`—the rate (per second) at which the daemon sent outbound bytes
- `pkts`—the rate (per second) at which the daemon sent outbound packets

See Also `Service` on page 43
`Service.getInboundRates()` on page 50

For the corresponding browser page, see Service Information on page 60 in *TIBCO Rendezvous Administration*

Service.getOutboundTotals()

Method

Declaration `java.util.Map` **`getOutboundTotals`**`()`

Purpose Get cumulative statistics about outbound data.

Remarks The return value contains a set of running totals for outbound data on this network service, accumulated since the start of the daemon process:

- `msgs`—number of messages
- `bytes`—number of bytes
- `pkts`—number of packets
- `retrans`—number of packets retransmitted (multicast and point-to-point)
- `lostMc`—number of multicast packets the daemon could not retransmit (too old)
- `lostPtp`—number of point-to-point packets the daemon could not retransmit (too old)

See Also `Service` on page 43
`Service.getInboundTotals()` on page 51

For the corresponding browser page, see Service Information on page 60 in *TIBCO Rendezvous Administration*

Service.getPortNumber()

Method

Declaration `int getPortNumber()`

Purpose Get the UDP service number.

See Also `Service` on page 43

For the corresponding browser page, see Service Information on page 60 in *TIBCO Rendezvous Administration*

Service.getSubscriptions()

Method

Declaration `java.lang.String[] getSubscriptions()`

Purpose Get the client subscriptions registered with this daemon on the network service.

Remarks Each string in the table is the subject name of one subscription.

The daemon limits this list to 50 subscriptions; if the list would have more than 50 subscriptions, the daemon replaces them with only one item, which describes the approximate number of subscriptions.

See Also `Service` on page 43

For the corresponding browser page, see Service Information on page 60 in *TIBCO Rendezvous Administration*.

PortMap

Class

Declaration

```
class com.tibco.tibrv.config.PortMap
  extends java.lang.Object
```

Purpose Represent an RVDM port map page.

Remarks This object represents the RVDM port map page of a managed daemon.

The method `RvdProxy.getPortMap()` returns objects of this class.

Method	Description	Page
`PortMap.isEnabled()`	Determine whether the port map feature is enabled.	58
`PortMap.getLastUpdate()`	Get the timestamp indicating when RVDM last updated the port map on the managed daemon.	59
`PortMap.getPortMapEntries()`	Get the rows of the port map page.	60

See Also `RvdProxy.getPortMap()` on page 24

For the corresponding browser page, see Port Map on page 69 in *TIBCO Rendezvous Administration*
For concept information, see Port Map on page 222 in *TIBCO Rendezvous Administration*

PortMap.isEnabled()

Method

Declaration `boolean isEnabled()`

Purpose Determine whether the port map feature is enabled.

See Also `PortMap` on page 57

For the corresponding browser page, see Port Map on page 69 in *TIBCO Rendezvous Administration*
For concept information, see Port Map on page 222 in *TIBCO Rendezvous Administration*

PortMap.getLastUpdate()

Method

Declaration `java.util.Date` **`getLastUpdate`**`()`

Purpose Get the timestamp indicating when RVDM last updated the port map on the managed daemon.

See Also `PortMap` on page 57

For the corresponding browser page, see Port Map on page 69 in *TIBCO Rendezvous Administration*
For concept information, see Port Map on page 222 in *TIBCO Rendezvous Administration*

PortMap.getPortMapEntries()

Method

Declaration `PortMapEntry[] getPortMapEntries()`

Purpose Get the rows of the port map page.

See Also `PortMap` on page 57

For the corresponding browser page, see Port Map on page 69 in *TIBCO Rendezvous Administration*
For concept information, see Port Map on page 222 in *TIBCO Rendezvous Administration*

PortMapEntry

Class

Declaration

```
class com.tibco.tibrv.config.PortMapEntry
  extends java.lang.Object
```

Purpose Represent a row within a port map table.

Remarks This object represents one table row of the RVDM port map page of a managed daemon.

The method `PortMap.getPortMapEntries()` returns an array of objects of this class.

Method	Description	Page
`PortMapEntry.getClientCount()`	Get the number of clients affected by a port mapping.	62
`PortMapEntry.getEffectivePort()`	Get the effective port (of a service pair) from a port map row.	63
`PortMapEntry.getOriginalPort()`	Get the original port (of a service pair) from a port map row.	64

See Also `PortMap.getPortMapEntries()` on page 60

For the corresponding browser page, see Port Map on page 69 in *TIBCO Rendezvous Administration*
For concept information, see Port Map on page 222 in *TIBCO Rendezvous Administration*

PortMapEntry.getClientCount()

Method

Declaration

```
short getClientCount()
```

Purpose

Get the number of clients affected by a port mapping.

See Also

`PortMapEntry` on page 61

For the corresponding browser page, see Port Map on page 69 in *TIBCO Rendezvous Administration*
For concept information, see Port Map on page 222 in *TIBCO Rendezvous Administration*

PortMapEntry.getEffectivePort()

Method

Declaration `short` **`getEffectivePort`**`()`

Purpose Get the effective port (of a service pair) from a port map row.

See Also `PortMapEntry` on page 61

For the corresponding browser page, see Port Map on page 69 in *TIBCO Rendezvous Administration*
For concept information, see Port Map on page 222 in *TIBCO Rendezvous Administration*

PortMapEntry.getOriginalPort()

Method

Declaration `short` **`getOriginalPort`**`()`

Purpose Get the original port (of a service pair) from a port map row.

See Also `PortMapEntry` on page 61

For the corresponding browser page, see Port Map on page 69 in *TIBCO Rendezvous Administration*
For concept information, see Port Map on page 222 in *TIBCO Rendezvous Administration*

SubjectMap

Class

Declaration

```
class com.tibco.tibrv.config.SubjectMap
  extends java.lang.Object
```

Purpose Represent a row of the subject map summary page of a managed daemon.

Remarks This object represents one subject map (that is, one row of the summary table) corresponding to a specific service.

The method `RvdProxy.getSubectMaps()` returns an array of objects of this class.

Method	Description	Page
`SubjectMap.getClientCount()`	Get the number of clients affected by a subject map.	66
`SubjectMap.getLastUpdate()`	Get the timestamp indicating when RVDM last updated the subject map on the managed daemon.	67
`SubjectMap.getSubjectMapEntries()`	Get the details for each subject map.	68
`SubjectMap.getSubscriptionCount()`	Get the number of subscriptions using the service of a subject map.	69

See Also `RvdProxy.getSubectMaps()` on page 26
For the corresponding browser page, see Subject Map Summary on page 66 in *TIBCO Rendezvous Administration*
For concept information, see Subject Map on page 213 in *TIBCO Rendezvous Administration*

SubjectMap.getClientCount()

Method

Declaration `int getClientCount()`

Purpose Get the number of clients affected by a subject map.

See Also `SubjectMap` on page 65
For the corresponding browser page, see Subject Map Summary on page 66 in *TIBCO Rendezvous Administration*
For concept information, see Subject Map on page 213 in *TIBCO Rendezvous Administration*

SubjectMap.getLastUpdate()

Method

Declaration `java.util.Date` **`getLastUpdate`**`()`

Purpose Get the timestamp indicating when RVDM last updated the subject map on the managed daemon.

See Also `SubjectMap` on page 65
For the corresponding browser page, see Subject Map Summary on page 66 in *TIBCO Rendezvous Administration*
For concept information, see Subject Map on page 213 in *TIBCO Rendezvous Administration*

SubjectMap.getSubjectMapEntries()

Method

Declaration `SubjectMapEntry[] getSubjectMapEntries()`

Purpose Get the details for each subject map.

See Also `SubjectMapEntry` on page 70
For the corresponding browser page, see Subject Map Detail on page 66 in *TIBCO Rendezvous Administration*
For concept information, see Subject Map on page 213 in *TIBCO Rendezvous Administration*

SubjectMap.getSubscriptionCount()

Method

Declaration `int getSubscriptionCount()`

Purpose Get the number of subscriptions using the service of a subject map.

See Also `SubjectMap` on page 65
For the corresponding browser page, see Subject Map Summary on page 66 in *TIBCO Rendezvous Administration*
For concept information, see Subject Map on page 213 in *TIBCO Rendezvous Administration*

SubjectMapEntry

Class

Declaration

```
class com.tibco.tibrv.config.SubjectMapEntry
  extends java.lang.Object
```

Purpose Represent a row on a subject map detail page.

Remarks The method `SubjectMap.getSubjectMapEntries()` returns objects of this class.

Method	Description	Page
`SubjectMapEntry.getGroup()`	Get the multicast group of a mapping row.	71
`SubjectMapEntry.getRank()`	Get the rank of a mapping row.	72
`SubjectMapEntry.getSubject()`	Get the subject of a mapping row.	73
`SubjectMapEntry.getSubscriberCount()`	Get the number of client transports that have at least one matching listener.	74

See Also

`SubjectMap.getSubjectMapEntries()` on page 68

For the corresponding browser page, see Subject Map Detail on page 66 in *TIBCO Rendezvous Administration*

For concept information, see Subject Map on page 213 in *TIBCO Rendezvous Administration*

SubjectMapEntry.getGroup()

Method

Declaration `java.lang.String getGroup()`

Purpose Get the multicast group of a mapping row.

See Also `SubjectMapEntry` on page 70
For the corresponding browser page, see Subject Map Detail on page 66 in *TIBCO Rendezvous Administration*
For concept information, see Subject Map on page 213 in *TIBCO Rendezvous Administration*

SubjectMapEntry.getRank()

Method

Declaration `int getRank()`

Purpose Get the rank of a mapping row.

Remarks For default system groups and default user groups, this method returns -1 as the rank.

See Also `SubjectMapEntry` on page 70
For the corresponding browser page, see Subject Map Detail on page 66 in *TIBCO Rendezvous Administration*
For concept information, see Subject Map on page 213 in *TIBCO Rendezvous Administration*

SubjectMapEntry.getSubject()

Method

Declaration `java.lang.String` **`getSubject`**`()`

Purpose Get the subject of a mapping row.

Remarks For the default system groups row, this method returns the string `Default System Groups` instead of a Rendezvous subject.

For the default user groups row, this method returns the string `Default User Groups` instead of a Rendezvous subject.

See Also `SubjectMapEntry` on page 70
For the corresponding browser page, see Subject Map Detail on page 66 in *TIBCO Rendezvous Administration*
For concept information, see Subject Map on page 213 in *TIBCO Rendezvous Administration*

SubjectMapEntry.getSubscriberCount()

Method

Declaration `int` **`getSubscriberCount`**`()`

Purpose Get the number of client transports that have at least one matching listener.

Remarks A matching listener is a listener with a subscription subject that matches the subject of this subject mapping row.

See Also `SubjectMapEntry` on page 70
For the corresponding browser page, see Subject Map Detail on page 66 in *TIBCO Rendezvous Administration*
For concept information, see Subject Map on page 213 in *TIBCO Rendezvous Administration*

Chapter 4 Routing Daemon—rvrd

This chapter describes the proxy interface for the Rendezvous routing daemon (`rvrd`), and the immediate-access data objects that support it.

Topics

See Also

- RvrdInformation, page 288

RvrdProxy

Interface

Declaration

```
interface com.tibco.tibrv.config.RvrdProxy
  extends DaemonProxy
```

Purpose Define methods for Rendezvous routing daemons.

Method	Description	Page
RvrdProxy.addBorderRouter()	Add a border router to the routing daemon.	78
RvrdProxy.addRouter() RvrdProxy.addRouters()	Add router names to the routing daemon.	79
RvrdProxy.getLoggingParams()	Get the flags that guide the routing daemon's log output.	80
RvrdProxy.getRouter() RvrdProxy.getRouters()	Get routers from the routing daemon.	81
RvrdProxy.removeRouter() RvrdProxy.removeRouters()	Remove router names from the routing daemon.	82
RvrdProxy.setLoggingParams()	Set the flags that guide the routing daemon's log output.	83

Inherited Methods

DaemonProxy.getComponentName()
DaemonProxy.getComponentInformation()

XmlSerializable.printXml()
XmlSerializable.toXml()

Components These components support this interface:

```
rvrd
rvsrd
```

See Also For information about the parameters that this interface can access, see these sections:

- Routing Daemon (rvrd) on page 71 in *TIBCO Rendezvous Administration*

- Browser Administration Interface—rvrd on page 128 in *TIBCO Rendezvous Administration*
- General Information on page 132 in *TIBCO Rendezvous Administration*

RvrdProxy.addBorderRouter()

Method

Declaration

```
BorderRouter addBorderRouter(
    java.lang.String routerName)
  throws ConfigurationException
```

Purpose Add a border router to the routing daemon.

Remarks Border routing restricts permissible configurations. When an `rvrd` process is configured as a border router, that border router must be the only routing table entry for the process.

As a result, you can configure an `rvrd` process either as a collection of one or more first-tier routers, or as exactly one border router. You cannot configure more than one border router in a process, nor mix first-tier and border routers in the same process.

Once a border router is configured, you cannot remove it, rename it, nor convert it to a first-tier router. Nor can you convert a first-tier router to a border router.

Parameter	Description
`routerName`	Add this router name.

See Also

`RvrdProxy.getRouter()` on page 81
`BorderRouter` on page 144

For the corresponding browser pages, see Border Routing on page 145 in *TIBCO Rendezvous Administration*, and the sections that follow it

For background information, see Border Routing on page 105 in *TIBCO Rendezvous Administration*

RvrdProxy.addRouter()

Method

Related Forms RvrdProxy.addRouters()

Declaration

```
Router addRouter(
    java.lang.String routerName)
  throws ConfigurationException

Router[] addRouters(
    java.lang.String[] routerNames)
  throws ConfigurationException
```

Purpose Add router names to the routing daemon.

Remarks When adding more than one router name, the second method is faster than repeatedly calling the first method.

Parameter	Description
`routerName`	Add this router name.
`routerNames`	Add all the router names in this array.

See Also

`RvrdProxy.getRouter()` on page 81
`RvrdProxy.removeRouter()` on page 82
`Router` on page 121

For background information, see Routing Table Entry on page 79 in *TIBCO Rendezvous Administration*

For the corresponding browser page, see Routers on page 144 in *TIBCO Rendezvous Administration*

RvrdProxy.getLoggingParams()

Method

Declaration

```
LoggingParams getLoggingParams()
  throws ConfigurationException
```

Purpose

Get the flags that guide the routing daemon's log output.

Remarks

This method returns an object encapsulating the logging parameters. To examine the individual values, use the methods of the class `LoggingParams` on page 103.

See Also

`RvrdProxy` on page 76
`RvrdProxy.setLoggingParams()` on page 83
`LoggingParams` on page 103

For background information, see Interpreting Log Output on page 117 in *TIBCO Rendezvous Administration*

For the corresponding browser page, see Daemon Parameters on page 141 in *TIBCO Rendezvous Administration*, and Logging on page 143 in *TIBCO Rendezvous Administration*

RvrdProxy.getRouter()

Method

Related Forms RvrdProxy.getRouters()

Declaration

```
Router getRouter(
    java.lang.String routerName)
  throws ConfigurationException

Router[] getRouters()
  throws ConfigurationException
```

Purpose Get routers from the routing daemon.

Remarks `getRouters()` returns an array containing all the routers configured for the routing daemon.

`getRouter()` queries the routing daemon for a router with a specific router name, and returns that router (if it exists). If it does not exist, the method throws an exception.

Parameter	Description
`routerName`	Return the router with this router name, if it exists.

See Also `RvrdProxy.addRouter()` on page 79
`Router` on page 121

For background information, see Routing Table Entry on page 79 in *TIBCO Rendezvous Administration*

For the corresponding browser page, see Routers on page 144 in *TIBCO Rendezvous Administration*

RvrdProxy.removeRouter()

Method

Related Forms RvrdProxy.removeRouters()

Declaration

```
RvrdProxy removeRouter(
    java.lang.String routerName)
  throws ConfigurationException

RvrdProxy removeRouters(
    java.lang.String[] routerNames)
  throws ConfigurationException
```

Purpose Remove router names from the routing daemon.

Remarks When removing more than one router name, the second method is faster than repeatedly calling the first method.

These methods return the `RvrdProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`routerName`	Remove this router name.
`routerNames`	Remove all the router names in this array.

See Also

`RvrdProxy.addRouter()` on page 79
`RvrdProxy.getRouter()` on page 81
`Router` on page 121

For background information, see Routing Table Entry on page 79 in *TIBCO Rendezvous Administration*

For the corresponding browser page, see Routers on page 144 in *TIBCO Rendezvous Administration*

RvrdProxy.setLoggingParams()

Method

Declaration

```
RvrdProxy setLoggingParams(
    boolean connections,
    boolean subjectInterest,
    boolean subjectData)
  throws ConfigurationException
```

Purpose Set the flags that guide the routing daemon's log output.

Remarks This method sets three boolean flags in the routing daemon. The program must supply all three values.

This method returns the `RvrdProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`connections`	When `true`, log connection activity whenever this routing daemon establishes or closes a connection to a neighbor.
`subjectInterest`	When `true`, log all subscription requests (notification of listening) that this routing daemon sends to its neighbors or receives from its neighbors.
`subjectData`	When `true`, log all messages that this routing daemon forwards to its neighbors or receives from its neighbors.

See Also

`RvrdProxy` on page 76
`RvrdProxy.getLoggingParams()` on page 80

For background information, see Interpreting Log Output on page 117 in *TIBCO Rendezvous Administration*

For the corresponding browser page, see Daemon Parameters on page 141 in *TIBCO Rendezvous Administration*, and Logging on page 143 in *TIBCO Rendezvous Administration*

ImportSubject

Class

Declaration

```
class com.tibco.tibrv.config.ImportSubject
  extends java.lang.Object
```

Purpose Represent an import subject.

Remarks Import subjects pair a subject name with an optional import weight value.

The method `LocalNetworkInterface.getImportSubjects()` returns an array of objects of this class.

Constant	Description
`DEFAULT_WEIGHT`	This constant specifies the default import weight of an import subject (10).

Method	Description	Page
`ImportSubject.getSubject()`	Get the import subject name.	85
`ImportSubject.getWeight()`	Get the import weight.	86

See Also `LocalNetworkInterface.getImportSubjects()` on page 95

For the corresponding browser pages, see Subject Gating on page 149 in *TIBCO Rendezvous Administration*, and the sections that follow it

For background information, see these sections:

- Subject Gating on page 81 in *TIBCO Rendezvous Administration*
- Load Balancing on page 91 in *TIBCO Rendezvous Administration*

ImportSubject.getSubject()

Method

Declaration `java.lang.String` **`getSubject`**`()`

Purpose Get the import subject name.

See Also `ImportSubject` on page 84

For the corresponding browser pages, see Subject Gating on page 149 in *TIBCO Rendezvous Administration*, and the sections that follow it

For background information, see Subject Gating on page 81 in *TIBCO Rendezvous Administration*

ImportSubject.getWeight()

Method

Declaration `int getWeight()`

Purpose Get the import weight.

See Also `ImportSubject` on page 84

For the corresponding browser pages, see Subject Gating on page 149 in *TIBCO Rendezvous Administration*, and the sections that follow it

For background information, see these sections:

- Subject Gating on page 81 in *TIBCO Rendezvous Administration*
- Load Balancing on page 91 in *TIBCO Rendezvous Administration*

LocalNetworkInterface

Class

Declaration

```
class com.tibco.tibrv.config.LocalNetworkInterface
  extends java.lang.Object
```

Purpose Represent a local network interface of a router.

Remarks The method `Router.getLocalNetworkInterfaces()` returns an array of objects of this class.

Constant	Description
`LocalNetworkInterface.DEFAULT_COST`	Default path cost (1) between a local network and a router. To use the default cost, supply this constant to `Router.addLocalNetworkInterface()` on page 128.
`LocalNetworkInterface.UNSPECIFIED`	To use default values for the service parameter of a local network interface, supply this constant to `Router.addLocalNetworkInterface()` on page 128.

(Sheet 1 of 2)

Method	Description	Page
`LocalNetworkInterface.addExportSubject()`	Permit the router to export a subject from this local network.	90
`LocalNetworkInterface.addImportSubject()`	Permit the router to import a subject into this local network.	91
`LocalNetworkInterface.addSubject()`	Permit the router to import and export a subject.	92
`LocalNetworkInterface.getCost()`	Get the path cost for routing between the local network and its router.	93

(Sheet 2 of 2)

Method	Description	Page
`LocalNetworkInterface.getExportSubjects()`	Get the subjects that the router may export from this local network.	94
`LocalNetworkInterface.getImportSubjects()`	Get the subjects that the router may import into this local network.	95
`LocalNetworkInterface.getName()`	Get the unique name of the local network.	96
`LocalNetworkInterface.getNetwork()`	Get the network specification of the local network.	97
`LocalNetworkInterface.getService()`	Get the UDP service of the local network.	98
`LocalNetworkInterface.removeExportSubject()` `LocalNetworkInterface.removeExportSubjects()`	Stop exporting a subject from this local network.	99
`LocalNetworkInterface.removeImportSubject()` `LocalNetworkInterface.removeImportSubjects()`	Stop importing a subject from this local network.	100
`LocalNetworkInterface.removeSubject()` `LocalNetworkInterface.removeSubjects()`	Stop importing and exporting a subject.	101
`LocalNetworkInterface.toXml()`	Format the local network information as an XML document.	102

See Also `Router.addLocalNetworkInterface()` on page 128
`Router.getLocalNetworkInterfaces()` on page 136

For the corresponding browser pages, see Local Network Interfaces Configuration on page 147 in *TIBCO Rendezvous Administration*, and the sections that follow it

For background information, see Local Network on page 80 in *TIBCO Rendezvous Administration*

LocalNetworkInterface.addExportSubject()

Method

Declaration

```
void AddExportSubject(
    java.lang.String  subject)
  throws ConfigurationException
```

Purpose Permit the router to export a subject from this local network.

Parameter	Description
`subject`	Add this subject for export.

See Also

`LocalNetworkInterface` on page 87
`LocalNetworkInterface.getExportSubjects()` on page 94
`LocalNetworkInterface.removeExportSubject()` on page 99

For the corresponding browser page, see Subject Gating on page 149 in *TIBCO Rendezvous Administration*

For background information, see Subject Gating on page 81 in *TIBCO Rendezvous Administration*.

LocalNetworkInterface.addImportSubject()

Method

Declaration

```
void addImportSubject(
    java.lang.String  subject)
  throws ConfigurationException

void addImportSubject(
    java.lang.String  subject,
    int               weight)
  throws ConfigurationException
```

Purpose Permit the router to import a subject into this local network.

Parameter	Description
`subject`	Add this subject for import.
`weight`	When present, add the subject with this weight. When absent, add the subject with the default import weight (10). For background information, see Load Balancing on page 91 in *TIBCO Rendezvous Administration*.

See Also

`LocalNetworkInterface` on page 87
`LocalNetworkInterface.getImportSubjects()` on page 95
`LocalNetworkInterface.removeImportSubject()` on page 100

For the corresponding browser page, see Subject Gating on page 149 in *TIBCO Rendezvous Administration*

For background information, see Subject Gating on page 81 in *TIBCO Rendezvous Administration*.

LocalNetworkInterface.addSubject()

Method

Declaration

```
void addSubject(
    java.lang.String  subject)
  throws ConfigurationException

void addSubject(
    java.lang.String  subject,
    int               weight)
  throws ConfigurationException
```

Purpose Permit the router to import and export a subject.

Parameter	Description
`subject`	Add this subject for import and export.
`weight`	When present, add the subject with this weight for import. When absent, add the subject with the default import weight (10). For background information, see Load Balancing on page 91 in *TIBCO Rendezvous Administration*.

See Also

`LocalNetworkInterface` on page 87
`LocalNetworkInterface.getExportSubjects()` on page 94
`LocalNetworkInterface.getImportSubjects()` on page 95
`LocalNetworkInterface.removeSubject()` on page 101

For the corresponding browser page, see Subject Gating on page 149 in *TIBCO Rendezvous Administration*

For background information, see Subject Gating on page 81 in *TIBCO Rendezvous Administration*.

LocalNetworkInterface.getCost()

Method

Declaration `int getCost()`

Purpose Get the path cost for routing between the local network and its router.

See Also `LocalNetworkInterface` on page 87
`Router.addLocalNetworkInterface()` on page 128

For the corresponding browser page, see Local Network Interfaces Configuration on page 147 in *TIBCO Rendezvous Administration*

For background information, see Load Balancing on page 91 in *TIBCO Rendezvous Administration*.

LocalNetworkInterface.getExportSubjects()

Method

Declaration

```
java.lang.String[] getExportSubjects()
  throws ConfigurationException
```

Purpose

Get the subjects that the router may export from this local network.

See Also

LocalNetworkInterface on page 87
LocalNetworkInterface.addExportSubject() on page 90
LocalNetworkInterface.removeExportSubject() on page 99

For the corresponding browser page, see Subject Gating on page 149 in *TIBCO Rendezvous Administration*

For background information, see Subject Gating on page 81 in *TIBCO Rendezvous Administration*.

LocalNetworkInterface.getImportSubjects()

Method

Declaration

```
ImportSubject[] getImportSubjects()
  throws ConfigurationException
```

Purpose

Get the subjects that the router may import into this local network.

See Also

ImportSubject on page 84
LocalNetworkInterface on page 87
LocalNetworkInterface.addImportSubject() on page 91
LocalNetworkInterface.removeImportSubject() on page 100

For the corresponding browser page, see Subject Gating on page 149 in *TIBCO Rendezvous Administration*

For background information, see Subject Gating on page 81 in *TIBCO Rendezvous Administration*.

LocalNetworkInterface.getName()

Method

Declaration `java.lang.String` **`getName`**`()`

Purpose Get the unique name of the local network.

See Also `LocalNetworkInterface` on page 87
`Router.addLocalNetworkInterface()` on page 128

For the corresponding browser page, see Local Network Interfaces Configuration on page 147 in *TIBCO Rendezvous Administration*

For background information, see Local Network Name on page 80 in *TIBCO Rendezvous Administration*.

LocalNetworkInterface.getNetwork()

Method

Declaration `java.lang.String` **`getNetwork`**`()`

Purpose Get the network specification of the local network.

See Also `LocalNetworkInterface` on page 87
`Router.addLocalNetworkInterface()` on page 128

For the corresponding browser page, see Local Network Interfaces Configuration on page 147 in *TIBCO Rendezvous Administration*

For background information, see Network and Service on page 80 in *TIBCO Rendezvous Administration*.

LocalNetworkInterface.getService()

Method

Declaration

```
int getService()
```

Purpose

Get the UDP service of the local network.

See Also

`LocalNetworkInterface` on page 87

`Router.addLocalNetworkInterface()` on page 128

For the corresponding browser page, see Local Network Interfaces Configuration on page 147 in *TIBCO Rendezvous Administration*

For background information, see Network and Service on page 80 in *TIBCO Rendezvous Administration*.

LocalNetworkInterface.removeExportSubject()

Method

Related Forms LocalNetworkInterface.removeExportSubjects()

Declaration

```
void removeExportSubject(
    java.lang.String  subject)
  throws ConfigurationException

void removeExportSubjects(
    java.lang.String[]  subjects)
  throws ConfigurationException
```

Purpose Stop exporting a subject from this local network.

Parameter	Description
`subject`	Remove this export subject.
`subjects`	Remove these export subjects.

See Also

`LocalNetworkInterface` on page 87
`LocalNetworkInterface.addExportSubject()` on page 90
`LocalNetworkInterface.getExportSubjects()` on page 94

For the corresponding browser page, see Subject Gating on page 149 in *TIBCO Rendezvous Administration*

For background information, see Subject Gating on page 81 in *TIBCO Rendezvous Administration*.

LocalNetworkInterface.removeImportSubject()

Method

Related Forms LocalNetworkInterface.removeImportSubjects()

Declaration

```
void removeImportSubject(
    java.lang.String  subject)
  throws ConfigurationException

void removeImportSubjects(
    java.lang.String[]  subjects)
  throws ConfigurationException
```

Purpose Stop importing a subject from this local network.

Remarks This method removes the subject from the import list. If the subject is not on the list, the method returns without exception.

Parameter	Description
`subject`	Remove this import subject.
`subjects`	Remove these import subjects.

See Also

`LocalNetworkInterface` on page 87
`LocalNetworkInterface.addImportSubject()` on page 91
`LocalNetworkInterface.getImportSubjects()` on page 95

For the corresponding browser page, see Subject Gating on page 149 in *TIBCO Rendezvous Administration*

For background information, see Subject Gating on page 81 in *TIBCO Rendezvous Administration*.

LocalNetworkInterface.removeSubject()

Method

Related Forms LocalNetworkInterface.removeSubjects()

Declaration

```
void removeSubject(
    java.lang.String  subject)
  throws ConfigurationException

void removeSubjects(
    java.lang.String[]  subjects)
  throws ConfigurationException
```

Purpose Stop importing and exporting a subject.

Parameter	Description
`subject`	Remove this subject.
`subjects`	Remove these subjects.

See Also

`LocalNetworkInterface` on page 87
`LocalNetworkInterface.addSubject()` on page 92
`LocalNetworkInterface.getExportSubjects()` on page 94
`LocalNetworkInterface.getImportSubjects()` on page 95

For the corresponding browser page, see Subject Gating on page 149 in *TIBCO Rendezvous Administration*

For background information, see Subject Gating on page 81 in *TIBCO Rendezvous Administration*.

LocalNetworkInterface.toXml()

Method

Declaration `java.lang.String toXml()`

Purpose Format the local network information as an XML document.

See Also `LocalNetworkInterface` on page 87

For the corresponding browser page, see Local Network Interfaces Configuration on page 147 in *TIBCO Rendezvous Administration*

LoggingParams

Class

Declaration

```
class com.tibco.tibrv.config.LoggingParams
  extends java.lang.Object
```

Purpose Encapsulate logging parameters from Rendezvous routing daemons.

Method	Description	Page
`LoggingParams.connections()`	Extract a flag that reflects logging for neighbor connections.	104
`LoggingParams.getAsMap()`	Format the logging parameters as a map.	105
`LoggingParams.subjectData()`	Extract a flag that reflects logging for subject data (forwarded messages).	106
`LoggingParams.subjectInterest()`	Extract a flag that reflects logging for subject interest (subscription requests).	107

Remarks Instances of this class are read-only objects. Methods do not interact with the component.

`RvrdProxy.getLoggingParams()` returns instances of this class.

See Also Read-Only Objects on page 5

`RvrdProxy.getLoggingParams()` on page 80

LoggingParams.connections()

Method

Declaration `boolean connections()`

Purpose Extract a flag that reflects logging for neighbor connections.

Remarks This method does not interact with the component.

LoggingParams.getAsMap()

Method

Declaration `java.util.Map` **`getAsMap`**`()`

Purpose Format the logging parameters as a map.

Remarks The resulting map is useful for iterative methods, such as printing.

This method does not interact with the component.

LoggingParams.subjectData()

Method

Declaration `boolean subjectData()`

Purpose Extract a flag that reflects logging for subject data (forwarded messages).

Remarks This method does not interact with the component.

LoggingParams.subjectInterest()

Method

Declaration `boolean subjectInterest()`

Purpose Extract a flag that reflects logging for subject interest (subscription requests).

Remarks This method does not interact with the component.

NeighborInterface

Class

Declaration

```
class com.tibco.tibrv.config.NeighborInterface
  extends java.lang.Object
```

Purpose Represent a neighbor interface of a router.

Remarks The method `Router.getNeighborInterfaces()` returns an array of objects of this class.

Constant	Description

`NeighborInterface.ACCEPT_ANY`
`NeighborInterface.ACTIVE`
`NeighborInterface.PASSIVE`
`NeighborInterface.SEEK_ANY`
	`NeighborInterface.getType()` returns these integer constants, which denote the four types of neighbor interfaces.

`NeighborInterface.DEFAULT_COST`	Default path cost between two routers. To use the default cost, supply this constant to `Router.addLocalNetworkInterface()` on page 128.
`NeighborInterface.DEFAULT_PORT`	Default TCP port (7501). To use the default port, supply this constant to any of the four methods of `Router` that add neighbor interfaces.

(Sheet 1 of 2)

Method	Description	Page
`NeighborInterface.getBacklog()`	Get the size of the current backlog on the neighbor link.	110
`NeighborInterface.getCost()`	Get the path cost of the neighbor link.	111
`NeighborInterface.getId()`	Get the interface ID of the neighbor interface.	112
`NeighborInterface.getLocalPort()`	Get the TCP connect port of the local endpoint.	113
`NeighborInterface.getNeighborHost()`	Get the host of the remote endpoint.	114
`NeighborInterface.getNeighborName()`	Get the router name of the remote endpoint.	115

(Sheet 2 of 2)

Method	Description	Page
`NeighborInterface.getNeighborPort()`	Get the TCP connect port of the remote endpoint.	116
`NeighborInterface.getType()`	Get the type of the neighbor interface.	117
`NeighborInterface.isEncrypted()`	Get the TLS requirement flag of the neighbor interface.	119
`NeighborInterface.toXml()`	Format the neighbor interface information as an XML document.	120

See Also

`Router.addAcceptAnyInterface()` on page 123
`Router.addActiveInterface()` on page 125
`Router.addPassiveInterface()` on page 130
`Router.addSeekAnyInterface()` on page 133
`Router.getNeighborInterfaces()` on page 139

For the corresponding browser pages, see Neighbor Interfaces on page 153 in *TIBCO Rendezvous Administration*, and the sections that follow it

For background information, see these sections:

- Neighbors on page 85 in *TIBCO Rendezvous Administration*
- Adding Neighbor Interfaces on page 87 in *TIBCO Rendezvous Administration*

NeighborInterface.getBacklog()

Method

Declaration `long` **`getBacklog`**`()`

Purpose Get the size of the current backlog on the neighbor link.

Remarks Backlog is the set of messages waiting for transfer from one router to a neighboring router. This call returns a snapshot size (in bytes) of the backlog—that is, the sum of the sizes of all waiting messages.

See Also `NeighborInterface` on page 108

For the corresponding browser page, see Router Connection Statistics on page 137 in *TIBCO Rendezvous Administration*

NeighborInterface.getCost()

Method

Declaration `int getCost()`

Purpose Get the path cost of the neighbor link.

See Also `NeighborInterface` on page 108

For the corresponding browser page, see Neighbor Interfaces on page 153 in *TIBCO Rendezvous Administration*

For background information, see Load Balancing on page 91 in *TIBCO Rendezvous Administration*

NeighborInterface.getId()

Method

Declaration `java.lang.String` **`getId`**`()`

Purpose Get the interface ID of the neighbor interface.

See Also `NeighborInterface` on page 108

For the corresponding browser page, see Neighbor Interfaces on page 153 in *TIBCO Rendezvous Administration*

NeighborInterface.getLocalPort()

Method

Declaration `int getLocalPort()`

Purpose Get the TCP connect port of the local endpoint.

See Also `NeighborInterface` on page 108

For the corresponding browser page, see Neighbor Interfaces on page 153 in *TIBCO Rendezvous Administration*

For background information, see Local Connect Port on page 85 in *TIBCO Rendezvous Administration*

NeighborInterface.getNeighborHost()

Method

Declaration `java.lang.String` **`getNeighborHost`**`()`

Purpose Get the host of the remote endpoint.

Remarks This method can return either the hostname or the IP address of the remote neighbor.

See Also `NeighborInterface` on page 108

For the corresponding browser page, see Neighbor Interfaces on page 153 in *TIBCO Rendezvous Administration*

For background information, see Remote Host on page 86 in *TIBCO Rendezvous Administration*

NeighborInterface.getNeighborName()

Method

Declaration `java.lang.String` **`getNeighborName`**`()`

Purpose Get the router name of the remote endpoint.

See Also `NeighborInterface` on page 108

For the corresponding browser page, see Neighbor Interfaces on page 153 in *TIBCO Rendezvous Administration*

For background information, see Remote Router Name on page 85 in *TIBCO Rendezvous Administration*

NeighborInterface.getNeighborPort()

Method

Declaration `int getNeighborPort()`

Purpose Get the TCP connect port of the remote endpoint.

See Also `NeighborInterface` on page 108

For the corresponding browser page, see Neighbor Interfaces on page 153 in *TIBCO Rendezvous Administration*

For background information, see Remote Connect Port on page 86 in *TIBCO Rendezvous Administration*

NeighborInterface.getType()

Method

Declaration `int getType()`

Purpose Get the type of the neighbor interface.

Remarks This method returns an integer constant denoting the type of the neighbor interface. For a table of constant values, see `NeighborInterface` on page 108.

See Also `NeighborInterface` on page 108

For the corresponding browser page, see Neighbor Interfaces on page 153 in *TIBCO Rendezvous Administration*

For background information, see Adding Neighbor Interfaces on page 87 in *TIBCO Rendezvous Administration*

NeighborInterface.isCompressed()

Method

Declaration

```
boolean isCompressed()
  throws java.lang.UnsupportedOperationException
```

Purpose

Get the data compression flag of the neighbor interface.

Remarks

This method returns a value that reflects communications between the local router and its neighbor (which the `NeighborInterface` object describes):

- `true` indicates that this neighbor interface compresses outbound data and uncompresses inbound data.
- `false` indicates that this neighbor interface does *not* compress data.

Notice that it is inconsistent for one neighbor to compress data while the other does not. Neighbors must agree concerning compression, otherwise they cannot establish a connection.

Rendezvous routing daemons support data compression in release 7.1 and later; earlier releases do not. When the routing daemon does not support data compression (as with `rvrd` from release 7.0), this method throws java.lang.UnsupportedOperationException.

See Also

`NeighborInterface` on page 108
`Router.addAcceptAnyInterface()` on page 123
`Router.addActiveInterface()` on page 125
`Router.addPassiveInterface()` on page 130
`Router.addSeekAnyInterface()` on page 133

For the corresponding browser page, see Neighbor Interfaces on page 153 in *TIBCO Rendezvous Administration*

For background information, see Data Compression on page 86 in *TIBCO Rendezvous Administration*

NeighborInterface.isEncrypted()

Method

Declaration `boolean isEncrypted()`

Purpose Get the TLS requirement flag of the neighbor interface.

Remarks This method returns a value that reflects communications between the local router and its neighbor (which the `NeighborInterface` object describes):

- `true` specifies that the two neighbors communicate using TLS protocols.
- `false` specifies that the two neighbors communicate using non-secure protocols.

See Also

`NeighborInterface` on page 108
`Router.addActiveInterface()` on page 125
`Router.addPassiveInterface()` on page 130

For the corresponding browser page, see Neighbor Interfaces on page 153 in *TIBCO Rendezvous Administration*

For background information, see Adding Neighbor Interfaces on page 87 in *TIBCO Rendezvous Administration*

NeighborInterface.toXml()

Method

Declaration `java.lang.String toXml()`

Purpose Format the neighbor interface information as an XML document.

See Also `NeighborInterface` on page 108

For the corresponding browser page, see Neighbor Interfaces on page 153 in *TIBCO Rendezvous Administration*

Router

Class

Declaration

```
class com.tibco.tibrv.config.Router
  extends java.lang.Object
```

Purpose Represent a router interface.

Remarks The method `RvrdProxy.getRouter()` returns an object of this class (or its subclass, `BorderRouter`).

(Sheet 1 of 2)

Method	Description	Page
`Router.addAcceptAnyInterface()`	Specify a neighbor interface that accepts any neighbor.	123
`Router.addActiveInterface()`	Specify a neighbor interface that actively connects with its neighbor.	125
`Router.addLocalNetworkInterface()`	Specify a local network interface.	128
`Router.addPassiveInterface()`	Specify a neighbor interface that passively accepts connections from its neighbor.	130
`Router.addSeekAnyInterface()`	Specify a neighbor interface that seeks any neighbor.	133
`Router.clearMaxBacklog()`	Disable protection against large backlog.	135
`Router.getLocalNetworkInterfaces()`	Get the local network interfaces of the router.	136
`Router.getMaxBacklog()`	Get the maximum backlog (in kilobytes).	137
`Router.getName()`	Get the name of the router.	138
`Router.getNeighborInterfaces()`	Get the neighbor interfaces of the router.	139
`Router.removeLocalNetworkInterface()` `Router.removeLocalNetworkInterfaces()`	Remove local network interfaces from the router.	140
`Router.removeNeighborInterface()` `Router.removeNeighborInterfaces()`	Remove neighbor interfaces from the router.	141

(Sheet 2 of 2)

Method	Description	Page
`Router.setMaxBacklog()`	Set the maximum backlog, and enable protection against large backlog.	142
`Router.toXml()`	Format the router information as an XML document.	143

See Also

`RvrdProxy.addRouter()` on page 79
`RvrdProxy.getRouter()` on page 81

For the corresponding browser pages, see Routers on page 144 in *TIBCO Rendezvous Administration*, and the sections that follow it

For background information, see Routing Table Entry on page 79 in *TIBCO Rendezvous Administration*

Router.addAcceptAnyInterface()

Method

Declaration

```
NeighborInterface addAcceptAnyInterface(
    java.lang.String  localHost,
    int               localPort,
    int               cost,
    boolean           compressed)
  throws ConfigurationException

NeighborInterface addAcceptAnyInterface(
    int       localPort,
    int       cost,
    boolean   compressed)
  throws ConfigurationException

NeighborInterface addAcceptAnyInterface(
    int localPort,
    int cost)
  throws ConfigurationException
```

Use the first form when the router is release 7.2 or later.

Use the second form when the router is release 7.1 (it is deprecated for later releases).

Use the third form when the router is release 7.0.

Purpose

Specify a neighbor interface that accepts any neighbor.

Remarks

Use this method to specify a neighbor interface in which this routing daemon accepts neighbor connections from any other routing daemon.

It is not possible to configure a router name with more than one *accept any* neighbor interface.

Accept any interfaces cannot use TLS neighbor connections.

(Sheet 1 of 2)

Parameter	Description
`localHost`	The local router will listen on this network interface for neighbor connection requests from remote routers. Supply an IP address or hostname denoting a local network interface. For more information, see Local Host on page 85 in *TIBCO Rendezvous Administration*.

(Sheet 2 of 2)

Parameter	Description
`localPort`	The local router will listen on this local TCP port for neighbor connection requests from remote routers. For more information, see Local Connect Port on page 85 in *TIBCO Rendezvous Administration*.
`cost`	The path cost of this neighbor link; see Load Balancing on page 91 in *TIBCO Rendezvous Administration*. You may supply the default cost, `NeighborInterface.DEFAULT_COST`.
`compressed`	When `true`, the new neighbor interface compresses outbound data and uncompresses inbound data. When `false`, the new neighbor interface does *not* compress data. Notice that it is inconsistent for one neighbor to compress data while the other does not. Neighbors must agree concerning compression, otherwise they cannot establish a connection. Rendezvous routing daemons support data compression in release 7.1 and later; earlier releases do not. When the routing daemon does not support data compression, use the form of this method that omits this parameter.

See Also

`NeighborInterface` on page 108
`Router` on page 121
`Router.getNeighborInterfaces()` on page 139
`Router.removeNeighborInterface()` on page 141

For the corresponding browser pages, see Add New Neighbor Interface on page 155 in *TIBCO Rendezvous Administration*

For background information, see these sections:

- Neighbors on page 85 in *TIBCO Rendezvous Administration*
- Accept Any as Neighbor on page 88 in *TIBCO Rendezvous Administration*

Router.addActiveInterface()

Method

Declaration

```
NeighborInterface addActiveInterface(
    java.lang.String  localHost,
    int               localPort
    java.lang.String  remoteHost,
    int               remotePort,
    java.lang.String  neighborName,
    int               cost,
    boolean           compressed,
    boolean           encrypted,
    java.lang.String  peerCertificate)
  throws ConfigurationException

NeighborInterface addActiveInterface(
    int               localPort
    java.lang.String  remoteHost,
    int               remotePort,
    java.lang.String  neighborName,
    int               cost,
    boolean           compressed,
    boolean           encrypted,
    java.lang.String  peerCertificate)
  throws ConfigurationException

NeighborInterface addActiveInterface(
    int               localPort
    java.lang.String  remoteHost,
    int               remotePort,
    java.lang.String  neighborName,
    int               cost,
    boolean           encrypted,
    java.lang.String  peerCertificate)
  throws ConfigurationException
```

Use the first form when the router is release 7.2 or later.

Use the second form when the router is release 7.1 (it is deprecated for later releases).

Use the third form when the router is release 7.0.

Purpose

Specify a neighbor interface that actively connects with its neighbor.

Remarks

Use this method to specify a neighbor interface in which the local router actively attempts to connect to the remote neighbor.

On a `BorderRouter`, this method automatically configures the default policy for all pairings of the new interface with every other existing interface. The default policy forwards `_INBOX.>` (all inbox subjects); for additional details, see Policy on page 106 in *TIBCO Rendezvous Administration*.

(Sheet 1 of 2)

Parameter	Description
`localHost`	The local router will listen on this network interface for neighbor connection requests from remote routers. Supply an IP address or hostname denoting a local network interface. For more information, see Local Host on page 85 in *TIBCO Rendezvous Administration*.
`localPort`	The local router will listen on this local TCP port for neighbor connection requests from its remote neighbor. For more information, see Local Connect Port on page 85 in *TIBCO Rendezvous Administration*.
`remoteHost`	The local router will seek its neighbor running on this host computer. Supply either a resolvable hostname, or the IP address of the computer in a remote network where the neighboring daemon is running. For more information, see Remote Connection Information on page 85 in *TIBCO Rendezvous Administration*.
`remotePort`	The local router will use this TCP port to request a neighbor connection with the remote router. The remote router must listen for neighbor requests on this port. For more information, see Remote Connection Information on page 85 in *TIBCO Rendezvous Administration*.
`neighborName`	The local router will connect only with this remote router.
`cost`	The path cost of this neighbor link; see Load Balancing on page 91 in *TIBCO Rendezvous Administration*. You may supply the default cost, `NeighborInterface.DEFAULT_COST`.
`compressed`	When `true`, the new neighbor interface compresses outbound data and uncompresses inbound data. When `false`, the new neighbor interface does *not* compress data. Notice that it is inconsistent for one neighbor to compress data while the other does not. Neighbors must agree concerning compression, otherwise they cannot establish a connection. Rendezvous routing daemons support data compression in release 7.1 and later; earlier releases do not. When the routing daemon does not support data compression, use the form of this method that omits this parameter.

(Sheet 2 of 2)

Parameter	Description
`encrypted`	When `true`, the two neighbors must communicate using TLS protocols. When `false`, the two neighbors must communicate using non-secure protocols. Notice that both neighbors must use the same protocols, otherwise they cannot establish a connection.
`peerCertificate`	In TLS protocols, the local router expects the remote router to present this certificate as evidence of its identity. Supply the text of the public certificate (in PEM encoding).

See Also

`NeighborInterface` on page 108
`Router` on page 121
`Router.getNeighborInterfaces()` on page 139
`Router.removeNeighborInterface()` on page 141

For the corresponding browser pages, see Add New Neighbor Interface on page 155 in *TIBCO Rendezvous Administration*

For background information, see these sections:

- Neighbors on page 85 in *TIBCO Rendezvous Administration*
- Active Neighbor on page 87 in *TIBCO Rendezvous Administration*

Router.addLocalNetworkInterface()

Method

Declaration

```
LocalNetworkInterface addLocalNetworkInterface(
    java.lang.String  localNetworkName,
    int               service,
    java.lang.String  networkSpecification,
    int               cost)
  throws ConfigurationException
```

Purpose Specify a local network interface.

Remarks On a `BorderRouter`, this method automatically configures the default policy for all pairings of the new interface with every other existing interface. The default policy forwards `_INBOX.>` (all inbox subjects); for additional details, see Policy on page 106 in *TIBCO Rendezvous Administration*.

Item	Description
`localNetworkName`	Supply the name of the local network. Local network names must be globally unique. For more information, see Local Network on page 80 in *TIBCO Rendezvous Administration*.
`service`	Supply the UDP service for communication on the local network. Programs within the local network communicate using this service. For more information, see Specifying the UDP Service on page 15 in *TIBCO Rendezvous Administration*. You may supply the default value `LocalNetworkInterface.UNSPECIFIED`.
`networkSpecification`	Supply the network specification. For more information, see Constructing the Network Parameter on page 17 in *TIBCO Rendezvous Administration*. You may supply the default value, the empty string.
`cost`	Supply the path cost for routing between the local network and the router. For more information, see Load Balancing on page 91 in *TIBCO Rendezvous Administration*. You may supply the default cost `LocalNetworkInterface.DEFAULT_COST`.

See Also `LocalNetworkInterface` on page 87

`Router` on page 121
`Router.getLocalNetworkInterfaces()` on page 136
`Router.removeLocalNetworkInterface()` on page 140

For the corresponding browser page, see Local Network Interfaces Configuration on page 147 in *TIBCO Rendezvous Administration*

For background information, see these sections:

- Routing Table Entry on page 79 in *TIBCO Rendezvous Administration*
- Local Network on page 80 in *TIBCO Rendezvous Administration*

Router.addPassiveInterface()

Method

Declaration

```
NeighborInterface addPassiveInterface(
    java.lang.String  localHost,
    int               localPort
    java.lang.String  neighborName,
    int               cost,
    boolean           compressed,
    boolean           encrypted,
    java.lang.String  peerCertificate)
  throws ConfigurationException

NeighborInterface addPassiveInterface(
    int               localPort
    java.lang.String  neighborName,
    int               cost,
    boolean           compressed,
    boolean           encrypted,
    java.lang.String  peerCertificate)
  throws ConfigurationException

NeighborInterface addPassiveInterface(
    int               localPort
    java.lang.String  neighborName,
    int               cost,
    boolean           encrypted,
    java.lang.String  peerCertificate)
  throws ConfigurationException
```

Use the first form when the router is release 7.2 or later.

Use the second form when the router is release 7.1 (it is deprecated for later releases).

Use the third form when the router is release 7.0.

Purpose Specify a neighbor interface that passively accepts connections from its neighbor.

Remarks Use this method to specify a neighbor interface in which the local router does not actively attempt to connect to the remote neighbor. Instead, it passively waits for the remote neighbor to request a connection.

On a `BorderRouter`, this method automatically configures the default policy for all pairings of the new interface with every other existing interface. The default policy forwards `_INBOX.>` (all inbox subjects); for additional details, see Policy on page 106 in *TIBCO Rendezvous Administration*.

Parameter	Description
`localHost`	The local router will listen on this network interface for neighbor connection requests from remote routers. Supply an IP address or hostname denoting a local network interface. For more information, see Local Host on page 85 in *TIBCO Rendezvous Administration*.
`localPort`	The local router will listen on this local TCP port for neighbor connection requests from its remote neighbor. For more information, see Local Connect Port on page 85 in *TIBCO Rendezvous Administration*.
`neighborName`	The local router will passively accept neighbor connections only from this remote router.
`cost`	The path cost of this neighbor link; see Load Balancing on page 91 in *TIBCO Rendezvous Administration*. You may supply the default cost, `NeighborInterface.DEFAULT_COST`.
`compressed`	When `true`, the new neighbor interface compresses outbound data and uncompresses inbound data. When `false`, the new neighbor interface does *not* compress data. Notice that it is inconsistent for one neighbor to compress data while the other does not. Neighbors must agree concerning compression, otherwise they cannot establish a connection. Rendezvous routing daemons support data compression in release 7.1 and later; earlier releases do not. When the routing daemon does not support data compression, use the form of this method that omits this parameter.
`encrypted`	When `true`, the two neighbors must communicate using TLS protocols. When `false`, the two neighbors must communicate using non-secure protocols. Notice that both neighbors must use the same protocols, otherwise they cannot establish a connection.
`peerCertificate`	In TLS protocols, the local router expects the remote router to present this certificate as evidence of its identity. Supply the text of the public certificate (in PEM encoding).

See Also

`NeighborInterface` on page 108
`Router` on page 121
`Router.getNeighborInterfaces()` on page 139
`Router.removeNeighborInterface()` on page 141

For the corresponding browser pages, see Add New Neighbor Interface on page 155 in *TIBCO Rendezvous Administration*

For background information, see these sections:

- Neighbors on page 85 in *TIBCO Rendezvous Administration*
- Passive Neighbor on page 87 in *TIBCO Rendezvous Administration*

Router.addSeekAnyInterface()

Method

Declaration

```
NeighborInterface addSeekAnyInterface(
    java.lang.String  remoteHost,
    int               remotePort,
    int               cost,
    boolean           compressed)
  throws ConfigurationException

NeighborInterface addSeekAnyInterface(
    java.lang.String  remoteHost,
    int               remotePort,
    int               cost)
  throws ConfigurationException
```

Use the first form when the router is release 7.1 or later.

Use the second form when the router is release 7.0.

Purpose

Specify a neighbor interface that seeks any neighbor.

Remarks

Use this method to specify a neighbor interface in which this routing daemon attempts to connect to any remote routing daemon that matches the specification.

It is illegal to configure a router name with two or more *seek any* neighbor interfaces with the same remote host.

Seek any interfaces cannot use TLS neighbor connections.

Parameter	Description
`remoteHost`	The local router will seek a neighbor running on this host computer. Supply either a *DNS* hostname that can resolve to more than one IP address, or a *virtual* IP address.
`remotePort`	The local router will use this TCP port to request a neighbor connection with remote routers. All potential neighbors must listen for neighbor requests on this port. For more information, see Remote Connection Information on page 85 in *TIBCO Rendezvous Administration*.
`cost`	The path cost of this neighbor link; see Load Balancing on page 91 in *TIBCO Rendezvous Administration*. You may supply the default cost, `NeighborInterface.DEFAULT_COST`.

Parameter	Description
`compressed`	When `true`, the new neighbor interface compresses outbound data and uncompresses inbound data. When `false`, the new neighbor interface does *not* compress data. Notice that it is inconsistent for one neighbor to compress data while the other does not. Neighbors must agree concerning compression, otherwise they cannot establish a connection. Rendezvous routing daemons support data compression in release 7.1 and later; earlier releases do not. When the routing daemon does not support data compression, use the form of this method that omits this parameter.

See Also

`NeighborInterface` on page 108
`Router` on page 121
`Router.getNeighborInterfaces()` on page 139
`Router.removeNeighborInterface()` on page 141

For the corresponding browser pages, see Add New Neighbor Interface on page 155 in *TIBCO Rendezvous Administration*

For background information, see these sections:

- Neighbors on page 85 in *TIBCO Rendezvous Administration*
- Seek Neighbor with Any Name on page 88 in *TIBCO Rendezvous Administration*

Router.clearMaxBacklog()

Method

Declaration

```
Router clearMaxBacklog()
  throws ConfigurationException
```

Purpose Disable protection against large backlog.

Remarks This method returns the `Router` object, so programs can conveniently chain additional method calls to the return value.

See Also

`Router` on page 121
`Router.getMaxBacklog()` on page 137
`Router.setMaxBacklog()` on page 142

For the corresponding browser page, see Routers on page 144 in *TIBCO Rendezvous Administration*

For background information, see Backlog Protection on page 115 in *TIBCO Rendezvous Administration*

Router.getLocalNetworkInterfaces()

Method

Declaration

```
LocalNetworkInterface[] getLocalNetworkInterfaces()
  throws ConfigurationException
```

Purpose

Get the local network interfaces of the router.

See Also

`LocalNetworkInterface` on page 87
`Router` on page 121
`Router.addLocalNetworkInterface()` on page 128
`Router.removeLocalNetworkInterface()` on page 140

For the corresponding browser page, see Local Network Interfaces Configuration on page 147 in *TIBCO Rendezvous Administration*

For background information, see these sections:

- Routing Table Entry on page 79 in *TIBCO Rendezvous Administration*
- Local Network on page 80 in *TIBCO Rendezvous Administration*

Router.getMaxBacklog()

Method

Declaration

```
int getMaxBacklog()
  throws ConfigurationException
```

Purpose Get the maximum backlog (in kilobytes).

Remarks Zero indicates that the backlog protection feature is disabled (see `Router.clearMaxBacklog()` on page 135).

See Also

`Router` on page 121
`Router.clearMaxBacklog()` on page 135
`Router.setMaxBacklog()` on page 142

For the corresponding browser page, see Routers on page 144 in *TIBCO Rendezvous Administration*

For background information, see Backlog Protection on page 115 in *TIBCO Rendezvous Administration*

Router.getName()

Method

Declaration `java.lang.String getName()`

Purpose Get the name of the router.

See Also

`Router` on page 121
`Router.clearMaxBacklog()` on page 135
`Router.setMaxBacklog()` on page 142

For the corresponding browser page, see Routers on page 144 in *TIBCO Rendezvous Administration*

For background information, see Router Name on page 79 in *TIBCO Rendezvous Administration*

Router.getNeighborInterfaces()

Method

Declaration

```
NeighborInterface[] getNeighborInterfaces()
  throws ConfigurationException
```

Purpose

Get the neighbor interfaces of the router.

See Also

`LocalNetworkInterface` on page 87
`Router` on page 121
`Router.addAcceptAnyInterface()` on page 123
`Router.addActiveInterface()` on page 125
`Router.addPassiveInterface()` on page 130
`Router.addSeekAnyInterface()` on page 133
`Router.removeNeighborInterface()` on page 141

For the corresponding browser page, see Neighbor Interfaces on page 153 in *TIBCO Rendezvous Administration*

For background information, see Neighbors on page 85 in *TIBCO Rendezvous Administration*

Router.removeLocalNetworkInterface()

Method

Related Forms Router.removeLocalNetworkInterfaces()

Declaration

```
Router removeLocalNetworkInterface(
    java.lang.String  localNetworkName)
  throws ConfigurationException

Router removeLocalNetworkInterfaces(
    java.lang.String[]  localNetworkNames)
  throws ConfigurationException
```

Purpose Remove local network interfaces from the router.

Remarks When removing more than network interface, the second method is faster than repeatedly calling the first method.

These methods return the `Router` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`localNetworkName`	Remove the network interface with this name.
`localNetworkNames`	Remove all the network interfaces named in this array.

See Also

`LocalNetworkInterface` on page 87
`Router` on page 121
`Router.addLocalNetworkInterface()` on page 128
`Router.getLocalNetworkInterfaces()` on page 136

For the corresponding browser page, see Local Network Interfaces Configuration on page 147 in *TIBCO Rendezvous Administration*

For background information, see these sections:

- Routing Table Entry on page 79 in *TIBCO Rendezvous Administration*
- Local Network on page 80 in *TIBCO Rendezvous Administration*

Router.removeNeighborInterface()

Method

Related Forms Router.removeNeighborInterfaces()

Declaration

```
Router removeNeighborInterface(
    java.lang.String  interfaceId)
  throws ConfigurationException

Router removeNeighborInterfaces(
    java.lang.String[]  interfaceIds)
  throws ConfigurationException
```

Purpose Remove neighbor interfaces from the router.

Remarks When removing more than neighbor interface, the second method is faster than repeatedly calling the first method.

These methods return the `Router` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`interfaceId`	Remove the neighbor interface with this ID.
`interfaceIds`	Remove all the neighbor interfaces specified in this array.

See Also

`LocalNetworkInterface` on page 87
`Router` on page 121
`Router.addAcceptAnyInterface()` on page 123
`Router.addActiveInterface()` on page 125
`Router.addPassiveInterface()` on page 130
`Router.addSeekAnyInterface()` on page 133
`Router.getNeighborInterfaces()` on page 139

For the corresponding browser page, see Neighbor Interfaces on page 153 in *TIBCO Rendezvous Administration*

For background information, see Neighbors on page 85 in *TIBCO Rendezvous Administration*

Router.setMaxBacklog()

Method

Declaration

```
Router setMaxBacklog(
    int  maxBacklog)
  throws ConfigurationException
```

Purpose Set the maximum backlog, and enable protection against large backlog.

Remarks This method returns the `Router` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`maxBacklog`	Use this maximum (in kilobytes) to limit router backlog.

See Also

`Router` on page 121
`Router.clearMaxBacklog()` on page 135
`Router.getMaxBacklog()` on page 137

For the corresponding browser page, see Routers on page 144 in *TIBCO Rendezvous Administration*

For background information, see Backlog Protection on page 115 in *TIBCO Rendezvous Administration*

Router.toXml()

Method

Declaration `java.lang.String` **`toXml`**`()`

Purpose Format the router information as an XML document.

See Also `Router` on page 121

For the corresponding browser page, see Routers on page 144 in *TIBCO Rendezvous Administration*

BorderRouter

Class

Declaration

```
class com.tibco.tibrv.config.BorderRouter
  extends Router
```

Purpose Represent a border router interface.

Remarks The method `RvrdProxy.getRouter()` can return an object of this class.

Once a border router is configured, you cannot remove it, rename it, nor convert it to a first-tier router. Nor can you convert a first-tier router to a border router.

Method	Description	Page
`BorderRouter.addPolicyRule()`	Add a policy rule for a pair of interfaces.	146
`BorderRouter.getPolicyRule()` `BorderRouter.getPolicyRules()`	Get policy rules of a border router.	148
`BorderRouter.removePolicyRule()`	Remove a policy rule from the border router.	149
`BorderRouter.toXml()`	Format the border router information as an XML document.	150

Inherited Methods

`Router.addAcceptAnyInterface()`
`Router.addActiveInterface()`
`Router.addLocalNetworkInterface()`
`Router.addPassiveInterface()`
`Router.addSeekAnyInterface()`
`Router.clearMaxBacklog()`
`Router.getLocalNetworkInterfaces()`
`Router.getMaxBacklog()`
`Router.getName()`
`Router.getNeighborInterfaces()`
`Router.removeLocalNetworkInterface()`
`Router.removeLocalNetworkInterfaces()`
`Router.removeNeighborInterface()`
`Router.removeNeighborInterfaces()`
`Router.setMaxBacklog()`
`Router.toXml()` override

See Also

`RvrdProxy.addBorderRouter()` on page 78
`RvrdProxy.getRouter()` on page 81
`PolicyRule` on page 151

For the corresponding browser pages, see Border Routing on page 145 in *TIBCO Rendezvous Administration*, and the sections that follow it

For background information, see Border Routing on page 105 in *TIBCO Rendezvous Administration*

BorderRouter.addPolicyRule()

Method

Declaration

```
PolicyRule addPolicyRule(
    java.lang.String    fromInterface,
    java.lang.String    toInterface,
    java.lang.String    subject,
    boolean             firstBorder)
  throws ConfigurationException

PolicyRule addPolicyRule(
    java.lang.String    fromInterface,
    java.lang.String    toInterface,
    java.lang.String[]  subjects,
    boolean             firstBorder)
  throws ConfigurationException
```

Purpose Add a policy rule for a pair of interfaces.

Remarks Use the first form to add only one subject. Use the second form to add several subjects with one method call. (The first form calls the second form with an array of one element.)

If a policy rule already exists for the From/To pair, then this method (first form) behaves like `PolicyRule.addAllowedSubject()` on page 152—it attempts to add the new subject to the existing rule.

- If the subject does not already exist in the rule, then the method adds it.
- If the subject already exists in the rule, then the method throws a `ConfigurationException`; the rule and subject remain unchanged.

(Sheet 1 of 2)

Parameter	Description
fromInterface	Supply the From-Interface for the new policy rule.
toInterface	Supply the To-Interface for the new policy rule.
subject	Specify a subject for which this rule permits forwarding across the two interfaces.
subjects	Specify an array of subjects for which this rule permits forwarding across the two interfaces.

(Sheet 2 of 2)

Parameter	Description
firstBorder	When `true`, the rule instructs the border router to forward the subjects only when a message has not yet crossed another border. When `false`, the border router always forwards the subjects. This property applies to all the subjects that this method adds. It does not affect pre-existing subjects.

See Also

`BorderRouter` on page 144
`BorderRouter.getPolicyRule()` on page 148
`BorderRouter.removePolicyRule()` on page 149
`PolicyRule` on page 151

For the corresponding browser pages, see Border Routing on page 145 in *TIBCO Rendezvous Administration*

For background information, see Border Routing on page 105 in *TIBCO Rendezvous Administration*

BorderRouter.getPolicyRule()

Method

Related Forms BorderRouter.getPolicyRules()

Declaration

```
PolicyRule getPolicyRule(
    java.lang.String  fromInterface,
    java.lang.String  toInterface)
  throws ConfigurationException

PolicyRule[] getPolicyRules()
  throws ConfigurationException
```

Purpose Get policy rules of a border router.

Remarks Use the first form to get the rule for a specific pairing of From- and To-Interfaces.

Use the second form to get all policy rules of the border router.

Even if you have not configured a policy rule for a From/To pair, a default rule might still exist. Default rules allow the subject `_INBOX.>` (with `firstBorder false`).

After a rule has been removed, getting the rule for that pair returns an empty rule—that is, a rule without any allowed subjects.

Parameter	Description
fromInterface	Supply the From-Interface of the rule to get.
toInterface	Supply the To-Interface of the rule to get.

See Also

`BorderRouter` on page 144
`BorderRouter.addPolicyRule()` on page 146
`BorderRouter.removePolicyRule()` on page 149
`PolicyRule` on page 151

For the corresponding browser pages, see Border Routing on page 145 in *TIBCO Rendezvous Administration*

For background information, see Border Routing on page 105 in *TIBCO Rendezvous Administration*

BorderRouter.removePolicyRule()

Method

Declaration

```
Router removePolicyRule(
    java.lang.String  fromInterface,
    java.lang.String  toInterface)
  throws ConfigurationException
```

Purpose Remove a policy rule from the border router.

Remarks This method removes a policy rule. As a result, the border router no longer forwards any of the subjects that the rule had specified.

Without explicit configuration, border routers forward `_INBOX.>` (all inbox subjects). Whenever you add an interface, `rvrd` automatically configures this default policy for every pairing of that interface with every other existing interface. Removing a policy rule explicitly removes this subject, which disables forwarding of inbox messages for the From/To pair.

If no rule exists for the From/To pair—that is, even the default rule has been removed—then this method returns normally (it does not throw an exception).

Parameter	Description
fromInterface	Supply the From-Interface of the rule to remove.
toInterface	Supply the To-Interface of the rule to remove.

See Also

`LocalNetworkInterface` on page 87
`Router` on page 121
`Router.addLocalNetworkInterface()` on page 128
`Router.getLocalNetworkInterfaces()` on page 136

For the corresponding browser pages, see Border Routing on page 145 in *TIBCO Rendezvous Administration*

For background information, see Border Routing on page 105 in *TIBCO Rendezvous Administration*

BorderRouter.toXml()

Method

Declaration

```
java.lang.String toXml()
  throws ConfigurationException
```

Purpose Format the border router information as an XML document.

See Also BorderRouter on page 144

For the corresponding browser pages, see Border Routing on page 145 in *TIBCO Rendezvous Administration*

PolicyRule

Class

Declaration

```
class com.tibco.tibrv.config.PolicyRule
  extends java.lang.Object
```

Purpose Represent a policy rule of a border router.

Remarks The method `BorderRouter.getPolicyRule()` returns objects of this class.

Method	Description	Page
`PolicyRule.addAllowedSubject()` `PolicyRule.addAllowedSubjects()`	Update an existing policy rule by adding one or more subjects.	152
`PolicyRule.getAllowedSubjects()`	Get subjects from an existing policy rule.	154
`PolicyRule.getBorderRouterName()`	Get the name of the border router to which the policy rule pertains.	155
`PolicyRule.getFromInterface()`	Get the From-Interface of a policy rule.	156
`PolicyRule.getToInterface()`	Get the To-Interface of a policy rule.	157
`PolicyRule.removeAllowedSubject()` `PolicyRule.removeAllowedSubjects()`	Update an existing policy rule by removing one or more subjects.	158
`PolicyRule.toXml()`	Format the policy rule information as an XML document.	159

See Also

`BorderRouter` on page 144
`BorderRouter.getPolicyRule()` on page 148
`PolicyRule` on page 151

For the corresponding browser pages, see Border Policy on page 151 in *TIBCO Rendezvous Administration*

For background information, see Border Routing on page 105 in *TIBCO Rendezvous Administration*

PolicyRule.addAllowedSubject()

Method

Related Forms PolicyRule.addAllowedSubjects()

Declaration

```
PolicyRule addAllowedSubject(
    java.lang.String  subject,
    boolean           firstBorder)
  throws ConfigurationException

PolicyRule addAllowedSubjects(
    java.lang.String[]  subjects,
    boolean             firstBorder)
  throws ConfigurationException
```

Purpose Update an existing policy rule by adding one or more subjects.

Remarks This method returns the updated policy rule.

The first form adds only one subject. The second form adds several subjects. (The first form calls the second form with an array of one element.)

If the subject does not already exist in the rule, then the method adds it.

If the subject already exists in the rule, then the method throws a `ConfigurationException`; the rule and subject remain unchanged.

Parameter	Description
subject	Append a subject to the rule, so the border router forwards that subject.
subjects	Append an array of subjects to the rule, so the border router forwards those subjects.
firstBorder	When `true`, the rule instructs the border router to forward the new subjects only when a message has not yet crossed another border. When `false`, the border router always forwards the new subjects. This property applies to all the new subjects that this method adds. It does not affect pre-existing subjects.

See Also `BorderRouter.getPolicyRule()` on page 148
`PolicyRule` on page 151

For the corresponding browser pages, see Border Policy on page 151 in *TIBCO Rendezvous Administration*

For background information, see Border Routing on page 105 in *TIBCO Rendezvous Administration*

PolicyRule.getAllowedSubjects()

Method

Declaration

```
java.lang.String[] getAllowedSubjects(
    boolean           firstBorder)
  throws ConfigurationException
```

Purpose Get subjects from an existing policy rule.

Parameter	Description
firstBorder	This method returns an array of all the subjects in the rule for which the first-border property matches this argument. To get all the rule's subjects, you must call this method twice—once with `true`, and once with `false`.

See Also

`BorderRouter.getPolicyRule()` on page 148
`PolicyRule` on page 151

For the corresponding browser pages, see Border Policy on page 151 in *TIBCO Rendezvous Administration*

For background information, see Border Routing on page 105 in *TIBCO Rendezvous Administration*

PolicyRule.getBorderRouterName()

Method

Declaration `java.lang.String` **`getBorderRouterName`**`()`

Purpose Get the name of the border router to which the policy rule pertains.

See Also `BorderRouter` on page 144
`PolicyRule` on page 151

For background information, see Border Routing on page 105 in *TIBCO Rendezvous Administration*

PolicyRule.getFromInterface()

Method

Declaration `java.lang.String` **`getFromInterface`**`()`

Purpose Get the From-Interface of a policy rule.

See Also `PolicyRule` on page 151

For background information, see Border Routing on page 105 in *TIBCO Rendezvous Administration*

PolicyRule.getToInterface()

Method

Declaration `java.lang.String` **`getToInterface`**`()`

Purpose Get the To-Interface of a policy rule.

See Also `PolicyRule` on page 151

For background information, see Border Routing on page 105 in *TIBCO Rendezvous Administration*

PolicyRule.removeAllowedSubject()

Method

Related Forms PolicyRule.removeAllowedSubjects()

Declaration

```
PolicyRule removeAllowedSubject(
    java.lang.String  subject)
  throws ConfigurationException

PolicyRule removeAllowedSubjects(
    java.lang.String[]  subjects)
  throws ConfigurationException
```

Purpose Update an existing policy rule by removing one or more subjects.

Remarks This method returns the updated policy rule.

The first form removes only one subject. The second form removes several subjects.

Removing a subject that is not in the rule results in a `ConfigurationException`.

Without explicit configuration, border routers forward `_INBOX.>` (all inbox subjects). Whenever you add an interface, `rvrd` automatically configures this default policy for every pairing of that interface with every other existing interface. You may explicitly remove this subject to disable forwarding of inbox messages.

Parameter	Description
subject	Remove a subject from the rule, so the border router no longer forwards that subject.
subjects	Remove an array of subjects from the rule, so the border router no longer forwards those subjects.

See Also

`BorderRouter.getPolicyRule()` on page 148
`PolicyRule` on page 151

For the corresponding browser pages, see Border Policy on page 151 in *TIBCO Rendezvous Administration*

For background information, see Border Routing on page 105 in *TIBCO Rendezvous Administration*

PolicyRule.toXml()

Method

Declaration

```
java.lang.String toXml()
  throws ConfigurationException
```

Purpose Format the policy rule information as an XML document.

See Also `PolicyRule` on page 151

For the corresponding browser pages, see Border Routing on page 105 in *TIBCO Rendezvous Administration*

Chapter 5 Security

This chapter describes the proxy interface for the Rendezvous components that require either passwords or certificates for security.

Topics

SecurityProxy

Interface

Declaration

```
interface com.tibco.tibrv.config.SecurityProxy
  extends DaemonProxy
```

Purpose Define methods for components that require passwords or certificates for security.

Constant	Description
These constants correspond to positions in the certificate usage bit vector; see `SecurityProxy.getValidUses()` on page 166.	
`SecurityProxy.HTTPS`	Represents usage of certificates in HTTPS protocols.
`SecurityProxy.ROUTERS_TO_ROUTERS`	Represents usage of certificates in TLS protocols among routing daemons.
`SecurityProxy.DAEMON_TO_CLIENTS`	Represents usage of certificates in TLS protocols with client transports of the daemon.

Method	Description	Page
`SecurityProxy.getAdministratorName()`	Get the administrator name.	164
`SecurityProxy.getCertificateSlot()` `SecurityProxy.getCertificateSlots()`	Get daemon certificate slots.	165
`SecurityProxy.getValidUses()`	Get the valid uses of certificates for this daemon.	166
`SecurityProxy.setCertificateUses()`	Assign certificates to each valid use for this daemon.	167
`SecurityProxy.setCredentials()`	Set administrator identification for the component.	168
`SecurityProxy.useCredentials()`	Record administrator identification for the configuration program.	169

Inherited Methods

`DaemonProxy.getComponentName()`
`DaemonProxy.getComponentInformation()`

`XmlSerializable.printXml()`

Components These components support this interface:

```
rvrd
rvsd
rvsrd
rvcache
```

See Also For information about the parameters that this interface can access, see these sections:

- Secure Daemons (rvsd and rvsrd) on page 163 in *TIBCO Rendezvous Administration*
- Browser Administration Interface—rvrd on page 128 in *TIBCO Rendezvous Administration*
- Browser Administration Interface—rvsd and rvsrd on page 186 in *TIBCO Rendezvous Administration*
- Current Value Cache on page 265 in *TIBCO Rendezvous Administration*

SecurityProxy.getAdministratorName()

Method

Declaration

```
java.lang.String getAdministratorName()
  throws ConfigurationException
```

Purpose

Get the administrator name.

See Also

SecurityProxy on page 162
SecurityProxy.setCredentials() on page 168
SecurityProxy.useCredentials() on page 169

For the corresponding browser page, see Administrator and Password on page 141 in *TIBCO Rendezvous Administration*

SecurityProxy.getCertificateSlot()

Method

Related Forms SecurityProxy.getCertificateSlots()

Declaration

```
CertificateSlot getCertificateSlot(
    int  certificateIndex)
  throws ConfigurationException

CertificateSlot[] getCertificateSlots()
  throws ConfigurationException
```

Purpose Get daemon certificate slots.

Remarks Component daemons have four slots in which they can store certificates. The first method gets a specific slot. The second method gets an array of all four slots.

Parameter	Description
`certificateIndex`	Get the certificate slot corresponding to this index. In Java, indexing is zero-based; in Rendezvous browser administration interfaces, certificate indexing is one-based. So to specify certificate slot #1, supply zero; to specify certificate slot #4, supply 3.

See Also

`SecurityProxy` on page 162
`CertificateSlot` on page 170

For the corresponding browser pages, see Certificate List on page 161 in *TIBCO Rendezvous Administration*, or Certificate List on page 204 in *TIBCO Rendezvous Administration*

SecurityProxy.getValidUses()

Method

Declaration

```
int getValidUses()
  throws ConfigurationException
```

Purpose Get the valid uses of certificates for this daemon.

Remarks Each daemon component that supports the `SecurityProxy` interface uses certificates in up to three ways. This method returns a bit vector that describes the legitimate uses of certificates for the actual component.

The three constants defined for `SecurityProxy` on page 162 correspond to positions in the bit vector. To determine whether a specific use applies to the component, probe the corresponding bit with Java's bitwise AND (`&`) operator; for example:

```
if ( (SecurityProxy) myDmnProxy.getValidUses() & SecurityProxy.HTTPS
     != 0 {
           // My daemon supports certificates for HTTPS.
           ... }
```

See Also

`SecurityProxy` on page 162

`SecurityProxy.setCertificateUses()` on page 167

For the corresponding browser pages, see Certificate Uses on page 159 in *TIBCO Rendezvous Administration*, or Certificate Uses on page 202 in *TIBCO Rendezvous Administration*

SecurityProxy.setCertificateUses()

Method

Declaration

```
SecurityProxy setCertificateUses(
    int  httpsCertificateIndex,
    int  routersToRoutersCertificateIndex,
    int  daemonToClientsCertificateIndex)
  throws ConfigurationException
```

Purpose Assign certificates to each valid use for this daemon.

Remarks Each daemon component that supports the `SecurityProxy` interface uses certificates in up to three ways. This method assigns one of the component's four possible certificates to each use that the component supports.

Supply a certificate index in each parameter position. This method ignores parameters that correspond to invalid uses for the daemon.

Component daemons can store up to four certificates. In Java, indexing is zero-based; in Rendezvous browser administration interfaces, certificate indexing is one-based. So to specify certificate #1, supply zero; to specify certificate #4, supply 3.

This method returns the `SecurityProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`httpsCertificateIndex`	Supply the index of the certificate to use for HTTPS communication (for example, with browsers or with Java configuration programs).
`routersToRoutersCertificateIndex`	Supply the index of the certificate to use for TLS communication with other routing daemons.
`daemonToClientsCertificateIndex`	Supply the index of the certificate to use for TLS communications with client transports.

See Also

`SecurityProxy` on page 162
`SecurityProxy.getValidUses()` on page 166

For the corresponding browser pages, see Certificate Uses on page 159 in *TIBCO Rendezvous Administration*, or Certificate Uses on page 202 in *TIBCO Rendezvous Administration*

SecurityProxy.setCredentials()

Method

Declaration

```
SecurityProxy setCredentials(
    java.lang.String  name,
    java.lang.String  password)
  throws ConfigurationException
```

Purpose Set administrator identification for the component.

Remarks This method returns the `SecurityProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`name`	Set the daemon to expect this administrator name.
`password`	Set the daemon to expect this administrator password.

See Also

`SecurityProxy` on page 162
`SecurityProxy.getAdministratorName()` on page 164
`SecurityProxy.useCredentials()` on page 169

For the corresponding browser page, see Administrator and Password on page 141 in *TIBCO Rendezvous Administration*

SecurityProxy.useCredentials()

Method

Declaration

```
void useCredentials(
    java.lang.String  name,
    java.lang.String  password)
  throws ConfigurationException
```

Purpose Record administrator identification for the configuration program.

Remarks This method records administrator identification credentials within your Java program. When the daemon component requests identification, the program automatically transmits these credentials to the daemon.

Of course, these credentials must match the credentials stored at the daemon, otherwise the daemon rejects them and closes the connection.

Parameter	Description
`name`	Set the program to offer this administrator name.
`password`	Set the program to offer this administrator password.

See Also

`SecurityProxy` on page 162
`SecurityProxy.getAdministratorName()` on page 164
`SecurityProxy.setCredentials()` on page 168

CertificateSlot

Class

Declaration

```
class com.tibco.tibrv.config.CertificateSlot
  extends java.lang.Object
```

Purpose Represent a slot in a daemon component that can store an X.509 certificate.

Remarks The method `SecurityProxy.getCertificateSlot()` and related methods return objects of this class.

Method	Description	Page
`CertificateSlot.getIndex()`	Get the index of this certificate slot.	171
`CertificateSlot.getPathname()`	Get the file name from which the component read the certificate in this slot.	172
`CertificateSlot.getText()`	Get the certificate data as a text string.	173
`CertificateSlot.getUses()`	Get the uses of this certificate within the component.	174
`CertificateSlot.setFromText()`	Interpret a text string as certificate data, and put the certificate in this slot.	176
`CertificateSlot.setFromFile()`	Read certificate data from a file, and put the certificate in this slot.	175
`CertificateSlot.toXml()`	Format the certificate slot information as an XML document.	177

See Also `SecurityProxy.getCertificateSlot()` on page 165

For the corresponding browser pages, see Certificate List on page 161 in *TIBCO Rendezvous Administration*, or Certificate List on page 204 in *TIBCO Rendezvous Administration*

For background information, see Certificates and Security on page 31 in *TIBCO Rendezvous Concepts*.

CertificateSlot.getIndex()

Method

Declaration `int getIndex()`

Purpose Get the index of this certificate slot.

Remarks Component daemons have four slots in which they can store certificates (some slots might be empty). In Java, indexing is zero-based; in Rendezvous browser administration interfaces, certificate indexing is one-based. So zero indicates certificate slot #1, and 3 indicates certificate slot #4.

See Also `CertificateSlot` on page 170

For the corresponding browser pages, see Certificate List on page 161 in *TIBCO Rendezvous Administration*, or Certificate List on page 204 in *TIBCO Rendezvous Administration*

CertificateSlot.getPathname()

Method

Declaration `java.lang.String` **`getPathname`**`()`

Purpose Get the file name from which the component read the certificate in this slot.

Remarks Components can obtain certificate data either from a file, or directly as a text string.

- When the certificate data came from a file, this method returns the file name.
- When the certificate data came directly as a text string, this method returns the string `N/A` (not applicable).

See Also `SecurityProxy` on page 162
`CertificateSlot.setFromFile()` on page 175

For the corresponding browser pages, see Certificate List on page 161 in *TIBCO Rendezvous Administration*, or Certificate List on page 204 in *TIBCO Rendezvous Administration*

CertificateSlot.getText()

Method

Declaration `java.lang.String` **`getText`**`()`

Purpose Get the certificate data as a text string.

Remarks When the certificate slot is empty, this method returns `null`.

See Also
`SecurityProxy` on page 162
`CertificateSlot.setFromFile()` on page 175
`CertificateSlot.setFromText()` on page 176

For the corresponding browser pages, see Certificate List on page 161 in *TIBCO Rendezvous Administration*, or Certificate List on page 204 in *TIBCO Rendezvous Administration*

CertificateSlot.getUses()

Method

Declaration

```
int getUses()
```

Purpose

Get the uses of this certificate within the component.

Remarks

Each daemon component that supports the `SecurityProxy` interface uses certificates in up to three ways. This method returns a bit vector that describes the set of ways that the component actually uses the certificate in this slot.

The three constants defined for `SecurityProxy` on page 162 correspond to positions in the bit vector. To determine whether a specific use applies to the certificate, probe the corresponding bit with Java's bitwise AND (`&`) operator; for example:

```
if ( myCert.getUses() & SecurityProxy.HTTPS
     != 0 {
           // My daemon uses this certificate for HTTPS.
           ... }
```

See Also

`SecurityProxy` on page 162

`SecurityProxy.setCertificateUses()` on page 167

For the corresponding browser pages, see Certificate Uses on page 159 in *TIBCO Rendezvous Administration*, or Certificate Uses on page 202 in *TIBCO Rendezvous Administration*

CertificateSlot.setFromFile()

Method

Declaration

```
CertificateSlot setFromFile(
    java.lang.String  pathname,
    java.lang.String  password)
  throws ConfigurationException
```

Purpose Read certificate data from a file, and put the certificate in this slot.

Remarks This method returns the `CertificateSlot` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`pathname`	Read encrypted certificate data from this file.
`password`	Decrypt the certificate data with this password.

See Also `SecurityProxy` on page 162

For the corresponding browser pages, see Certificate List on page 161 in *TIBCO Rendezvous Administration*, or Certificate List on page 204 in *TIBCO Rendezvous Administration*

CertificateSlot.setFromText()

Method

Declaration

```
CertificateSlot setFromText(
    java.lang.String  text,
    java.lang.String  password)
  throws ConfigurationException
```

Purpose Interpret a text string as certificate data, and put the certificate in this slot.

Remarks This method returns the `CertificateSlot` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`text`	Use this text as encrypted certificate data.
`password`	Decrypt the certificate data with this password.

See Also `SecurityProxy` on page 162

For the corresponding browser pages, see Certificate List on page 161 in *TIBCO Rendezvous Administration*, or Certificate List on page 204 in *TIBCO Rendezvous Administration*

CertificateSlot.toXml()

Method

Declaration `java.lang.String` **`toXml`**`()`

Purpose Format the certificate slot information as an XML document.

See Also `CertificateSlot` on page 170

Chapter 6 Secure Daemons—rvsd & rvsrd

This chapter describes the proxy interface for the Rendezvous secure daemons (`rvsd` and `rvsrd`), and the immediate-access data objects that support it.

Topics

See Also

- RvsdInformation, page 291
- RvsrdInformation, page 293

SecureDaemonProxy

Interface

Declaration

```
interface com.tibco.tibrv.config.SecureDaemonProxy
  extends DaemonProxy
```

Purpose Define methods for Rendezvous secure daemons.

(Sheet 1 of 2)

Method	Description	Page
SecureDaemonProxy.addUser() SecureDaemonProxy.addUsers()	Authorize a user to connect to the secure daemon.	183
SecureDaemonProxy.authorizeListen()	Authorize all users to listen to a subject.	184
SecureDaemonProxy.authorizeListenAndSend()	Authorize all users to listen and to send to a subject.	185
SecureDaemonProxy.authorizeNetworkAndService() SecureDaemonProxy.authorizeNetworksAndServices()	Authorize all users to communicate on a network and service pair.	186
SecureDaemonProxy.authorizeSend()	Authorize all users to send to a subject.	188
SecureDaemonProxy.getDefaultNetworkAndService()	Get the default network and service pair of the secure daemon.	189
SecureDaemonProxy.getListen()	Get the subjects authorized for listening.	190
SecureDaemonProxy.getNetworksAndServices()	Get the authorized network and service pairs of the secure daemon.	191
SecureDaemonProxy.getSend()	Get the subjects authorized for sending.	192
SecureDaemonProxy.getUser() SecureDaemonProxy.getUsers()	Get the authorized users of the secure daemon.	193

(Sheet 2 of 2)

Method	Description	Page
`SecureDaemonProxy.removeListen()`	Remove authorization to listen to a subject.	194
`SecureDaemonProxy.removeListenAndSend()`	Remove authorization to listen and send to a subject.	195
`SecureDaemonProxy.removeNetworkAndService()` `SecureDaemonProxy.removeNetworksAndServices()`	Remove authorization to communicate on a network and service pair.	196
`SecureDaemonProxy.removeSend()`	Remove authorization to send to a subject.	198
`SecureDaemonProxy.removeUser()` `SecureDaemonProxy.removeUsers()`	Remove a user of the secure daemon.	199
`SecureDaemonProxy.setDefaultNetworkAndService()`	Set the default network and service pair of the secure daemon.	200

Inherited Methods

`DaemonProxy.getComponentName()`
`DaemonProxy.getComponentInformation()`

`XmlSerializable.printXml()`

Components These components support this interface:

```
rvsd
rvsrd
```

See Also For information about the parameters that this interface can access, see these sections:

- Secure Daemons (rvsd and rvsrd) on page 163 in *TIBCO Rendezvous Administration*
- Browser Administration Interface—rvrd on page 128 in *TIBCO Rendezvous Administration*

- Browser Administration Interface—rvsd and rvsrd on page 186 in *TIBCO Rendezvous Administration*

SecureDaemonProxy.addUser()

Method

Related Form SecureDaemonProxy.addUsers()

Declaration

```
User addUser(
    java.lang.String  username)
  throws ConfigurationException

User[] addUsers(
    java.lang.String[]  usernames)
  throws ConfigurationException
```

Purpose Authorize a user to connect to the secure daemon.

Remarks After this method returns, you must configure the security credentials of the resulting `User` object.

When adding more than one user, the second method is faster than repeatedly calling the first method.

Parameter	Description
`username`	Add a new user with this name.
`usernames`	Add a new user for each name in this array.

See Also

`SecureDaemonProxy` on page 180
`SecureDaemonProxy.getUser()` on page 193
`SecureDaemonProxy.removeUser()` on page 199
`User` on page 205

For the corresponding browser pages, see Users on page 197 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*

SecureDaemonProxy.authorizeListen()

Method

Declaration

```
SecureDaemonProxy authorizeListen(
    java.lang.String  subject)
  throws ConfigurationException

SecureDaemonProxy authorizeListen(
    java.lang.String[]  subjects)
  throws ConfigurationException
```

Purpose Authorize all users to listen to a subject.

Remarks When authorizing more than one subject, the second method is faster than repeatedly calling the first method.

These methods return the `SecureDaemonProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`subject`	Authorize subscriptions to this subject.
`subjectS`	Authorize subscriptions to all the subjects in this array.

See Also

`SecureDaemonProxy` on page 180
`SecureDaemonProxy.authorizeListenAndSend()` on page 185
`SecureDaemonProxy.authorizeSend()` on page 188
`SecureDaemonProxy.getListen()` on page 190
`SecureDaemonProxy.removeListen()` on page 194

For the corresponding browser pages, see Authorize Subjects on page 200 in *TIBCO Rendezvous Administration*

For background information, see Subject Authorization on page 169 in *TIBCO Rendezvous Administration*

SecureDaemonProxy.authorizeListenAndSend()

Method

Declaration

```
SecureDaemonProxy authorizeListenAndSend(
    java.lang.String  subject)
  throws ConfigurationException

SecureDaemonProxy authorizeListenAndSend(
    java.lang.String[]  subjects)
  throws ConfigurationException
```

Purpose Authorize all users to listen and to send to a subject.

Remarks When authorizing more than one subject, the second method is faster than repeatedly calling the first method.

These methods return the `SecureDaemonProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`subject`	Authorize subscriptions and sending to this subject.
`subjectS`	Authorize subscriptions and sending to all the subjects in this array.

See Also

`SecureDaemonProxy` on page 180
`SecureDaemonProxy.authorizeListen()` on page 184
`SecureDaemonProxy.authorizeSend()` on page 188
`SecureDaemonProxy.removeListenAndSend()` on page 195

For the corresponding browser pages, see Authorize Subjects on page 200 in *TIBCO Rendezvous Administration*

For background information, see Subject Authorization on page 169 in *TIBCO Rendezvous Administration*

SecureDaemonProxy.authorizeNetworkAndService()

Method

Related Form SecureDaemonProxy.authorizeNetworksAndServices()

Declaration

```
SecureDaemonProxy authorizeNetworkAndService(
    java.lang.String  networkSpecification,
    int               servicePort)
  throws ConfigurationException

SecureDaemonProxy authorizeNetworksAndServices(
    java.lang.String[]  networksServices)
  throws ConfigurationException
```

Purpose Authorize all users to communicate on a network and service pair.

Remarks When authorizing more than one pairing of network and service, the second method is faster than repeatedly calling the first method.

These methods return the `SecureDaemonProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`networkSpecification`	Authorize communication on this network.
`servicePort`	Authorize communication on this UDP service.
`networksServices`	Authorize communication on each pairing of network and service in this array. Each element in the array is a string that combines the network parameter with a UDP service, separated by a colon; for example: `";225.1.1.1:5238"` To construct the two parts of these strings, see: • Constructing the Network Parameter on page 17 in *TIBCO Rendezvous Administration* • Specifying the UDP Service on page 15 in *TIBCO Rendezvous Administration*.

See Also

`SecureDaemonProxy` on page 180
`SecureDaemonProxy.getNetworksAndServices()` on page 191
`SecureDaemonProxy.removeNetworkAndService()` on page 196
`NetworkServicePair` on page 201

For the corresponding browser pages, see Authorize Network and Service Pairs on page 199 in *TIBCO Rendezvous Administration*

For background information, see Network and Service Authorization on page 169 in *TIBCO Rendezvous Administration*

SecureDaemonProxy.authorizeSend()

Method

Declaration

```
SecureDaemonProxy authorizeSend(
    java.lang.String  subject)
  throws ConfigurationException

SecureDaemonProxy authorizeSend(
    java.lang.String[]  subjects)
  throws ConfigurationException
```

Purpose Authorize all users to send to a subject.

Remarks When authorizing more than one subject, the second method is faster than repeatedly calling the first method.

These methods return the `SecureDaemonProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`subject`	Authorize sending to this subject.
`subjectS`	Authorize sending to all the subjects in this array.

See Also

`SecureDaemonProxy` on page 180
`SecureDaemonProxy.authorizeListen()` on page 184
`SecureDaemonProxy.authorizeListenAndSend()` on page 185
`SecureDaemonProxy.getSend()` on page 192
`SecureDaemonProxy.removeSend()` on page 198

For the corresponding browser pages, see Authorize Subjects on page 200 in *TIBCO Rendezvous Administration*

For background information, see Subject Authorization on page 169 in *TIBCO Rendezvous Administration*

SecureDaemonProxy.getDefaultNetworkAndService()

Method

Declaration

```
NetworkServicePair getDefaultNetworkAndService()
  throws ConfigurationException
```

Purpose

Get the default network and service pair of the secure daemon.

Remarks

When a client transport does not specify particular network and service parameters, it automatically communicates over this default network and service. For example, when the client program supplies null values for either of these parameters, the secure daemon supplies this pair as a default.

See Also

`SecureDaemonProxy` on page 180

`SecureDaemonProxy.setDefaultNetworkAndService()` on page 200

`NetworkServicePair` on page 201

For the corresponding browser pages, see Default Network and Service on page 174 in *TIBCO Rendezvous Administration*

For background information, see Network and Service Authorization on page 169 in *TIBCO Rendezvous Administration*

SecureDaemonProxy.getListen()

Method

Declaration

```
java.lang.String[] getListen()
  throws ConfigurationException
```

Purpose Get the subjects authorized for listening.

Remarks This method returns an array of subject names. All authenticated users can subscribe to any of the subjects in the array.

See Also

`SecureDaemonProxy` on page 180
`SecureDaemonProxy.authorizeListen()` on page 184
`SecureDaemonProxy.getSend()` on page 192
`SecureDaemonProxy.removeListen()` on page 194

For the corresponding browser pages, see Authorize Subjects on page 200 in *TIBCO Rendezvous Administration*

For background information, see Subject Authorization on page 169 in *TIBCO Rendezvous Administration*

SecureDaemonProxy.getNetworksAndServices()

Method

Declaration

```
NetworkServicePair[] getNetworksAndServices()
  throws ConfigurationException
```

Purpose

Get the authorized network and service pairs of the secure daemon.

Remarks

To convert the resulting array of `NetworkServicePair` objects to an array of strings (suitable as an argument to `SecureDaemonProxy.removeNetworkAndService()` on page 196), iterate through the array, applying `NetworkServicePair.toString()` to each element.

See Also

`SecureDaemonProxy` on page 180
`SecureDaemonProxy.authorizeNetworkAndService()` on page 186
`SecureDaemonProxy.removeNetworkAndService()` on page 196
`NetworkServicePair` on page 201

For the corresponding browser pages, see Authorize Network and Service Pairs on page 199 in *TIBCO Rendezvous Administration*

For background information, see Network and Service Authorization on page 169 in *TIBCO Rendezvous Administration*

SecureDaemonProxy.getSend()

Method

Declaration

```
java.lang.String[] getSend()
  throws ConfigurationException
```

Purpose Get the subjects authorized for sending.

Remarks This method returns an array of subject names. All authenticated users can send to any of the subjects in the array.

See Also

`SecureDaemonProxy` on page 180
`SecureDaemonProxy.authorizeSend()` on page 188
`SecureDaemonProxy.getListen()` on page 190
`SecureDaemonProxy.removeSend()` on page 198

For the corresponding browser pages, see Authorize Subjects on page 200 in *TIBCO Rendezvous Administration*

For background information, see Subject Authorization on page 169 in *TIBCO Rendezvous Administration*

SecureDaemonProxy.getUser()

Method

Related Form SecureDaemonProxy.getUsers()

Declaration

```
User getUser(
    java.lang.String  username)
  throws ConfigurationException

User[] getUsers()
  throws ConfigurationException
```

Purpose Get the authorized users of the secure daemon.

Remarks The first method finds a user by name. The second method gets a list of all users.

See Also

SecureDaemonProxy on page 180
SecureDaemonProxy.addUser() on page 183
SecureDaemonProxy.removeUser() on page 199
User on page 205

For the corresponding browser pages, see Users on page 197 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*

SecureDaemonProxy.removeListen()

Method

Declaration

```
SecureDaemonProxy removeListen(
    java.lang.String  subject)
  throws ConfigurationException

SecureDaemonProxy removeListen(
    java.lang.String[]  subjects)
  throws ConfigurationException
```

Purpose Remove authorization to listen to a subject.

Remarks When removing more than one subject, the second method is faster than repeatedly calling the first method.

These methods return the `SecureDaemonProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`subject`	Remove authorization to listen to this subject.
`subjectS`	Remove authorization to listen to the subjects in this array.

See Also

`SecureDaemonProxy` on page 180
`SecureDaemonProxy.authorizeListen()` on page 184
`SecureDaemonProxy.getListen()` on page 190
`SecureDaemonProxy.removeListenAndSend()` on page 195
`SecureDaemonProxy.removeSend()` on page 198

For the corresponding browser pages, see Authorize Subjects on page 200 in *TIBCO Rendezvous Administration*

For background information, see Subject Authorization on page 169 in *TIBCO Rendezvous Administration*

SecureDaemonProxy.removeListenAndSend()

Method

Declaration

```
SecureDaemonProxy removeListenAndSend(
    java.lang.String  subject)
  throws ConfigurationException

SecureDaemonProxy removeListenAndSend(
    java.lang.String[]  subjects)
  throws ConfigurationException
```

Purpose Remove authorization to listen and send to a subject.

Remarks When removing more than one subject, the second method is faster than repeatedly calling the first method.

These methods return the `SecureDaemonProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`subject`	Remove authorization to listen and send to this subject.
`subjectS`	Remove authorization to listen and send to the subjects in this array.

See Also

`SecureDaemonProxy` on page 180
`SecureDaemonProxy.authorizeListenAndSend()` on page 185
`SecureDaemonProxy.removeListen()` on page 194
`SecureDaemonProxy.removeSend()` on page 198

For the corresponding browser pages, see Authorize Subjects on page 200 in *TIBCO Rendezvous Administration*

For background information, see Subject Authorization on page 169 in *TIBCO Rendezvous Administration*

SecureDaemonProxy.removeNetworkAndService()

Method

Related Form SecureDaemonProxy.removeNetworksAndServices()

Declaration

```
SecureDaemonProxy removeNetworkAndService(
    java.lang.String  networkSpecification,
    int               servicePort)
  throws ConfigurationException

SecureDaemonProxy removeNetworksAndServices(
    java.lang.String[]  networksServices)
  throws ConfigurationException
```

Purpose Remove authorization to communicate on a network and service pair.

Remarks When removing more than one pairing of network and service, the second method is faster than repeatedly calling the first method.

These methods return the `SecureDaemonProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`networkSpecification`	Remove authorization for this network.
`servicePort`	Remove authorization for this UDP service.
`networksServices`	Remove authorization for each pairing of network and service in this array. Each element in the array is a string that combines the network parameter with a UDP service; for example: `";225.1.1.1:5238"` To construct the two parts of these strings, see: • Constructing the Network Parameter on page 17 in *TIBCO Rendezvous Administration* • Specifying the UDP Service on page 15 in *TIBCO Rendezvous Administration*.

See Also

`SecureDaemonProxy` on page 180
`SecureDaemonProxy.authorizeNetworkAndService()` on page 186
`SecureDaemonProxy.getNetworksAndServices()` on page 191
`NetworkServicePair` on page 201

For the corresponding browser pages, see Authorize Network and Service Pairs on page 199 in *TIBCO Rendezvous Administration*

For background information, see Network and Service Authorization on page 169 in *TIBCO Rendezvous Administration*

SecureDaemonProxy.removeSend()

Method

Declaration

```
SecureDaemonProxy removeSend(
    java.lang.String  subject)
  throws ConfigurationException

SecureDaemonProxy removeSend(
    java.lang.String[]  subjects)
  throws ConfigurationException
```

Purpose Remove authorization to send to a subject.

Remarks When removing more than one subject, the second method is faster than repeatedly calling the first method.

These methods return the `SecureDaemonProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`subject`	Remove authorization to send to this subject.
`subjectS`	Remove authorization to send to the subjects in this array.

See Also

`SecureDaemonProxy` on page 180
`SecureDaemonProxy.authorizeSend()` on page 188
`SecureDaemonProxy.getSend()` on page 192
`SecureDaemonProxy.removeListen()` on page 194
`SecureDaemonProxy.removeListenAndSend()` on page 195

For the corresponding browser pages, see Authorize Subjects on page 200 in *TIBCO Rendezvous Administration*

For background information, see Subject Authorization on page 169 in *TIBCO Rendezvous Administration*

SecureDaemonProxy.removeUser()

Method

Related Form SecureDaemonProxy.removeUsers()

Declaration

```
SecureDaemonProxy removeUser(
    java.lang.String  username)
  throws ConfigurationException

SecureDaemonProxy[] removeUsers(
    java.lang.String[]  usernames)
  throws ConfigurationException
```

Purpose Remove a user of the secure daemon.

Remarks Removing a user prevents the user from connecting to the secure daemon.

When removing more than one user, the second method is faster than repeatedly calling the first method.

If this method throws an exception while removing more than one user, your code must handle the exception by checking the remaining users carefully. When the exception interrupted the call, some items might have been removed, while other items might not yet have been removed.

These methods return the `SecureDaemonProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`username`	Remove a user with this name.
`usernames`	Remove all the users named in this array.

See Also

`SecureDaemonProxy` on page 180
`SecureDaemonProxy.addUser()` on page 183
`SecureDaemonProxy.getUser()` on page 193

For the corresponding browser pages, see Users on page 197 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*

SecureDaemonProxy.setDefaultNetworkAndService()

Method

Declaration

```
SecureDaemonProxy setDefaultNetworkAndService(
    java.lang.String  networkSpecification,
    int               servicePort)
  throws ConfigurationException
```

Purpose Set the default network and service pair of the secure daemon.

Remarks When a client transport does not specify particular network and service parameters, it automatically communicates over this default network and service. For example, when the client program supplies null values for either of these parameters, the secure daemon supplies this pair as a default.

This method returns the `SecureDaemonProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`networkSpecification`	Use this network as the default.
`servicePort`	Use this UDP service as the default.

See Also

`SecureDaemonProxy` on page 180

`SecureDaemonProxy.getDefaultNetworkAndService()` on page 189

`NetworkServicePair` on page 201

For the corresponding browser pages, see Default Network and Service on page 174 in *TIBCO Rendezvous Administration*

For background information, see Network and Service Authorization on page 169 in *TIBCO Rendezvous Administration*

NetworkServicePair

Class

Declaration

```
class com.tibco.tibrv.config.NetworkServicePair
  extends java.lang.Object
```

Purpose Represent an network and service pair in a daemon component.

Remarks The method `SecureDaemonProxy.getNetworksAndServices()` and related methods return objects of this class.

Constant	Description
`NetworkServicePair.UNSPECIFIED`	No value is set for this parameter.

Method	Description	Page
`NetworkServicePair.getNetwork()`	Get the network.	202
`NetworkServicePair.getService()`	Get the UDP service.	203
`NetworkServicePair.toXml()`	Format the network and service pair as an XML document.	204

Inherited Methods

```
java.lang.Object.equals()
java.lang.Object.getClass()
java.lang.Object.hashCode()
java.lang.Object.notify()
java.lang.Object.notifyAll()
java.lang.Object.toString() override
java.lang.Object.wait()
```

See Also

`SecureDaemonProxy.getDefaultNetworkAndService()` on page 189
`SecureDaemonProxy.getNetworksAndServices()` on page 191

For the corresponding browser pages, see Authorize Network and Service Pairs on page 199 in *TIBCO Rendezvous Administration*

For background information, see Limiting Access on page 169 in *TIBCO Rendezvous Administration*.

NetworkServicePair.getNetwork()

Method

Declaration `java.lang.String` **`getNetwork`**`()`

Purpose Get the network.

See Also `NetworkServicePair` on page 201

For the corresponding browser pages, see Authorize Network and Service Pairs on page 199 in *TIBCO Rendezvous Administration*

For background information, see Limiting Access on page 169 in *TIBCO Rendezvous Administration*.

NetworkServicePair.getService()

Method

Declaration `int getService()`

Purpose Get the UDP service.

See Also `NetworkServicePair` on page 201

For the corresponding browser pages, see Authorize Network and Service Pairs on page 199 in *TIBCO Rendezvous Administration*

For background information, see Limiting Access on page 169 in *TIBCO Rendezvous Administration*.

NetworkServicePair.toXml()

Method

Declaration `java.lang.String` **`toXml`**`()`

Purpose Format the network and service pair as an XML document.

See Also `NetworkServicePair` on page 201

For the corresponding browser pages, see Authorize Network and Service Pairs on page 199 in *TIBCO Rendezvous Administration*

For background information, see Limiting Access on page 169 in *TIBCO Rendezvous Administration*.

User

Class

Declaration

```
class com.tibco.tibrv.config.User
  extends java.lang.Object
```

Purpose Represent a user record in a secure daemon component.

Remarks The method `SecureDaemonProxy.getUser()` and related methods return objects of this class.

For security, you cannot get an existing password from a `User` object; however, you may clear it and set a new one.

Method	Description	Page
`User.addCertificateFromFile()`	Read user certificate data from a PEM file.	206
`User.addCertificateFromText()`	Add a user certificate.	207
`User.addCertificateFromPKCS12File()`	Read user certificate data from a PKCS #12 file.	208
`User.clearPassword()`	Clear the user's password.	209
`User.getCertificates()`	Get all public certificates of the user.	210
`User.getUsername()`	Get the username of the user.	211
`User.removeCertificate()` `User.removeCertificates()`	Remove a user certificate.	212
`User.setPassword()`	Set a user's password.	213
`User.toXml()`	Format the user information as an XML document.	214

See Also

`SecureDaemonProxy.addUser()` on page 183
`SecureDaemonProxy.getUser()` on page 193

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

User.addCertificateFromFile()

Method

Declaration

```
UserCertificate addCertificateFromFile(
    java.lang.String  pathname)
  throws ConfigurationException
```

Purpose Read user certificate data from a PEM file.

Parameter	Description
`pathname`	Read certificate data from this file. The file must contain a public certificate in PEM encoding.

See Also

`User` on page 205
`User.addCertificateFromText()` on page 207
`User.getCertificates()` on page 210
`User.removeCertificate()` on page 212
`UserCertificate` on page 215

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

User.addCertificateFromText()

Method

Declaration

```
UserCertificate addCertificateFromText(
    java.lang.String  PEM_data)
  throws ConfigurationException
```

Purpose Add a user certificate.

Parameter	Description
`PEM_data`	Add a new user certificate corresponding to this data. The data must specify a public certificate in PEM encoding.

See Also

`User` on page 205
`User.addCertificateFromFile()` on page 206
`User.getCertificates()` on page 210
`User.removeCertificate()` on page 212
`UserCertificate` on page 215

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

User.addCertificateFromPKCS12File()

Method

Declaration

```
UserCertificate addCertificateFromPKCS12File(
    java.lang.String  pathname,
    java.lang.String  password)
  throws ConfigurationException
```

Purpose Read user certificate data from a PKCS #12 file.

Parameter	Description
`pathname`	Read certificate data from this file. The file must contain a public certificate in PKCS #12 format.
`password`	Supply a password to decode the PKCS #12 file.

See Also

`User` on page 205
`User.addCertificateFromFile()` on page 206
`User.getCertificates()` on page 210
`User.removeCertificate()` on page 212
`UserCertificate` on page 215

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

User.clearPassword()

Method

Declaration

```
User clearPassword()
  throws ConfigurationException
```

Purpose Clear the user's password.

Remarks This method returns the `User` object, so programs can conveniently chain additional method calls to the return value.

See Also

`User` on page 205

`User.setPassword()` on page 213

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

User.getCertificates()

Method

Declaration

```
UserCertificate[] getCertificates()
  throws ConfigurationException
```

Purpose

Get all public certificates of the user.

See Also

`User` on page 205
`User.addCertificateFromFile()` on page 206
`User.addCertificateFromText()` on page 207
`User.removeCertificate()` on page 212
`UserCertificate` on page 215

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

User.getUsername()

Method

Declaration `java.lang.String` **`getUsername`**`()`

Purpose Get the username of the user.

Remarks User credentials can be either a username and password pair, or an X.509 certificate.

See Also `User` on page 205

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

User.removeCertificate()

Method

Related Form User.removeCertificates()

Declaration

```
User removeCertificate(
    int  id)
  throws ConfigurationException

User removeCertificates(
    int[]  ids)
  throws ConfigurationException
```

Purpose Remove a user certificate.

Remarks These methods use internal certificate ID numbers to select the certificates to remove; see `UserCertificate.getId()` on page 218.

These methods return the `User` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`id`	Remove the user certificate with this certificate ID.
`ids`	Remove the user certificates with these certificate IDs.

See Also

`User` on page 205
`User.addCertificateFromFile()` on page 206
`User.addCertificateFromText()` on page 207
`User.getCertificates()` on page 210
`UserCertificate` on page 215
`UserCertificate.getId()` on page 218

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

User.setPassword()

Method

Declaration

```
User setPassword(
    java.lang.String  password)
  throws ConfigurationException
```

Purpose Set a user's password.

Remarks This method returns the `User` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`password`	Use this string as the user's new password.

See Also

`User` on page 205

`User.clearPassword()` on page 209

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

User.toXml()

Method

Declaration `java.lang.String` **`toXml`**`()`

Purpose Format the user information as an XML document.

See Also `User` on page 205

For the corresponding browser page, see Users on page 167 in *TIBCO Rendezvous Administration*

UserCertificate

Class

Declaration

```
class com.tibco.tibrv.config.UserCertificate
  extends java.lang.Object
```

Purpose Represent an user's public certificate.

Remarks The method `User.getCertificates()` and related methods return objects of this class.

(Sheet 1 of 2)

Method	Description	Page
`UserCertificate.getAssignmentDate()`	Get the date that the daemon registered the certificate and assigned its ID.	217
`UserCertificate.getId()`	Get the certificate ID assigned by the daemon.	218
`UserCertificate.getIndex()`	Get the index of the certificate.	219
`UserCertificate.getIssuer()`	Get the certificate authority that issued the certificate.	220
`UserCertificate.getFileName()`	Get the name of the certificate file.	221
`UserCertificate.getPublicKeyEngine()`	Get the name of the public key algorithm that the certificate uses to create digital signatures.	222
`UserCertificate.getSerialNumber()`	Get the internal serial number of the certificate.	223
`UserCertificate.getSubject()`	Get information describing the authorized certificate holder.	224
`UserCertificate.getValidNotAfter()`	Get the certificate's expiration date.	225
`UserCertificate.getValidNotBefore()`	Get the date that the certificate is first valid for use.	226
`UserCertificate.getVersion()`	Get the certificate version number assigned by the issuer.	227

(Sheet 2 of 2)

Method	Description	Page
`UserCertificate.toXml()`	Format the user's X.509 certificate information as an XML document.	228

See Also `User.getCertificates()` on page 210

For the corresponding browser pages, see Users on page 197 in *TIBCO Rendezvous Administration*

UserCertificate.getAssignmentDate()

Method

Declaration

```
java.lang.String getAssignmentDate()
  throws ConfigurationException
```

Purpose Get the date that the daemon registered the certificate and assigned its ID.

See Also `UserCertificate` on page 215

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

UserCertificate.getId()

Method

Declaration `int getId()`

Purpose Get the certificate ID assigned by the daemon.

Remarks Use this ID to remove certificates with `User.removeCertificate()` on page 212.

See Also `UserCertificate` on page 215

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

UserCertificate.getIndex()

Method

Declaration

```
int getIndex()
  throws ConfigurationException
```

Purpose Get the index of the certificate.

Remarks The index reflects the position of the certificate within the user's list of certificates. Index 1 denotes the first certificate.

See Also `UserCertificate` on page 215

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

UserCertificate.getIssuer()

Method

Declaration

```
java.lang.String getIssuer()
  throws ConfigurationException
```

Purpose Get the certificate authority that issued the certificate.

See Also `UserCertificate` on page 215

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

UserCertificate.getFileName()

Method

Declaration

```
java.lang.String getFileName()
  throws ConfigurationException
```

Purpose Get the name of the certificate file.

Remarks If the method `User.addCertificateFromFile()` created this certificate, then this method returns the name of the file from which the daemon read the certificate data. Otherwise, this method returns `null`.

See Also `UserCertificate` on page 215

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

UserCertificate.getPublicKeyEngine()

Method

Declaration

```
java.lang.String getPublicKeyEngine()
  throws ConfigurationException
```

Purpose

Get the name of the public key algorithm that the certificate uses to create digital signatures.

See Also

`UserCertificate` on page 215

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

UserCertificate.getSerialNumber()

Method

Declaration

```
java.lang.String getSerialNumber()
  throws ConfigurationException
```

Purpose Get the internal serial number of the certificate.

See Also `UserCertificate` on page 215

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

UserCertificate.getSubject()

Method

Declaration

```
java.lang.String getSubject()
  throws ConfigurationException
```

Purpose Get information describing the authorized certificate holder.

See Also `UserCertificate` on page 215

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

UserCertificate.getValidNotAfter()

Method

Declaration `java.lang.String` **`getValidNotAfter`**`()`

Purpose Get the certificate's expiration date.

See Also `UserCertificate` on page 215

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

UserCertificate.getValidNotBefore()

Method

Declaration `java.lang.String` **`getValidNotBefore`**`()`

Purpose Get the date that the certificate is first valid for use.

See Also `UserCertificate` on page 215

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

UserCertificate.getVersion()

Method

Declaration `java.lang.String` **`getVersion`**`()`

Purpose Get the certificate version number assigned by the issuer.

See Also `UserCertificate` on page 215

For the corresponding browser page, see Existing Users on page 198 in *TIBCO Rendezvous Administration*

For background information, see Users on page 167 in *TIBCO Rendezvous Administration*.

UserCertificate.toXml()

Method

Declaration `java.lang.String` **`toXml`**`()`

Purpose Format the user's X.509 certificate information as an XML document.

See Also `User` on page 205

For the corresponding browser page, see Users on page 197 in *TIBCO Rendezvous Administration*

Chapter 7 Current Value Cache—rvcache

This chapter describes the proxy interface for the Rendezvous recent values cache component (`rvcache`), and the immediate-access data objects that support it.

Topics

See Also

- RvcacheInformation, page 278

RvcacheProxy

Interface

Declaration

```
interface com.tibco.tibrv.config.RvcacheProxy
  extends DaemonProxy
```

Purpose Define methods for a Rendezvous cache process.

Constant	Description
RvcacheProxy.DEFAULT_ACTIVATION	These three constants represent default values for rvcache fault tolerance parameters. You may supply them to RvcacheProxy.setFaultToleranceParams() on page 241.
RvcacheProxy.DEFAULT_HEARTBEAT	
RvcacheProxy.DEFAULT_WEIGHT	
RvcacheProxy.IDLE_STATE	Represent the idle state of rvcache. Supply this constant to RvcacheProxy.changeState().
RvcacheProxy.RUNNING_STATE	Represent the running state of rvcache. Supply this constant to RvcacheProxy.changeState().
RvcacheProxy.UNSPECIFIED	Represent a parameter that is not currently specified in rvcache.

(Sheet 1 of 2)

Method	Description	Page
RvcacheProxy.addSubjectMerge() RvcacheProxy.addSubjectsMerge()	Add a subject with merge semantics.	232
RvcacheProxy.addSubjectReplace() RvcacheProxy.addSubjectsReplace()	Add a subject with replace semantics.	233
RvcacheProxy.changeState()	Change the state of an rvcache process.	234
RvcacheProxy.disableFaultTolerance()	Disable fault tolerance machinery.	235

(Sheet 2 of 2)

Method	Description	Page
`RvcacheProxy.getCachedSubjects()`	Get subjects that this process caches	236
`RvcacheProxy.getFaultToleranceParams()`	Get fault tolerance parameters.	237
`RvcacheProxy.getNetworkParams()`	Get transport parameters.	238
`RvcacheProxy.isRunning()`	Determine the state of rvcache—running or idle.	239
`RvcacheProxy.removeSubject()` `RvcacheProxy.removeSubjects()`	Remove a subject from the cache.	240
`RvcacheProxy.setFaultToleranceParams()`	Enable fault tolerance machinery, and set its parameters.	241
`RvcacheProxy.setNetworkParams()`	Set transport parameters.	242

Inherited Methods
`DaemonProxy.getComponentName()` `DaemonProxy.getComponentInformation()`
`XmlSerializable.printXml()` `XmlSerializable.toXml()`

Components The component `rvcache` supports this interface.

See Also For information about the parameters that this interface can access, see these sections:

- Current Value Cache on page 265 in *TIBCO Rendezvous Administration*
- Browser Administration Interface on page 281 in *TIBCO Rendezvous Administration*

RvcacheProxy.addSubjectMerge()

Method

Related Forms RvcacheProxy.addSubjectsMerge()

Declaration

```
RvcacheProxy addSubjectMerge(
    java.lang.String  subject)
  throws ConfigurationException

RvcacheProxy addSubjectsMerge(
    java.lang.String[]  subjects)
  throws ConfigurationException
```

Purpose Add a subject with merge semantics.

Remarks Merge semantics merges stored data from previous messages with data from new messages; see Replace and Merge on page 273 in *TIBCO Rendezvous Administration*.

This method returns the `RvcacheProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`subject`	Begin caching subjects that match this specification. Wildcards are permitted.
`subjects`	Begin caching subjects that match these specifications. Wildcards are permitted.

See Also `RvcacheProxy` on page 230

For background information, see Browser Administration Interface on page 281 in *TIBCO Rendezvous Administration*.

RvcacheProxy.addSubjectReplace()

Method

Related Forms RvcacheProxy.addSubjectsReplace()

Declaration

```
RvcacheProxy addSubjectReplace(
    java.lang.String  subject)
  throws ConfigurationException

RvcacheProxy addSubjectsReplace(
    java.lang.String[]  subjects)
  throws ConfigurationException
```

Purpose Add a subject with replace semantics.

Remarks Replace semantics replaces stored data from previous messages with data from new messages; see Replace and Merge on page 273 in *TIBCO Rendezvous Administration*.

This method returns the `RvcacheProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`subject`	Begin caching subjects that match this specification. Wildcards are permitted.
`subjects`	Begin caching subjects that match these specifications. Wildcards are permitted.

See Also `RvcacheProxy` on page 230

For background information, see Browser Administration Interface on page 281 in *TIBCO Rendezvous Administration*.

RvcacheProxy.changeState()

Method

Declaration

```
RvcacheProxy changeState(
    int  state)
  throws ConfigurationException

RvcacheProxy changeState()
  throws ConfigurationException
```

Purpose Change the state of an rvcache process.

Remarks The first method changes to the state that the argument specifies. The second method toggles the current state.

These methods return the `RvcacheProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`state`	Change to this state. Supply one of two constants; either `RvcacheProxy.IDLE_STATE` or `RvcacheProxy.RUNNING_STATE`.

See Also `RvcacheProxy` on page 230

For background information, see Browser Administration Interface on page 281 in *TIBCO Rendezvous Administration*.

RvcacheProxy.disableFaultTolerance()

Method

Declaration

```
RvcacheProxy disableFaultTolerance()
  throws ConfigurationException
```

Purpose Disable fault tolerance machinery.

Remarks This method returns the `RvcacheProxy` object, so programs can conveniently chain additional method calls to the return value.

See Also `RvcacheProxy` on page 230

For background information, see these sections:

- Fault Tolerance on page 272 in *TIBCO Rendezvous Administration*
- Browser Administration Interface on page 281 in *TIBCO Rendezvous Administration*

RvcacheProxy.getCachedSubjects()

Method

Declaration

```
CachedSubject[] getCachedSubjects()
  throws ConfigurationException
```

Purpose Get subjects that this process caches

See Also

RvcacheProxy on page 230
CachedSubject on page 247

For background information, see Browser Administration Interface on page 281 in *TIBCO Rendezvous Administration*.

RvcacheProxy.getFaultToleranceParams()

Method

Declaration

```
FaultToleranceParams getFaultToleranceParams()
  throws ConfigurationException
```

Purpose

Get fault tolerance parameters.

Remarks

This method returns a read only collection of fault tolerance parameters. For descriptions of the parameters, see `RvcacheProxy.setFaultToleranceParams()` on page 241.

See Also

`RvcacheProxy` on page 230
`FaultToleranceParams` on page 255

For background information, see these sections:

- Fault Tolerance on page 272 in *TIBCO Rendezvous Administration*
- Browser Administration Interface on page 281 in *TIBCO Rendezvous Administration*

RvcacheProxy.getNetworkParams()

Method

Declaration

```
RvcacheNetworkParams getNetworkParams()
  throws ConfigurationException
```

Purpose

Get transport parameters.

Remarks

`rvcache` uses the transport for all network communication.

This method returns an array of parameters that specify the transport. For descriptions of the parameters, see `RvcacheProxy.setNetworkParams()` on page 242.

See Also

`RvcacheProxy` on page 230
`RvcacheProxy.setNetworkParams()` on page 242
`RvcacheNetworkParams` on page 264

For background information, see Browser Administration Interface on page 281 in *TIBCO Rendezvous Administration*.

RvcacheProxy.isRunning()

Method

Declaration

```
boolean isRunning()
  throws ConfigurationException
```

Purpose Determine the state of rvcache—running or idle.

Remarks This method returns `true` if the state of `rvcache` is *running*; `false` if its state is *idle*.

See Also `RvcacheProxy` on page 230

For background information, see Browser Administration Interface on page 281 in *TIBCO Rendezvous Administration*.

RvcacheProxy.removeSubject()

Method

Related Forms RvcacheProxy.removeSubjects()

Declaration

```
RvcacheProxy removeSubject(
    java.lang.String  subject)
  throws ConfigurationException

RvcacheProxy removeSubjects(
    java.lang.String[]  subjects)
  throws ConfigurationException
```

Purpose Remove a subject from the cache.

Remarks After removing a subject, `rvcache` no longer stores it, nor forwards its values.

This method returns the `RvcacheProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`subject`	Stop caching subjects that match this specification. Wildcards are permitted.
`subjects`	Stop caching subjects that match these specifications. Wildcards are permitted.

See Also `RvcacheProxy` on page 230

For background information, see Browser Administration Interface on page 281 in *TIBCO Rendezvous Administration*.

RvcacheProxy.setFaultToleranceParams()

Method

Declaration

```
RvcacheProxy setFaultToleranceParams(
    int                servicePort,
    java.lang.String   network,
    java.lang.String   group,
    int                weight,
    double             heartbeat,
    double             activation)
  throws ConfigurationException
```

Purpose Enable fault tolerance machinery, and set its parameters.

Remarks This method returns the `RvcacheProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`servicePort`	Use this UDP service for internal fault tolerance protocols.
`network`	Use this network for internal fault tolerance protocols.
`group`	Join a fault tolerance group with this group name.
`weight`	Use this member weight. You may supply the constant value `RvcacheProxy.DEFAULT_WEIGHT` (10).
`heartbeat`	Use this heartbeat interval (in seconds). You may supply the constant value `RvcacheProxy.DEFAULT_HEARTBEAT` (3).
`activation`	Use this activation interval (in seconds). You may supply the constant value `RvcacheProxy.DEFAULT_ACTIVATION` (10).

See Also `RvcacheProxy` on page 230

For background information, see these sections:

- Fault Tolerance on page 272 in *TIBCO Rendezvous Administration*
- Browser Administration Interface on page 281 in *TIBCO Rendezvous Administration*

RvcacheProxy.setNetworkParams()

Method

Declaration

```
RvcacheProxy setNetworkParams(
    int               servicePort,
    java.lang.String  network,
    java.lang.String  daemon)
  throws ConfigurationException
```

Purpose Set transport parameters.

Remarks `rvcache` uses the transport for all network communication.

This method returns the `RvcacheProxy` object, so programs can conveniently chain additional method calls to the return value.

Parameter	Description
`servicePort`	Communicate on this UDP service.
`network`	Communicate on this network.
`daemon`	Request a client connection to `rvd` on this TCP port.

See Also

`RvcacheProxy` on page 230
`RvcacheProxy.getNetworkParams()` on page 238

For background information, see Browser Administration Interface on page 281 in *TIBCO Rendezvous Administration*.

CachedField

Class

Declaration

```
class com.tibco.tibrv.config.CachedField
  extends java.lang.Object
```

Purpose Represent a field in a cached message.

Remarks The method `CachedSubject.getFields()` returns objects of this class.

Method	Description	Page
`CachedField.getDataType()`	Represent a field in a cached message.	241
`CachedField.getFieldName()`	Get the Rendezvous datatype of the field.	244
`CachedField.getValue()`	Get the data value of the field.	246

Inherited Methods

```
java.lang.Object.equals()
java.lang.Object.getClass()
java.lang.Object.hashCode()
java.lang.Object.notify()
java.lang.Object.notifyAll()
java.lang.Object.toString() override
java.lang.Object.wait()
```

See Also `CachedSubject.getFields()` on page 249

CachedField.getDataType()

Method

Declaration `java.lang.String` **`getDataType`**`()`

Purpose Get the Rendezvous datatype of the field.

See Also `CachedField` on page 243

For background information, see Rendezvous Datatypes on page 51 in *TIBCO Rendezvous Concepts*.

CachedField.getFieldName()

Method

Declaration `java.lang.String` **`getFieldName`**`()`

Purpose Get the name of the field.

See Also `CachedField` on page 243

For background information, see Field Names and Field Identifiers on page 55 in *TIBCO Rendezvous Concepts*.

CachedField.getValue()

Method

Declaration `java.lang.String` **`getValue`**`()`

Purpose Get the data value of the field.

See Also `CachedField` on page 243

For background information, see Messages on page 53 in *TIBCO Rendezvous Concepts*.

CachedSubject

Class

Declaration

```
class com.tibco.tibrv.config.CachedSubject
  extends java.lang.Object
```

Purpose Represent a subject in the cache.

Remarks The method `RvcacheProxy.getCachedSubjects()` returns objects of this class.

Constant	Description
`CachedSubject.MERGE_MODE`	`CachedSubject.getStorageMethod()` on page 252 returns these constants.
`CachedSubject.REPLACE_MODE`	

Method	Description	Page
`CachedSubject.getFields()`	Get the message fields of the cached subject.	249
`CachedSubject.getInitialValuesServed()`	Get the number of times rvcache has delivered this subject.	250
`CachedSubject.getMessageSize()`	Get the size of the cached message (in bytes).	251
`CachedSubject.getStorageMethod()`	Get the mode for storing inbound messages for the subject.	252
`CachedSubject.getSubject()`	Get the subject name.	253
`CachedSubject.getUpdatesApplied()`	Get the number of times that an inbound message has updated the cached data for this subject.	254

Inherited Methods

`java.lang.Object.equals()`
`java.lang.Object.getClass()`
`java.lang.Object.hashCode()`
`java.lang.Object.notify()`
`java.lang.Object.notifyAll()`
`java.lang.Object.toString()` override
`java.lang.Object.wait()`

See Also `RvcacheProxy.getCachedSubjects()` on page 236

For background information, see Browser Administration Interface on page 281 in *TIBCO Rendezvous Administration*.

CachedSubject.getFields()

Method

Declaration `CachedField[] getFields()`

Purpose Get the message fields of the cached subject.

See Also `CachedField` on page 243
`CachedSubject` on page 247

For background information, see Browser Administration Interface on page 281 in *TIBCO Rendezvous Administration*.

CachedSubject.getInitialValuesServed()

Method

Declaration `int getInitialValuesServed()`

Purpose Get the number of times rvcache has delivered this subject.

See Also `CachedSubject` on page 247

CachedSubject.getMessageSize()

Method

Declaration `int getMessageSize()`

Purpose Get the size of the cached message (in bytes).

See Also `CachedSubject` on page 247

For background information, see Browser Administration Interface on page 281 in *TIBCO Rendezvous Administration*.

CachedSubject.getStorageMethod()

Method

Declaration `int getStorageMethod()`

Purpose Get the mode for storing inbound messages for the subject.

Remarks This method returns the constants `CachedSubject.MERGE_MODE` and `CachedSubject.REPLACE_MODE`.

See Also `CachedSubject` on page 247

For background information, see Replace and Merge on page 273 in *TIBCO Rendezvous Administration*

CachedSubject.getSubject()

Method

Declaration `java.lang.String` **`getSubject`**`()`

Description Get the subject name.

See Also `CachedSubject` on page 247

For background information, see Replace and Merge on page 273 in *TIBCO Rendezvous Administration*

CachedSubject.getUpdatesApplied()

Method

Declaration `int getUpdatesApplied()`

Description Get the number of times that an inbound message has updated the cached data for this subject.

See Also `CachedSubject` on page 247

FaultToleranceParams

Class

Declaration

```
class com.tibco.tibrv.config.FaultToleranceParams
  extends java.lang.Object
```

Purpose Encapsulate fault tolerance parameters from Rendezvous recent value caches.

Constant	Description
`FaultToleranceParams.UNSPECIFIED`	No value is set for the parameter.

Method	Description	Page
`FaultToleranceParams.getActivation()`	Extract the activation interval.	256
`FaultToleranceParams.getAsMap()`	Format the fault tolerance parameters as a map.	257
`FaultToleranceParams.getGroup()`	Extract the name of the fault tolerance group.	258
`FaultToleranceParams.getHeartbeat()`	Extract the heartbeat interval.	259
`FaultToleranceParams.getNetwork()`	Extract the network parameter for fault tolerance protocols.	260
`FaultToleranceParams.getService()`	Extract the effective UDP service for fault tolerance protocols.	261
`FaultToleranceParams.getWeight()`	Extract the fault tolerance weight of this member.	262
`FaultToleranceParams.isEnabled()`	Extract a flag indicating whether fault tolerance is enabled.	263

Remarks Instances of this class are read-only objects. Methods do not interact with the component.

`RvcacheProxy.getFaultToleranceParams()` returns instances of this class.

See Also Read-Only Objects on page 5

`RvcacheProxy.getFaultToleranceParams()` on page 237

FaultToleranceParams.getActivation()

Method

Declaration `double` **`getActivation`**`()`

Purpose Extract the activation interval.

Remarks This method does not interact with the component.

FaultToleranceParams.getAsMap()

Method

Declaration `java.util.Map` **`getAsMap`**`()`

Purpose Format the fault tolerance parameters as a map.

Remarks The resulting map is useful for iterative methods, such as printing.

This method does not interact with the component.

FaultToleranceParams.getGroup()

Method

Declaration `java.lang.String` **`getGroup`**`()`

Purpose Extract the name of the fault tolerance group.

Remarks This method does not interact with the component.

FaultToleranceParams.getHeartbeat()

Method

Declaration `double getHeartbeat()`

Purpose Extract the heartbeat interval.

Remarks This method does not interact with the component.

FaultToleranceParams.getNetwork()

Method

Declaration `java.lang.String` **`getNetwork`**`()`

Purpose Extract the network parameter for fault tolerance protocols.

Remarks This method does not interact with the component.

FaultToleranceParams.getService()

Method

Declaration `int getService()`

Purpose Extract the effective UDP service for fault tolerance protocols.

Remarks This method does not interact with the component.

FaultToleranceParams.getWeight()

Method

Declaration `int getWeight()`

Purpose Extract the fault tolerance weight of this member.

Remarks This method does not interact with the component.

FaultToleranceParams.isEnabled()

Method

Declaration `boolean isEnabled()`

Purpose Extract a flag indicating whether fault tolerance is enabled.

Remarks This method does not interact with the component.

RvcacheNetworkParams

Class

Declaration

```
class com.tibco.tibrv.config.RvcacheNetworkParams
  extends java.lang.Object
```

Purpose Encapsulate network parameters from Rendezvous recent value caches.

Constant	Description
`RvcacheNetworkParams.UNSPECIFIED`	No value is set for the parameter.

Method	Description	Page
`RvcacheNetworkParams.getAsMap()`	Format the network parameters as a map.	265
`RvcacheNetworkParams.getDaemon()`	Extract the daemon parameter.	266
`RvcacheNetworkParams.getNetwork()`	Extract the network parameter.	267
`RvcacheNetworkParams.getService()`	Extract the service parameter.	268

Remarks Instances of this class are read-only objects. Methods do not interact with the component.

See Also Read-Only Objects on page 5

RvcacheNetworkParams.getAsMap()

Method

Declaration `java.util.Map` **`getAsMap`**`()`

Purpose Format the network parameters as a map.

Remarks The resulting map is useful for iterative methods, such as printing.

This method does not interact with the component.

RvcacheNetworkParams.getDaemon()

Method

Declaration `java.lang.String` **`getDaemon`**`()`

Purpose Extract the daemon parameter.

Remarks `rvcache` requests a client connection to `rvd` on this TCP port.

This method does not interact with the component.

RvcacheNetworkParams.getNetwork()

Method

Declaration `java.lang.String` **`getNetwork`**`()`

Purpose Extract the network parameter.

Remarks `rvcache` communicates with `rvd` on this network.

This method does not interact with the component.

RvcacheNetworkParams.getService()

Method

Declaration `int getService()`

Purpose Extract the service parameter.

Remarks `rvcache` communicates with `rvd` on this UDP service.

This method does not interact with the component.

Chapter 8 Component Information

This chapter describes the read-only objects encapsulate general information about Rendezvous components.

Topics

ComponentInformation

Class

Declaration

```
class com.tibco.tibrv.config.ComponentInformation
  extends java.lang.Object
```

Purpose Encapsulate general information from Rendezvous components.

Method	Description	Page
`ComponentInformation.getAsMap()`	Format the component information as a map.	271
`ComponentInformation.getHostname()`	Extract the name of the host computer where the component is running.	272
`ComponentInformation.getIpAddress()`	Extract the IP address of the host computer where the component is running.	273
`ComponentInformation.getLicenseTicket()`	Extract the Rendezvous license ticket that validates the component.	274
`ComponentInformation.getName()`	Extract the component name.	275
`ComponentInformation.getProcessID()`	Extract the process ID.	276
`ComponentInformation.getVersion()`	Extract the version number of the component (as a string).	277

Remarks Instances of this class are read-only objects. Methods do not interact with the component.

`DaemonProxy.getComponentInformation()` returns objects of this class.

Subclasses

`RvcacheInformation` on page 278
`RvdInformation` on page 284
`RvrdInformation` on page 288
`RvsdInformation` on page 291
`RvsrdInformation` on page 293

See Also

Read-Only Objects on page 5
`DaemonProxy.getComponentInformation()` on page 16

ComponentInformation.getAsMap()

Method

Declaration `java.util.Map` **`getAsMap`**`()`

Purpose Format the component information as a map.

Remarks The resulting map is useful for iterative methods, such as printing.

This method does not interact with the component.

ComponentInformation.getHostname()

Method

Declaration `java.lang.String` **`getHostname`**`()`

Purpose Extract the name of the host computer where the component is running.

This method does not interact with the component.

ComponentInformation.getIpAddress()

Method

Declaration `java.lang.String` **`getIpAddress`**`()`

Purpose Extract the IP address of the host computer where the component is running. This method does not interact with the component.

ComponentInformation.getLicenseTicket()

Method

Declaration `java.lang.String` **`getLicenseTicket`**`()`

Purpose Extract the Rendezvous license ticket that validates the component.

This method does not interact with the component.

ComponentInformation.getName()

Method

Declaration `java.lang.String` **`getName`**`()`

Purpose Extract the component name.

Remarks `DaemonProxy.getComponentName()` gets the same information (immediately from the component).

This method does not interact with the component.

See Also `DaemonProxy.getComponentName()` on page 15

ComponentInformation.getProcessID()

Method

Declaration

```
java.lang.String getProcessID()
  throws java.lang.UnsupportedOperationException
```

Purpose Extract the process ID.

Remarks This method does not interact with the component.

Process ID information is available only when the component is from release 7.1 or later. With release 7.0 components, this method throws an exception.

ComponentInformation.getVersion()

Method

Declaration `java.lang.String` **`getVersion`**`()`

Purpose Extract the version number of the component (as a string).

Remarks This method does not interact with the component.

RvcacheInformation

Class

Declaration

```
class com.tibco.tibrv.config.RvcacheInformation
  extends ComponentInformation
```

Purpose Encapsulate general information from Rendezvous recent value caches.

Constant	Description
RvcacheInformation.FAULT_TOLERANCE_ENABLED	Indicate whether a cache process can participate in a fault tolerance group.
RvcacheInformation.FAULT_TOLERANCE_DISABLED	
RvcacheInformation.SHALLOW	Indicate whether a cache process uses shallow or deep merge for nested messages.
RvcacheInformation.DEEP	
RvcacheInformation.STORE	Indicate whether a cache process writes cached values to its store file for persistence, or operates in memory-only mode.
RvcacheInformation.MEMORY_ONLY	

Method	Description	Page
RvcacheInformation.getFaultToleranceState()	Extract a constant indicating whether the cache has enabled fault tolerant operation.	280
RvcacheInformation.getMergeMode()	Extract a constant indicating whether the cache uses shallow or deep merge for nested messages.	281
RvcacheInformation.getCacheMode()	Extract a constant indicating whether a cache process writes cached values to its store file for persistence, or operates in memory-only mode.	282
RvcacheInformation.getState()	Extract the state (running or idle) of the cache.	283

Inherited Methods

`ComponentInformation.getAsMap()`
`ComponentInformation.getHostname()`
`ComponentInformation.getIpAddress()`
`ComponentInformation.getLicenseTicket()`
`ComponentInformation.getName()`
`ComponentInformation.getVersion()`

Remarks Instances of this class are read-only objects. Methods do not interact with the component.

`DaemonProxy.getComponentInformation()` can return instances of this class.

Superclasses `ComponentInformation` on page 270

See Also Read-Only Objects on page 5
`DaemonProxy.getComponentInformation()` on page 16

RvcacheInformation.getFaultToleranceState()

Method

Declaration `int getFaultToleranceState()`

Purpose Extract a constant indicating whether the cache has enabled fault tolerant operation.

Remarks This method returns one of these constants:

- `RvcacheInformation.FAULT_TOLERANCE_ENABLED`
- `RvcacheInformation.FAULT_TOLERANCE_DISABLED`

This method does not interact with the component.

See Also `RvcacheInformation` on page 278

RvcacheInformation.getMergeMode()

Method

Declaration `int getMergeMode()`

Purpose Extract a constant indicating whether the cache uses shallow or deep merge for nested messages.

Remarks This method returns one of these constants:

- `RvcacheInformation.SHALLOW`
- `RvcacheInformation.DEEP`

This method does not interact with the component.

See Also `RvcacheInformation` on page 278
Replace and Merge on page 273 in *TIBCO Rendezvous Administration*

RvcacheInformation.getCacheMode()

Method

Declaration `int getCacheMode()`

Purpose Extract a constant indicating whether a cache process writes cached values to its store file for persistence, or operates in memory-only mode.

Remarks This method returns one of these constants:

- `RvcacheInformation.STORE`
- `RvcacheInformation.MEMORY_ONLY`

This method does not interact with the component.

See Also `RvcacheInformation` on page 278
Memory-Only Mode on page 275 in *TIBCO Rendezvous Administration*

RvcacheInformation.getState()

Method

Declaration `int getState()`

Purpose Extract the state (running or idle) of the cache.

Remarks This method returns one of these constants:

- `RvcacheProxy.IDLE_STATE`
- `RvcacheProxy.RUNNING_STATE`

Remarks This method does not interact with the component.

See Also `RvcacheProxy` on page 230

RvdInformation

Class

Declaration

```
class com.tibco.tibrv.config.RvdInformation
  extends ComponentInformation
```

Purpose Encapsulate general information from Rendezvous communications daemons.

Method	Description	Page
`RvdInformation.getClientPort()`	Extract the TCP port where the daemon listens for client connections.	285
`RvdInformation.getNetworkServicesCount()`	Extract the number of network services on which this daemon's clients communicate.	286
`RvdInformation.getUsername()`	Extract the login name of the user that started the component process.	287

Inherited Methods

`ComponentInformation.getAsMap()`
`ComponentInformation.getHostname()`
`ComponentInformation.getIpAddress()`
`ComponentInformation.getLicenseTicket()`
`ComponentInformation.getName()`
`ComponentInformation.getVersion()`

Remarks Instances of this class are read-only objects. Methods do not interact with the component.

`DaemonProxy.getComponentInformation()` can return instances of this class.

Subclasses
`RvsdInformation` on page 291
`RvrdInformation` on page 288

Superclasses
`ComponentInformation` on page 270

See Also
Read-Only Objects on page 5
`DaemonProxy.getComponentInformation()` on page 16

RvdInformation.getClientPort()

Method

Declaration `int getClientPort()`

Purpose Extract the TCP port where the daemon listens for client connections.

Remarks This method does not interact with the component.

RvdInformation.getNetworkServicesCount()

Method

Declaration `int getNetworkServicesCount()`

Purpose Extract the number of network services on which this daemon's clients communicate.

Remarks This method does not interact with the component.

RvdInformation.getUsername()

Method

Declaration `java.lang.String` **`getUsername`**`()`

Purpose Extract the login name of the user that started the component process.

Remarks This method does not interact with the component.

RvrdInformation

Class

Declaration

```
class com.tibco.tibrv.config.RvrdInformation
  extends RvdInformation
```

Purpose Encapsulate general information from Rendezvous routing daemons.

Method	Description	Page
`RvrdInformation.getRoutingNamesCount()`	Extract the number of routing names that the daemon embodies.	289
`RvrdInformation.getStoreFilePath()`	Extract the file name of the daemon's store file.	290

Inherited Methods

`RvdInformation.getClientPort()`
`RvdInformation.getNetworkServicesCount()`
`RvdInformation.getUsername()`

`ComponentInformation.getAsMap()`
`ComponentInformation.getHostname()`
`ComponentInformation.getIpAddress()`
`ComponentInformation.getLicenseTicket()`
`ComponentInformation.getName()`
`ComponentInformation.getVersion()`

Remarks Instances of this class are read-only objects. Methods do not interact with the component.

`DaemonProxy.getComponentInformation()` can return instances of this class.

Subclasses `RvsrdInformation` on page 293

Superclasses `ComponentInformation` on page 270
`RvdInformation` on page 284

See Also Read-Only Objects on page 5
`DaemonProxy.getComponentInformation()` on page 16

RvrdInformation.getRoutingNamesCount()

Method

Declaration `int getRoutingNamesCount()`

Purpose Extract the number of routing names that the daemon embodies.

Remarks This method does not interact with the component.

RvrdInformation.getStoreFilePath()

Method

Declaration `java.lang.String` **`getStoreFilePath`**`()`

Purpose Extract the file name of the daemon's store file.

Remarks This method does not interact with the component.

RvsdInformation

Class

Declaration

```
class com.tibco.tibrv.config.RvsdInformation
  extends RvdInformation
```

Purpose Encapsulate general information from Rendezvous secure communications daemons.

Method	Description	Page
`RvsdInformation.getStoreFilePath()`	Extract the file name of the daemon's store file.	292

Inherited Methods

`RvdInformation.getClientPort()`
`RvdInformation.getNetworkServicesCount()`
`RvdInformation.getUsername()`

`ComponentInformation.getAsMap()`
`ComponentInformation.getHostname()`
`ComponentInformation.getIpAddress()`
`ComponentInformation.getLicenseTicket()`
`ComponentInformation.getName()`
`ComponentInformation.getVersion()`

Remarks Instances of this class are read-only objects. Methods do not interact with the component.

`DaemonProxy.getComponentInformation()` can return instances of this class.

Superclasses
`ComponentInformation` on page 270
`RvdInformation` on page 284

See Also
Read-Only Objects on page 5
`DaemonProxy.getComponentInformation()` on page 16

RvsdInformation.getStoreFilePath()

Method

Declaration `java.lang.String` **`getStoreFilePath`**`()`

Purpose Extract the file name of the daemon's store file.

Remarks This method does not interact with the component.

RvsrdInformation

Class

Declaration

```
class com.tibco.tibrv.config.RvsrdInformation
  extends RvrdInformation
```

Purpose Encapsulate general information from Rendezvous secure routing daemons.

Inherited Methods

`RvrdInformation.getRoutingNamesCount()`
`RvrdInformation.getStoreFilePath()`

`RvdInformation.getClientPort()`
`RvdInformation.getNetworkServicesCount()`
`RvdInformation.getUsername()`

`ComponentInformation.getAsMap()`
`ComponentInformation.getHostname()`
`ComponentInformation.getIpAddress()`
`ComponentInformation.getLicenseTicket()`
`ComponentInformation.getName()`
`ComponentInformation.getVersion()`

Remarks Instances of this class are read-only objects. Methods do not interact with the component.

`DaemonProxy.getComponentInformation()` can return instances of this class.

This class does not add any new methods to its superclass.

Superclasses

`ComponentInformation` on page 270
`RvdInformation` on page 284
`RvrdInformation` on page 288

See Also

Read-Only Objects on page 5
`DaemonProxy.getComponentInformation()` on page 16

Chapter 9 Exceptions

This chapter describes the exception classes for the Rendezvous configuration API.

Topics

ConfigurationException

Class

Declaration

```
class com.tibco.tibrv.config.ConfigurationException
  extends java.lang.Exception
```

Purpose

Encapsulate errors from the daemon component.

Remarks

Configuration methods throw this class of exceptions when the daemon component rejects a parameter or configuration command. The exception's message string contains the text of an error message from the daemon component. Programs can extract the message with `java.lang.Exception.getMessage()`.

To handle this exception, we recommend that programs log the error message, and display it to the user. When appropriate, the program may offer the user an opportunity to substitute a new parameter or command.

Inherited Methods

```
java.lang.Throwable.fillInStackTrace()
java.lang.Throwable.getCause()
java.lang.Throwable.getLocalizedMessage()
java.lang.Throwable.getMessage()
java.lang.Throwable.getStackTrace()
java.lang.Throwable.initCause
java.lang.Throwable.printStackTrace()
java.lang.Throwable.setStackTrace()
java.lang.Throwable.toString()
```

```
java.lang.Object.equals()
java.lang.Object.getClass()
java.lang.Object.hashCode()
java.lang.Object.notify()
java.lang.Object.notifyAll()
java.lang.Object.wait()
```

FatalConfigurationException

Class

Declaration

```
class com.tibco.tibrv.config.FatalConfigurationException
  extends ConfigurationException
```

Purpose Encapsulate fatal errors.

Remarks Configuration methods throw this class of exceptions when a problem prevents continued execution.

To handle these exceptions, we recommend that programs print a stack trace and exit.

Constant	Description
CORRUPTED_PACKAGE	One of the Java configuration classes is missing or invalid.
INVALID_ANSWER	The component could not answer a request from a method of the configuration API.
UNAUTHORIZED	The configuration program lacks correct administrator credentials.
UNEXPECTED_ANSWER	The configuration API method could not parse an answer from the component.
UNKNOWN_COMPONENT	The daemon or component does not support the configuration interface.

Inherited Methods

```
java.lang.Throwable.fillInStackTrace()
java.lang.Throwable.getCause()
java.lang.Throwable.getLocalizedMessage()
java.lang.Throwable.getMessage()
java.lang.Throwable.getStackTrace()
java.lang.Throwable.initCause
java.lang.Throwable.printStackTrace()
java.lang.Throwable.setStackTrace()
java.lang.Throwable.toString()
```

```
java.lang.Object.equals()
java.lang.Object.getClass()
java.lang.Object.hashCode()
java.lang.Object.notify()
java.lang.Object.notifyAll()
java.lang.Object.wait()
```

Chapter 10 Command Line Tool—tibrvcfg

This chapter describes a command line tool for configuring Rendezvous daemons.

Topics

Overview

The command line tool (`tibrvcfg`) lets you configure daemon components without writing Java programs. The tool interacts with a component to execute one configuration command, and outputs the results to `stdio`. Most of the commands parallel methods of the configuration API.

XML

Three commands interact with XML documents:

dumpXML `dumpXML` extracts the complete set of configuration information from a component, and outputs it as an XML document.

mergeXML matchXML `mergeXML` and `matchXML` parse an XML file that specifies a configuration, compare it with the current configuration of a component, arrange the minimal set of configuration commands that would implement the specified configuration on the component, and interact with the component to configure it accordingly.

Translation Rendezvous components do not accept or produce XML directly. Instead, methods of the configuration API act as translators:

- `dumpXML` gets current configuration data from a component, and builds an XML document incorporating that data.
- Conversely, `mergeXML` and `matchXML` read an XML document, and build a set of commands to achieve the specified configuration.

Sensitive Information `dumpXML` does not extract sensitive information from components—for example, passwords or private keys. Instead, this command replaces sensitive information with a static string.

This replacement is a security feature, but it can also create confusion for unwary users. If you dump an XML document that contains such replacement strings, and modify it to change some of the parameter values, you cannot successfully load those changes (with the `mergeXML` or `matchXML` commands) *unless* you first remove these replacement strings. For example, in the XML document below, remove the indicated text—that is, everything in the `security-parameters` section.

Example 1 Removing Security Replacement Strings from XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rendezvous SYSTEM "http://www.rv.tibco.com/dtd/rv">
```

```
<rendezvous url="http://arrakis:7580">
  <configuration timestamp="20040907152842-0700">
    <rvrd-parameters>
      <logging connections="no"
               subject-data="no" subject-interest="no" />
    </rvrd-parameters>
    <security-parameters>
      <administrator password="!!!SECRET!!!"
username="Administrator" />
      <certificate index="1" private-key-password="!!! SECRET !!!">
        <use for="HTTPS" />
        <use for="ROUTERS_TO_ROUTERS" />
        <PEM-data>
          -----BEGIN CERTIFICATE-----
MIICvzCCAiigAwIBAgIBATANBgkqhkiG9w0BAQQFADBhMQswCQYDVQQGEwJOQTEL
MAkGA1UECBMCTkExCzAJBgNVBAcTAk5BMRIwEAYDVQQKEwlBbm9ueW1vdXMxEjAQ
BgNVBAsTCUFub255bW91czEQMA4GA1UEAxMHYXJyYWtpczAeFw0wNDAzMzAyMjMy
MzFaFw0wNTAzMzAyMjMyMzFaMGExCzAJBgNVBAYTAk5BMQswCQYDVQQIEwJOQTEL
MAkGA1UEBxMCTkExEjAQBgNVBAoTCUFub255bW91czESMBAGA1UECxMJQW5vbnlt
b3VzMRAwDgYDVQQDEwdhcnJha2lzMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKB
gQDAf+fcyb3UdMjPFtaXetSldSGTfrwV9/t+1xUlXaVb79w2jtS6RtNCg9WkzUy2
78DoZxlD3szKoZt7rdZlssCGWSVx2jO568alYF+5r5RR8Dj5I3ZEQC+iFJyCRprn
t/Kh07UKd3fNy8s4KoQBGTxs7C+mQpuNgpSHV5uMSDIF2wIDAQABo4GGMIGDMBEG
CWCGSAGG+EIBAQQEAwICRDALBgNVHQ8EBAMCAuwwEwYDVR0lBAwwCgYIKwYBBQUH
AwEwHQYDVR0OBBYEFL9s9h3DVCx/asG/Bqfka0Z/fLdwMB8GA1UdIwQYMBaAFL9s
9h3DVCx/asG/Bqfka0Z/fLdwMAwGA1UdEwQFMAMBAf8wDQYJKoZIhvcNAQEEBQAD
gYEAcJQDKbusUaEYOpMD3iNHSKPYQsrfa76DzxA10MGvLtOEh3kpRoeNKUvqZw5+
dQZIxOnQeY8X7l60GKuwvAiOsyUAgDdOWEGnDJed52+R8NXHg/okmdMnzC05QxO0
gJaiC1mm4zl03gFMtHNUqABh9qmjNJDI34bVwXXC2isCgFg=
          -----END CERTIFICATE-----
          !!! SECRET PRIVATE KEY !!!
        </PEM-data>
      </certificate>
      <certificate index="2" />
      <certificate index="3" />
      <certificate index="4" />
    </security-parameters>
  </configuration>
</rendezvous>
```

After removal, the resulting XML document would look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rendezvous SYSTEM "http://www.rv.tibco.com/dtd/rv">
<rendezvous url="http://arrakis:7580">
 <configuration timestamp="20040907152842-0700">
   <rvrd-parameters>
     <logging connections="no"
              subject-data="no" subject-interest="no" />
   </rvrd-parameters>
   <security-parameters>
   </security-parameters>
 </configuration>
</rendezvous>
```

DTD

A set of DTD files defines the correct syntax for Rendezvous configuration XML documents. The configuration API verifies XML documents against this definition.

You can easily reconfigure a component by editing the output of `dumpXML`, and then supplying the modified file to `mergeXML` or `matchXML`. The edited document must conform to the DTD.

Requirements

Java SDK

The command line tool requires that you first install Java SDK runtime environment 1.4 (or later). You can download this software from java.sun.com.

Environment Variables

Variable	Description
TIBRV_HOME	This variable identifies the directory on your computer where you have installed Rendezvous software. The default location is /TIBCO/TIBRV.
JAVA_HOME	This variable identifies the directory on your computer where you have installed Java SDK.
CLASSPATH	Rendezvous installation places the Java archive file rvconfig.jar in the lib directory under TIBRV_HOME. If you have placed this file in any other location, the CLASSPATH variable must include the complete pathname. The CLASSPATH variable must also include the complete pathname to the SDK file jsse.jar, which implements TLS.

tibrvcfg

Command

Syntax

```
tibrvcfg [-http-only]
         [-login name:password ]
         [-url base_url ]
         command arg1 arg2 ...
```

Purpose Connect to a daemon component and run one configuration command.

Remarks This tool lets you send one configuration command to a component. The set of configuration commands parallels the methods of the configuration API. For details about any command, see page in this book that documents the corresponding method. For XML commands, see XML Commands on page 305.

The script `tibrvcfg` resides in the `bin` directory under `TIBRV_HOME`.

(Sheet 1 of 2)

Parameter	Description
`-http-only`	When present, the configuration tool uses non-secure HTTP protocols to connect to the component, instead of secure HTTPS protocols. You must specify `-http-only` to `tibrvcfg` if and only if the component command line specified `-http-only`.
`-login` *name*:*password*	When present, the configuration tool supplies this administrator name and password when the component requests them. For more information, see `SecurityProxy.useCredentials()` on page 169.
`-url` *base_url*	When present, the configuration tool connects to the browser administration interface of the component at this URL. When loading an XML file, this parameter (if present) overrides the URL in the XML file. When absent, the configuration tool seeks the component at the default URL, `http://localhost:7580`, which is appropriate for `rvd`, `rvrd`, `rvsd` or `rvsrd`, running on the same computer as the configuration tool. Construct the URL from the IP address of the daemon's host computer, and its HTTP port.

(Sheet 2 of 2)

Parameter	Description
command arg1 arg2 ...	The remaining parameters specify the command to the component, and its arguments. If you omit the command, or supply an invalid command, the configuration tool outputs a lengthy help message that lists all valid commands.

Execution as a Java Object

The `tibrvcfg` script is the most convenient way to use this tool, because it automatically arranges the environment properly. However, you can bypass the script, executing the tool as a Java object; for example:

```
java com.tibco.tibrv.config.tools.TibrvConfigurationTool ...
```

XML Commands

(Sheet 1 of 2)

Command	Description
`dumpXML [` *output_filename* `]`	`dumpXML` gets current configuration data from a component, and builds an XML document incorporating that data. When *output_filename* is present, `dumpXML` directs the XML document to the specified file. The file is suitable as input for the `mergeXML` or `matchXML` command. When *output_filename* absent, the default is `stdio`.
`mergeXML` *xml_file*	`mergeXML` reads an XML document, and builds a set of commands to update the component with configuration in the XML document—overlaying the existing configuration with changes specified in the XML file. Where the two configurations conflict, the XML document overrides the existing configuration. Elements of the existing configuration that do not conflict with the XML specification remain in force. The parameter *xml_file* must contain an XML document appropriate for configuring the component (its syntax must conform to the DTD specification).

(Sheet 2 of 2)

Command	Description
`matchXML` *xml_file*	`matchXML` reads an XML document, and builds a set of commands that force the component to conform to the XML document—removing elements of the existing configuration that are absent from the XML specification. The parameter *xml_file* must contain an XML document appropriate for configuring the component (its syntax must conform to the DTD specification).

Merge vs. Match

To illustrate the difference between `mergeXML` and `matchXML`, consider an example in which the existing configuration of `rvrd` has two routers, R1 and R2; the XML document specifies a change to R2 and a new router R3:

- With `mergeXML`, R1 remains unchanged, R2 is modified, and R3 is added.
- With `matchXML`, R1 is removed, R2 is modified, and R3 is added.

Creating XML Input

You may edit the XML output of `dumpXML`, and supply the edited file as input to `mergeXML` or `matchXML`. The edited file must conform to the DTD.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

P

R

S

T

U

X