Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN TIBCO Rendezvous Installation) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIB, TIBCO, TIBCO Adapter, Predictive Business, Information Bus, The Power of Now, Rendezvous, TIBCO Rendezvous and Messaging Appliance are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

EJB, Java EE, J2EE, and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

THIS SOFTWARE MAY BE AVAILABLE ON MULTIPLE OPERATING SYSTEMS. HOWEVER, NOT ALL OPERATING SYSTEM PLATFORMS FOR A SPECIFIC SOFTWARE VERSION ARE RELEASED AT THE SAME TIME. SEE THE README.TXT FILE FOR THE AVAILABILITY OF THIS SOFTWARE VERSION ON A SPECIFIC OPERATING SYSTEM PLATFORM.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

Copyright © 1997–2010 TIBCO Software Inc. ALL RIGHTS RESERVED.

TIBCO Software Inc. Confidential Information
Contents

Figures ................................................................. xv
Tables ................................................................. xix
New or Modified Sections .............................................. xxi
Preface ........................................................................ xxiii
Manual Organization ..................................................... xxiv
Related Documentation ................................................ xxvi
   TIBCO Product Documentation .................................. xxvi
Typographical Conventions ............................................. xxviii
How to Contact TIBCO Customer Support ......................... xxxi

Chapter 1  Do This First—Administrator’s Checklist. ..................... 1
Install the Rendezvous Software ........................................ 2
Install License Tickets .................................................. 3
Enable Access to Executable Binary Files. ............................ 4
   Microsoft Windows Platforms .................................... 4
   UNIX Platforms ..................................................... 4
   VMS Platforms ..................................................... 4
Add Service Entries ..................................................... 5
   UNIX Platforms ..................................................... 5
   Microsoft Windows Platforms .................................... 5
   VMS Platforms ..................................................... 5
Enable Packet Checksums .............................................. 6
Arrange Internetwork Communications ................................ 7
Register Windows Services ............................................. 8
File Descriptor Limits ................................................. 9
   UNIX Resource Inheritance ....................................... 9

Chapter 2  Licensing Information .......................................... 11
Licensing Overview ................................................... 12
License Ticket Types ................................................... 12
Chapter 3  Network Details  ................................................................. 17

Transport Parameters ........................................................................ 18
Service Selection ................................................................................ 20
Interaction between Service and Network Parameters ....................... 20
Specifying the UDP or PGM Service .................................................... 21
Network Selection .............................................................................. 23
Constructing the Network Parameter .................................................. 23
Multicast Addressing .......................................................................... 25
Limitation on Computers with Multiple Network Interfaces ............... 25
Daemon Client Socket—Establishing Connections ................................ 28
Specifying a Local Daemon .................................................................. 28
Remote Daemon .................................................................................. 29
POSIX Local IPC Sockets ..................................................................... 30
Default Port and Service Numbers ...................................................... 32
Browser Administration Interface ....................................................... 32
Service Ports ...................................................................................... 33
Listen Ports ......................................................................................... 33
Reliability and Message Retention Time ............................................. 34
Factory Default .................................................................................. 34
Using a Non-Default Reliability Interval ............................................. 34
Changing the Reliability Interval at a Daemon ...................................... 36
Changing the Reliability Interval within an Application Program .......... 36
Service Reliability Rule ....................................................................... 36
Recomputing the Reliability ................................................................ 37
Disabling Multicast ............................................................................ 38

Chapter 4  Rendezvous Daemon (rvd) .................................................. 41
rvd ....................................................................................................... 42
Retransmission Control ...................................................................... 50
Reusing Service Ports ......................................................................... 52
Motivation .......................................................................................... 52
Enabling Service Reuse ....................................................................... 52
Inbox Port ............................................................................................ 53
Migration from Earlier Releases ......................................................... 53
# Chapter 5 Routing Daemon (rvrd)

Routing Daemon Overview .................................................. 77
Concepts ........................................................................... 78
  Goal ............................................................................... 78
  Connections .................................................................. 78
  Local Network ............................................................... 78
  Neighbor ......................................................................... 79
  Route ............................................................................ 79
Requirements ................................................................. 80
  Routing Daemons ............................................................ 80
  Neighbor Connections ...................................................... 80
  Subject Gating ............................................................... 80
  Subject Interest ............................................................. 80
Restricting Message Flow .............................................. 81
  Restricting Messages by Service or Port ......................... 81
  Restricting Messages by Subject Name ............................ 81
rvrd Process ................................................................. 82
  Initial State .................................................................. 82
  Administrative Store File ................................................ 82
  HTTP Administration Interface ...................................... 82
  Logging ......................................................................... 82
  Routing Table Entry ....................................................... 82
Routing Table Entry .......................................................... 83
### Contents

<table>
<thead>
<tr>
<th>Subject Map</th>
<th>214</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managing Network Topology with RVDM</td>
<td>212</td>
</tr>
<tr>
<td>Component Architecture</td>
<td>210</td>
</tr>
<tr>
<td>Backward Compatibility</td>
<td>211</td>
</tr>
<tr>
<td>Behavioral Differences</td>
<td>178</td>
</tr>
<tr>
<td>Asymmetric Multicast</td>
<td>218</td>
</tr>
<tr>
<td>Semantics of Subject Maps</td>
<td>216</td>
</tr>
<tr>
<td>Example</td>
<td>215</td>
</tr>
<tr>
<td>Mixed Environment—Subject Maps</td>
<td>220</td>
</tr>
<tr>
<td>Certificate List</td>
<td>206</td>
</tr>
<tr>
<td>Certificate Uses</td>
<td>204</td>
</tr>
<tr>
<td>Existing Users</td>
<td>200</td>
</tr>
<tr>
<td>Add a New User</td>
<td>200</td>
</tr>
<tr>
<td>Default Network and Service</td>
<td>197</td>
</tr>
<tr>
<td>Default Network and Service</td>
<td>195</td>
</tr>
<tr>
<td>Administrator and Password</td>
<td>195</td>
</tr>
<tr>
<td>General Information</td>
<td>193</td>
</tr>
<tr>
<td>Daemon Parameters</td>
<td>195</td>
</tr>
<tr>
<td>Certificate Uses</td>
<td>204</td>
</tr>
<tr>
<td>DAEMON Certificates</td>
<td>175</td>
</tr>
<tr>
<td>Core-Dump Files</td>
<td>175</td>
</tr>
<tr>
<td>Core-Dump Files</td>
<td>175</td>
</tr>
<tr>
<td>Daemon Certificates</td>
<td>175</td>
</tr>
<tr>
<td>CA-Signed Certificates</td>
<td>175</td>
</tr>
<tr>
<td>Passwords</td>
<td>177</td>
</tr>
<tr>
<td>rvsd</td>
<td>180</td>
</tr>
<tr>
<td>rvsrd</td>
<td>184</td>
</tr>
<tr>
<td>Store Files</td>
<td>175</td>
</tr>
<tr>
<td>Core-Dump Files</td>
<td>175</td>
</tr>
<tr>
<td>Daemon Certificates</td>
<td>175</td>
</tr>
<tr>
<td>CA-Signed Certificates</td>
<td>175</td>
</tr>
<tr>
<td>Passwords</td>
<td>177</td>
</tr>
<tr>
<td>Browser Administration Interface—rvsd and rvsrd</td>
<td>188</td>
</tr>
<tr>
<td>Navigation</td>
<td>189</td>
</tr>
<tr>
<td>Navigation</td>
<td>189</td>
</tr>
<tr>
<td>Daemon Parameters</td>
<td>193</td>
</tr>
<tr>
<td>Daemon Parameters</td>
<td>195</td>
</tr>
<tr>
<td>Administrator and Password</td>
<td>195</td>
</tr>
<tr>
<td>Default Network and Service</td>
<td>197</td>
</tr>
<tr>
<td>Default Network and Service</td>
<td>195</td>
</tr>
<tr>
<td>Users</td>
<td>199</td>
</tr>
<tr>
<td>Users</td>
<td>199</td>
</tr>
<tr>
<td>Add a New User</td>
<td>200</td>
</tr>
<tr>
<td>Add a New User</td>
<td>200</td>
</tr>
<tr>
<td>Existing Users</td>
<td>200</td>
</tr>
<tr>
<td>Existing Users</td>
<td>200</td>
</tr>
<tr>
<td>Authorize Network and Service Pairs</td>
<td>201</td>
</tr>
<tr>
<td>Authorize Subjects</td>
<td>202</td>
</tr>
<tr>
<td>Certificates</td>
<td>204</td>
</tr>
<tr>
<td>Certificates</td>
<td>204</td>
</tr>
<tr>
<td>Certificate List</td>
<td>206</td>
</tr>
<tr>
<td>Certificate List</td>
<td>206</td>
</tr>
</tbody>
</table>

**Chapter 7 Daemon Manager**

<p>| Overview of RVDM               | 210 |
| Overview of RVDM               | 210 |
| Component Architecture         | 210 |
| Backward Compatibility         | 211 |
| Managing Network Topology with RVDM | 212 |
| Managing Network Topology with RVDM | 212 |
| Motivation                     | 212 |
| Network Topology Tools         | 212 |
| Transparency                   | 213 |
| Subject Map                    | 214 |
| Specification                  | 214 |
| Example                        | 215 |
| Semantics of Subject Maps      | 216 |
| Asymmetric Multicast           | 218 |
| Mixed Environment—Subject Maps | 220 |</p>
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-Network Messages</td>
<td>221</td>
</tr>
<tr>
<td>Deleting a Subject Map—the Point-to-Point Back Channel</td>
<td>222</td>
</tr>
<tr>
<td>Port Map</td>
<td>223</td>
</tr>
<tr>
<td>Service Pairs</td>
<td>223</td>
</tr>
<tr>
<td>Permitted &amp; Forbidden Service Pairs</td>
<td>225</td>
</tr>
<tr>
<td>Caution</td>
<td>226</td>
</tr>
<tr>
<td>Mixed Environment—Port Map</td>
<td>226</td>
</tr>
<tr>
<td>RVDM Server Operating Details</td>
<td>228</td>
</tr>
<tr>
<td>Daemon List</td>
<td>228</td>
</tr>
<tr>
<td>Configuration Updates</td>
<td>228</td>
</tr>
<tr>
<td>Files</td>
<td>228</td>
</tr>
<tr>
<td>Administrative Tasks</td>
<td>230</td>
</tr>
<tr>
<td>Establish a Control Channel</td>
<td>230</td>
</tr>
<tr>
<td>Start Server</td>
<td>230</td>
</tr>
<tr>
<td>Configure the Server</td>
<td>230</td>
</tr>
<tr>
<td>Start Managed Daemons</td>
<td>230</td>
</tr>
<tr>
<td>Forward RVDM Protocol Messages across Network Boundaries</td>
<td>231</td>
</tr>
<tr>
<td>Modifying the Subject Maps or Port Map</td>
<td>232</td>
</tr>
<tr>
<td>rvdm</td>
<td>233</td>
</tr>
<tr>
<td>Browser Administration Interface— RVDM Control Center</td>
<td>236</td>
</tr>
<tr>
<td>User Login</td>
<td>237</td>
</tr>
<tr>
<td>Navigation Tabs</td>
<td>238</td>
</tr>
<tr>
<td>Home Tab</td>
<td>239</td>
</tr>
<tr>
<td>Port Map Tab</td>
<td>240</td>
</tr>
<tr>
<td>Subject Maps Tab</td>
<td>243</td>
</tr>
<tr>
<td>Update</td>
<td>247</td>
</tr>
<tr>
<td>Daemons Tab</td>
<td>250</td>
</tr>
<tr>
<td>Daemon Port Statistics</td>
<td>251</td>
</tr>
<tr>
<td>Incidents Tab</td>
<td>254</td>
</tr>
<tr>
<td>Options Tab</td>
<td>255</td>
</tr>
<tr>
<td>RVDM JMX MBeans</td>
<td>259</td>
</tr>
<tr>
<td>Overview</td>
<td>259</td>
</tr>
<tr>
<td>JMX Ports</td>
<td>259</td>
</tr>
<tr>
<td>Using JConsole with RVDM</td>
<td>259</td>
</tr>
<tr>
<td>Chapter 8  Relay Agent</td>
<td>261</td>
</tr>
<tr>
<td>rvrad</td>
<td>262</td>
</tr>
<tr>
<td>Chapter 9  Rendezvous Agent (rva)</td>
<td>265</td>
</tr>
<tr>
<td>rva</td>
<td>266</td>
</tr>
</tbody>
</table>
Web Site Considerations ................................................. 274
  Home Computer and Port .................................... 274
  Isolate External from Internal. ............................. 276
HTTP Tunnel .............................................................. 277

Chapter 10 Current Value Cache .............................. 279
  Operation ................................................................. 280
  Resource Requirements ....................................... 282
  Avoid Duplicates ..................................................... 283
  Ensure Continuous Service ..................................... 284
  Crossing Network Boundaries ............................. 285
  Fault Tolerance .......................................................... 286
    Usage ................................................................. 286
  Replace and Merge ............................................... 287
  Memory-Only Mode ............................................. 289
rvcache ................................................................. 291

Chapter 11 Performance Assessment (rvperf) ............. 297
  Overview ................................................................. 298
    Components ......................................................... 298
  Principles of Operation ..................................... 299
    Listeners ............................................................. 299
  Single Mode and Automatic Mode ..................... 300
  Automatic Mode—Binary Search ......................... 301
  Dataloss Advisory .................................................. 302
  Multicast, Broadcast, Point-to-Point and Direct. .... 303
  Before You Test ................................................... 304
    Test in an Insulated Environment ...................... 304
    rvd Reliability .................................................. 304
rvperfm ................................................................. 306
rvperfs ................................................................. 311
  Interpreting the Report ..................................... 314
    Certified Delivery Agreements ......................... 316
    Elapsed Time ...................................................... 317
  Usage and Examples ........................................... 318
  Hardware Capabilities ....................................... 319
    Optimal Sustained Receive Rate ..................... 319
    Fixed Receive Rate ........................................... 320

TIBCO Rendezvous Administration
Passive Monitor ............................................................... 349
  Performance Effects .................................................... 349
The pcap Facility ............................................................ 350
  Obtaining pcap .......................................................... 350
  Packet Filtering ........................................................ 350
Selecting the Network Interface ........................................ 351
Data Capture Files ........................................................ 352
  Motivation ............................................................... 352
  Output File Rotation .................................................. 352
rvtrace ................................................................. 354
Filtering ........................................................................ 360
Interpreting the Report .................................................... 362
  General Network Load .................................................... 362
  Number of Senders ...................................................... 362
  Scanning for Problems .................................................. 363
  Bad Packets .............................................................. 364
Multicast Data Statistics .................................................. 365
  Gaps Diagnoses .......................................................... 368
Multicast Retransmit Statistics .......................................... 369
  Diagnoses ................................................................. 372
Point-to-Point Statistics .................................................. 376
  Nak Diagnoses ........................................................... 381
Subject Statistics .......................................................... 383
  Subject Table Diagnoses ................................................ 385
SNMP .......................................................................... 386
  SNMP Agent Configuration ............................................. 393

Chapter 15 Perl 5 Interface ................................................ 397
Features and Benefits ....................................................... 398
Installing the Perl 5 Interface ............................................ 399
Using the Perl 5 Interface .................................................. 400
Perl Example Programs .................................................... 401

Chapter 16 Certified Message Delivery ................................ 403
Forward RVCM Administrative Messages across Network Boundaries ........................................ 404
Ledger File Location ....................................................... 405

Chapter 17 Fault Tolerance ................................................ 407
Network Placement ........................................................ 408
Forward Fault Tolerance Messages across Network Boundaries ........................................ 409

**Chapter 18  Distributed Queues** ................................................................. 411
Forward Administrative Messages across Network Boundaries .................................. 412

**Appendix A  Windows Services** ................................................................. 415
rvntscfg .................................................................................................................. 416
rvntsreg .................................................................................................................... 417

**Appendix B  Store Files** .................................................................................. 419
Locking. ................................................................................................................... 420

**Index** ................................................................................................................ 421
<p>| Figure 1 | Disabling Multicast: Public Subjects Still Flow Among Local Clients | 39 |
| Figure 2 | Disabling Multicast: Routing Daemons | 39 |
| Figure 3 | rvd Navigation Panel | 57 |
| Figure 4 | rvd General Information Page | 59 |
| Figure 5 | rvd Clients Page | 61 |
| Figure 6 | rvd Client Information Page | 62 |
| Figure 7 | rvd Subscription List Page | 63 |
| Figure 8 | rvd Services Page | 64 |
| Figure 9 | rvd Service Information Page | 66 |
| Figure 10 | rvd Host List Page | 70 |
| Figure 11 | Managed Daemon Subject Map Summary Page | 71 |
| Figure 12 | Managed Daemon Subject Map Detail Page | 72 |
| Figure 13 | Managed Daemon Port Map Page | 74 |
| Figure 14 | Routing Daemons | 79 |
| Figure 15 | Fault Tolerance among Routing Daemons | 94 |
| Figure 16 | Path Cost and Subject Import Weight | 95 |
| Figure 17 | Routing Daemons Merge Subject Spaces | 99 |
| Figure 18 | Independent Routing Table Entries Keep Subject Spaces Separate | 100 |
| Figure 19 | Mutual Neighbors Merge Routes | 102 |
| Figure 20 | Erroneous Neighbors on the Same Network | 104 |
| Figure 21 | Routing Daemons and Duplication | 105 |
| Figure 22 | Routing Daemon WAN with Firewalls | 106 |
| Figure 23 | Bridge PGM and TRDP Networks with rvrd | 108 |
| Figure 24 | Retransmission and rvrd | 109 |
| Figure 25 | Border Router: Concepts | 112 |
| Figure 26 | Border Router: Implicit Internal First-Tier Routers | 113 |
| Figure 27 | Border Router: Second-Tier Routing Network | 114 |
| Figure 28 | Border Router: High-Fanout Network | 115 |</p>
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>Border Router: Zone Stability</td>
<td>116</td>
</tr>
<tr>
<td>30</td>
<td>Border Router: Fault Tolerance</td>
<td>117</td>
</tr>
<tr>
<td>31</td>
<td>Border Router: Isolating Enterprise Zones</td>
<td>118</td>
</tr>
<tr>
<td>32</td>
<td>Border Router: Isolating Enterprise Zones and Fault Tolerance</td>
<td>119</td>
</tr>
<tr>
<td>33</td>
<td>rvrd Navigation Panel</td>
<td>133</td>
</tr>
<tr>
<td>34</td>
<td>rvrd General Information Page</td>
<td>136</td>
</tr>
<tr>
<td>35</td>
<td>rvrd Local Networks Page</td>
<td>138</td>
</tr>
<tr>
<td>36</td>
<td>rvrd Connected Neighbors Page</td>
<td>140</td>
</tr>
<tr>
<td>37</td>
<td>rvrd Router Connection Statistics Page</td>
<td>142</td>
</tr>
<tr>
<td>38</td>
<td>rvrd Daemon Parameters Configuration Page</td>
<td>145</td>
</tr>
<tr>
<td>39</td>
<td>rvrd Routers Configuration Page</td>
<td>148</td>
</tr>
<tr>
<td>40</td>
<td>rvrd Local Network Interfaces Configuration Page</td>
<td>151</td>
</tr>
<tr>
<td>41</td>
<td>rvrd Subject Configuration (Gating) Page</td>
<td>153</td>
</tr>
<tr>
<td>42</td>
<td>rvrd Border Policy Configuration Page</td>
<td>155</td>
</tr>
<tr>
<td>43</td>
<td>rvrd Neighbor Interfaces Page—Existing</td>
<td>157</td>
</tr>
<tr>
<td>44</td>
<td>rvrd Neighbor Interface Configuration Form</td>
<td>159</td>
</tr>
<tr>
<td>45</td>
<td>rvrd Certificate Uses Form</td>
<td>163</td>
</tr>
<tr>
<td>46</td>
<td>rvrd Certificate List</td>
<td>165</td>
</tr>
<tr>
<td>47</td>
<td>rvsd—Secure Connections across Single Firewall</td>
<td>169</td>
</tr>
<tr>
<td>48</td>
<td>rvsrd—Secure Connections across Double Firewall</td>
<td>169</td>
</tr>
<tr>
<td>49</td>
<td>rvsd Navigation Panel</td>
<td>189</td>
</tr>
<tr>
<td>50</td>
<td>rvsd Navigation Panel</td>
<td>190</td>
</tr>
<tr>
<td>51</td>
<td>rvsd General Information Page</td>
<td>193</td>
</tr>
<tr>
<td>52</td>
<td>Secure Daemon Administrator and Password Area</td>
<td>195</td>
</tr>
<tr>
<td>53</td>
<td>Secure Daemon Default Network and Service</td>
<td>197</td>
</tr>
<tr>
<td>54</td>
<td>Secure Daemon Users Page</td>
<td>199</td>
</tr>
<tr>
<td>55</td>
<td>Secure Daemon Authorize Network and Service Pairs Page</td>
<td>201</td>
</tr>
<tr>
<td>56</td>
<td>Secure Daemon Authorize Subjects Page</td>
<td>202</td>
</tr>
<tr>
<td>57</td>
<td>rvsrd Certificate Uses Form</td>
<td>204</td>
</tr>
<tr>
<td>58</td>
<td>rvsrd Certificate List</td>
<td>206</td>
</tr>
<tr>
<td>59</td>
<td>RVDM Component Architecture</td>
<td>211</td>
</tr>
<tr>
<td>60</td>
<td>Asymmetric Multicast (with RVDM)</td>
<td>218</td>
</tr>
</tbody>
</table>
Figure 61  Mixed Environment—Subject Maps ......................................................... 220
Figure 62  Mixed Environment—Port Map .......................................................... 227
Figure 63  RVDM Navigation Tabs ................................................................. 238
Figure 64  RVDM Home Tab ............................................................................. 239
Figure 65  RVDM Port Map Tab ........................................................................ 240
Figure 66  RVDM Port Map Tab—Edit ............................................................... 241
Figure 67  RVDM Subject Maps Tab ................................................................. 243
Figure 68  RVDM Subject Map—Edit ................................................................. 244
Figure 69  RVDM Subject Maps—Add, Edit or Delete ....................................... 246
Figure 70  RVDM Update Progress ................................................................. 247
Figure 71  RVDM Update Report .................................................................... 249
Figure 72  RVDM Daemons Tab ....................................................................... 250
Figure 73  RVDM Daemon Statistics ............................................................... 251
Figure 74  RVDM Incidents Tab ....................................................................... 254
Figure 75  RVDM Options Tab .......................................................................... 256
Figure 76  Typical rva Web Site with Double Firewall ....................................... 275
Figure 77  Transparent Caching by rvcache ...................................................... 280
Figure 78  Query and Response with rvcache ................................................... 281
Figure 79  Report from rvperfm ...................................................................... 315
Figure 80  Report from rvperfs ...................................................................... 316
Figure 81  rvtrace Output with -addrinfo .......................................................... 363
Figure 82  rvtrace Output without -addrinfo ..................................................... 363
Figure 83  Multicast Packet Statistics .............................................................. 365
Figure 84  Multicast Retransmit Packet Statistics ............................................ 369
Figure 85  Rseq Reveals Difficulty at a Receiver ............................................. 373
Figure 86  Rseq Reveals Difficulty at a Sender ............................................... 375
Figure 87  Point-to-Point Statistics ................................................................. 376
Figure 88  Nak Indicates Faulty Network Card or Infrastructure Component .... 382
Figure 89  Multicast Subject Statistics ............................................................. 383
### Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>General Typographical Conventions</td>
<td>xxviii</td>
</tr>
<tr>
<td>Table 2</td>
<td>Syntax Typographical Conventions</td>
<td>xxix</td>
</tr>
<tr>
<td>Table 3</td>
<td>Specify UDP or PGM Service</td>
<td>21</td>
</tr>
<tr>
<td>Table 4</td>
<td>Specify Network</td>
<td>23</td>
</tr>
<tr>
<td>Table 5</td>
<td>Socket Type for Client-Daemon Communication—Daemon</td>
<td>30</td>
</tr>
<tr>
<td>Table 6</td>
<td>Socket Type for Client-Daemon Communication—Client</td>
<td>31</td>
</tr>
<tr>
<td>Table 7</td>
<td>Default HTTP Port Numbers</td>
<td>32</td>
</tr>
<tr>
<td>Table 8</td>
<td>Default UDP or PGM Service Numbers</td>
<td>33</td>
</tr>
<tr>
<td>Table 9</td>
<td>Default TCP Port Numbers</td>
<td>33</td>
</tr>
<tr>
<td>Table 10</td>
<td>Importing Wildcard Subjects</td>
<td>86</td>
</tr>
<tr>
<td>Table 11</td>
<td>Correctly Importing Wildcard Subjects</td>
<td>87</td>
</tr>
<tr>
<td>Table 12</td>
<td>Four Neighbor Interface Configuration Forms</td>
<td>160</td>
</tr>
<tr>
<td>Table 13</td>
<td>RVDM Subject Maps, Example 1</td>
<td>215</td>
</tr>
<tr>
<td>Table 14</td>
<td>RVDM: Examples of Exempt and Affected Subjects</td>
<td>221</td>
</tr>
<tr>
<td>Table 15</td>
<td>Port Map, Example 2a</td>
<td>224</td>
</tr>
<tr>
<td>Table 16</td>
<td>RVDM Subject Maps, Example 2b</td>
<td>224</td>
</tr>
<tr>
<td>Table 17</td>
<td>Port Map: Permitted &amp; Forbidden Service Pairs</td>
<td>225</td>
</tr>
<tr>
<td>Table 18</td>
<td>RVDM Files</td>
<td>228</td>
</tr>
<tr>
<td>Table 19</td>
<td>Critical Subjects for RVDM</td>
<td>231</td>
</tr>
<tr>
<td>Table 20</td>
<td>RVDM Port Map Interface</td>
<td>241</td>
</tr>
<tr>
<td>Table 21</td>
<td>RVDM Subject Maps Interface</td>
<td>243</td>
</tr>
<tr>
<td>Table 22</td>
<td>RVDM Subject Map Edit Interface</td>
<td>245</td>
</tr>
<tr>
<td>Table 23</td>
<td>RVDM Update</td>
<td>248</td>
</tr>
<tr>
<td>Table 24</td>
<td>RVDM Daemons Interface</td>
<td>250</td>
</tr>
<tr>
<td>Table 25</td>
<td>RVDM Daemon Statistics</td>
<td>252</td>
</tr>
<tr>
<td>Table 26</td>
<td>RVDM Incidents Interface</td>
<td>254</td>
</tr>
<tr>
<td>Table 27</td>
<td>RVDM Options Interface</td>
<td>256</td>
</tr>
<tr>
<td>Table 28</td>
<td>rvcache Replace and Merge</td>
<td>288</td>
</tr>
</tbody>
</table>
Table 29 rvlat Summary Output .................................................. 338
Table 30 Measuring Tools ......................................................... 342
Table 31 IPM Tools—Command Line Parameters ...................... 343
Table 32 Filter Expressions ..................................................... 360
Table 33 Multicast Packet Statistics—Column Headings .......... 366
Table 34 Multicast Retransmit Packet Statistics—Column Headings 370
Table 35 Point-to-Point Statistics—Column Headings ......... 377
Table 36 Subjects Statistics—Column Headings ............. 384
Table 37 SNMP Objects in rVTrace ............................................. 386
Table 38 SNMP Configuration Parameters ............................... 394
Table 39 Critical Subjects for Certified Delivery ............... 404
Table 40 Critical Subjects for Fault Tolerance .................... 409
Table 41 Critical Subjects for Distributed Queues .............. 412
Table 42 Critical Subjects for Fault Tolerance .................... 417
New or Modified Sections

POSIX Local IPC Sockets ................................................................. 30
UDP Buffer Size. ............................................................................. 49
IPM Transport Batch Size .............................................................. 49
Purge Statistics .............................................................................. 252
Measuring Tools for IPM ............................................................... 341
rvtrace Output with -addrinfo .......................................................... 363
rvtrace Output without -addrinfo. .................................................... 363
Multicast Packet Statistics ............................................................ 365
Multicast Packet Statistics—Column Headings ................................. 366
Multicast Retransmit Packet Statistics ............................................ 369
Multicast Retransmit Packet Statistics—Column Headings ............... 370
Rseq Reveals Difficulty at a Receiver .............................................. 373
Rseq Reveals Difficulty at a Sender ................................................. 375
Point-to-Point Statistics ................................................................. 376
Point-to-Point Statistics—Column Headings .................................... 377
Nak Diagnoses ................................................................................. 381
Preface

This manual explains administration of TIBCO Rendezvous® software and the distributed systems that use it. It is part of the documentation set for Rendezvous Software Release 8.3.0.

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

Topics

- Manual Organization, page xxiv
- Related Documentation, page xxvi
- Typographical Conventions, page xxviii
- How to Contact TIBCO Customer Support, page xxxi
Manual Organization

This book begins with important information for system administrators:

- Chapter 1, Do This First—Administrator’s Checklist, on page 1
- Chapter 2, Licensing Information, on page 11

The third chapter describes several details upon which programmers and administrators must agree for correct program operation:

- Chapter 3, Network Details, on page 17

The next several chapters describe Rendezvous components that run as executable processes. Administrators must ensure correct set-up and operation of these components:

- Chapter 4, Rendezvous Daemon (rvd), on page 41
- Chapter 5, Routing Daemon (rvrd), on page 75
- Chapter 6, Secure Daemons (rvsd and rvsrd), on page 167
- Chapter 7, Daemon Manager, on page 209
- Chapter 8, Relay Agent, on page 261
- Chapter 9, Rendezvous Agent (rva), on page 265
- Chapter 10, Current Value Cache, on page 279

These chapters describe utilities for measuring overall system capacity and performance, and for diagnosing network problems.

- Chapter 11, Performance Assessment (rverf), on page 297
- Chapter 12, Latency Assessment (rvlat), on page 327
- Chapter 13, Measuring Tools for IPM, on page 341
- Chapter 14, Protocol Monitor (rvtrace), on page 345

Many administrators use the Perl programming language for system administration tasks. A brief chapter describes the Rendezvous Perl API:

- Chapter 15, Perl 5 Interface, on page 397

Additional administrative tasks apply when distributed systems use advanced features of Rendezvous software:

- Chapter 16, Certified Message Delivery, on page 403
• Chapter 17, Fault Tolerance, on page 407
• Chapter 18, Distributed Queues, on page 411

To register Rendezvous software with the Microsoft Windows operating system, use these utilities:
• Appendix A, Windows Services, page 415
Related Documentation

This section lists documentation resources you may find useful.

TIBCO Product Documentation

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 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 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 .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 COM Reference**
  
  Detailed descriptions of each class and method in the Rendezvous COM component. Readers should already be familiar with the programming environment that uses COM and OLE automation interfaces, as well as the material in *TIBCO Rendezvous Concepts*. 
- **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 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 Installation**
  Includes step-by-step instructions for installing Rendezvous software on various operating system platforms.

- **TIBCO Rendezvous Release Notes**
  Lists new features, changes in functionality, deprecated features, migration and compatibility information, closed issues and known issues.
Typographical Conventions

The following typographical conventions are used in this manual.

Table 1  General Typographical Conventions

<table>
<thead>
<tr>
<th>Convention</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIBCO_HOME</td>
<td>All TIBCO products are installed under the same directory. This directory is referenced in documentation as TIBCO_HOME. The value of TIBCO_HOME depends on the operating system. For example, on Windows systems, the default value is C:\tibco.</td>
</tr>
<tr>
<td>TIBRV_HOME</td>
<td>TIBCO Rendezvous installs into a version-specific directory inside TIBCO_HOME. This directory is referenced in documentation as TIBRV_HOME. The value of TIBRV_HOME depends on the operating system. For example on Windows systems, the default value is C:\tibco\rv\8.3.0.</td>
</tr>
<tr>
<td>code font</td>
<td>Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example: Use MyCommand to start the foo process.</td>
</tr>
<tr>
<td>bold code font</td>
<td>Bold code font is used in the following ways:</td>
</tr>
<tr>
<td></td>
<td>• In procedures, to indicate what a user types. For example: Type admin.</td>
</tr>
<tr>
<td></td>
<td>• In large code samples, to indicate the parts of the sample that are of particular interest.</td>
</tr>
<tr>
<td></td>
<td>• In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, MyCommand is enabled: MyCommand [enable</td>
</tr>
<tr>
<td>italic font</td>
<td>Italic font is used in the following ways:</td>
</tr>
<tr>
<td></td>
<td>• To indicate a document title. For example: See TIBCO ActiveMatrix BusinessWorks Concepts.</td>
</tr>
<tr>
<td></td>
<td>• To introduce new terms For example: A portal page may contain several portlets. Portlets are mini-applications that run in a portal.</td>
</tr>
<tr>
<td></td>
<td>• To indicate a variable in a command or code syntax that you must replace. For example: MyCommand pathname</td>
</tr>
</tbody>
</table>
### Table 1  General Typographical Conventions (Cont’d)

<table>
<thead>
<tr>
<th>Convention</th>
<th>Use</th>
</tr>
</thead>
</table>
| Key combinations | Key name separated by a plus sign indicate keys pressed simultaneously. For example: Ctrl+C.  
Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q. |
|                  | The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances. |
|                  | The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result. |
|                  | The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken. |

### Table 2  Syntax Typographical Conventions

<table>
<thead>
<tr>
<th>Convention</th>
<th>Use</th>
</tr>
</thead>
</table>
| []         | An optional item in a command or code syntax.  
For example:  
MyCommand [optional_parameter] required_parameter |
| | A logical OR that separates multiple items of which only one may be chosen.  
For example, you can select only one of the following parameters:  
MyCommand para1 | param2 | param3 |
### Typographical Conventions

A logical group of items in a command. Other syntax notations may appear within each logical group.

For example, the following command requires two parameters, which can be either the pair `param1` and `param2`, or the pair `param3` and `param4`.

```
MyCommand {param1 param2} | {param3 param4}
```

In the next example, the command requires two parameters. The first parameter can be either `param1` or `param2` and the second can be either `param3` or `param4`:

```
MyCommand {param1 | param2} {param3 | param4}
```

In the next example, the command can accept either two or three parameters. The first parameter must be `param1`. You can optionally include `param2` as the second parameter. And the last parameter is either `param3` or `param4`.

```
MyCommand param1 [param2] {param3 | param4}
```

### Table 2  Syntax Typographical Conventions

<table>
<thead>
<tr>
<th>Convention</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ }</td>
<td>A logical group of items in a command. Other syntax notations may appear within each logical group. For example, the following command requires two parameters, which can be either the pair <code>param1</code> and <code>param2</code>, or the pair <code>param3</code> and <code>param4</code>. MyCommand {param1 param2}</td>
</tr>
<tr>
<td></td>
<td>In the next example, the command requires two parameters. The first parameter can be either <code>param1</code> or <code>param2</code> and the second can be either <code>param3</code> or <code>param4</code>: MyCommand {param1</td>
</tr>
<tr>
<td></td>
<td>In the next example, the command can accept either two or three parameters. The first parameter must be <code>param1</code>. You can optionally include <code>param2</code> as the second parameter. And the last parameter is either <code>param3</code> or <code>param4</code>. MyCommand param1 [param2] {param3</td>
</tr>
</tbody>
</table>
How to Contact TIBCO Customer Support

For comments or problems with this manual or the software it addresses, please contact TIBCO Support as follows.

- For an overview of TIBCO Support, and information about getting started with TIBCO Product Support, visit this site:
  http://www.tibco.com/services/support

- If you already have a valid maintenance or support contract, visit this site:
  http://support.tibco.com

Entry to this site requires a username and password. If you do not have a username, you can request one.
Chapter 1  Do This First—Administrator’s Checklist

This checklist outlines several important tasks for system administrators.

We recommend that you review this checklist when you install TIBCO Rendezvous® software on any of your networks, when you add new networks, when you add new platforms to your networks, when you reconfigure networks, and whenever users report problems with Rendezvous software.

Topics

- Install the Rendezvous Software, page 2
- Install License Tickets, page 3
- Enable Access to Executable Binary Files, page 4
- Add Service Entries, page 5
- Enable Packet Checksums, page 6
- Arrange Internetwork Communications, page 7
- Register Windows Services, page 8
- File Descriptor Limits, page 9
Install the Rendezvous Software

Install Rendezvous software at your site. The book *TIBCO Rendezvous Installation* describes the installation procedure on various hardware and operating system platforms.
Install License Tickets

If you obtained an evaluation copy of Rendezvous software, your licenses are valid for a limited period of time (usually 60 minutes). You do not need to take further action to use the evaluation license. To purchase licenses, contact TIBCO Rendezvous licensing.

If you purchased licenses for Rendezvous components, you must install them in the license ticket file (`tibrv.tkt`).

The license ticket file must be accessible by components that read them. Ensure that the execution path of each user includes the directory that contains the license ticket file.

For more information, see Chapter 2, Licensing Information, on page 11.
Enable Access to Executable Binary Files

**Microsoft Windows Platforms**

Skip this step—it is done automatically during the installation procedure.

**UNIX Platforms**

Add the Rendezvous binary directory to the execution path of each programmer and end user of Rendezvous programs. The exact directory name varies depending on where you installed Rendezvous; the installation procedure prints the correct location for your convenience (usually a name constructed like installation_point/tibco/tibrv/bin).

**VMS Platforms**

Place this command in the startup file:

```bash
$ @SYS$STARTUP:TIBRV_STARTUP.COM
```

The startup file is SYS$STARTUP:SYSTARTUP_VMS.COM.

Then ensure that the following line appears in the login.com file of each VMS programmer and end user of Rendezvous programs:

```bash
$ @TIBRV:[COM]TIBRV_SETUP.COM
```
Add Service Entries

Rendezvous transports use the service *rendezvous* as a default (when programmers do not explicitly specify a service). If *rendezvous* is not defined as a service, the secondary default is UDP port 7500, or PGM port 7550.

We recommend that you define *rendezvous* as a service name in your network database. If port 7500 is already in use on your network, you *must* define *rendezvous* as a service (designating an available port number). The examples below define *rendezvous* as port 7500, but you may use any UDP port number.

Some organizations may want to define additional services to segregate Rendezvous communications. For example, by convention, fault tolerance messages between Rendezvous components use service 7504. You may also define those services at this time. For more information, see Service Selection on page 20.

On all platforms, Rendezvous software obtains service names by calling the function `getservbyname()`. Ensure that this function returns the correct port numbers.

**UNIX Platforms**

Add these definitions to the services database:

```
rendezvous 7500/udp
rendezvous-ft 7504/udp
```

**Microsoft Windows Platforms**

On all supported Windows platforms, add these definitions to the services database:

```
rendezvous 7500/udp
rendezvous-ft 7504/udp
```

**VMS Platforms**

On VMS platforms define a service by entering these commands.

```
$ TCPIP SET SERVICE RENDEZVOUS/PORT=7500/PROTOCOL=UDP - /USER=SYSTEM/FILE=SYS$STARTUP:TIBRV_STARTUP.COM/PROCESS=TIBRV
$ TCPIP SET SERVICE RENDEZVOUS-FT/PORT=7504/PROTOCOL=UDP - /USER=SYSTEM/FILE=SYS$STARTUP:TIBRV_STARTUP.COM/PROCESS=TIBRVFT
```

These commands are for HP TCP/IP Services. If your system uses a different IP stack, consult the IP vendor documentation for equivalent commands.
Enable Packet Checksums

Rendezvous software requires that the operating system compute packet checksums. You must configure the operating system to enable packet checksums on every host computer that runs Rendezvous programs or executable components.

Most operating systems enable packet checksums by default. Nonetheless, we recommend that you ensure that this setting is still in effect.
Arrange Internetwork Communications

This step extends the Rendezvous software from intranetwork message exchange to internetwork message exchange.

- If you plan to run Rendezvous programs on a single network, you may skip this step.
- If you plan to link several networks, read Chapter 5, Routing Daemon (rvrd), on page 75.
- If you plan to link a web site, also read Chapter 9, Rendezvous Agent (rva), on page 265.

Arrange your network appropriately.
Register Windows Services

This step applies only to Microsoft Windows.

Some situations require certain Rendezvous components to start automatically. You can satisfy this requirement by registering those components with the Windows service manager. For example:

- Start a permanent `rvd` process so that remote client programs can connect to it.
- Start `rva` so Java client programs can connect to it.
- Start `rvrd` without operator intervention.

To facilitate registry with the Windows service manager, the Rendezvous `bin` directory includes the utility programs `rvntscfg.exe` and `rvntsreg.exe`.

See Also

`rvntscfg` on page 416
`rvntsreg` on page 417
File Descriptor Limits

On UNIX, VMS, IBM i and z/OS platforms, the operating system can limit the maximum number of file descriptors per process, as well as the total number of file descriptors summed over all processes. Because each connection uses a file descriptor, this limitation in turn limits the capacity of Rendezvous components:

- In `rvd` and its variants, it limits the maximum number of client connections (that is, transports) that a daemon can accept.
- In `rved` and its variants, it limits the combined total of neighbor connections and client connections.
- In `rva`, it limits the maximum number of client connections.
- In client programs, it limits the number of transport objects.

Symptoms

When operating system file descriptor limits are set too low, Rendezvous components might report errors indicating that too many files are open, or that file descriptor limits have been exceeded. In many situations, you can eliminate this problem by raising the limit.

VMS limits the number of open sockets, rather than file descriptors. The consequent limitations, symptoms and remedy are analogous.

UNIX Resource Inheritance

In a UNIX environment, when a Rendezvous client program automatically starts a daemon process, the daemon becomes a child process of the client. As a child, the daemon inherits all the file descriptors and sockets that are open in the client process (for example, the descriptor associated with an open log file). Even after the client process (parent) closes the resource and exits, the resource remains open in the daemon process (child) until the daemon exits. Furthermore, the operating system does not release the space associated with a file that was open in the parent process until the child process exits.

For this reason we do not recommend using the auto-start feature in production environments. Consider the following solutions:

- Start daemons explicitly (instead of relying on the client to automatically start its daemon).

  For a tactic that might be helpful in implementing this solution, see Suppress Daemon Auto-Start on page 111 in TIBCO Rendezvous Concepts.
• Ensure that clients auto-start the daemon before opening any file descriptors or sockets. In this way, the daemon does not inherit these resources (other than stderr).

• Create a script that explicitly closes the inherited resources before starting the daemon.

  The auto-start feature calls an executable file named rvd. You can substitute your own script named rvd that explicitly closes inherited resources.
Chapter 2  Licensing Information

Administrators must arrange licenses and license tickets for each host computer.

**Licensing Framework**

In Rendezvous release 6 (and later) we changed the details of licensing and license tickets. Please read this chapter to familiarize yourself with the new framework.

**Topics**

- Licensing Overview, page 12
- License Ticket File, page 13
- Purchasing License Tickets, page 15
Licensing Overview

License tickets authenticate and activate licenses.

Each of these components is licensed separately—you may receive separate license tickets for each component:

- Rendezvous daemon (rvd, rvsd)
- Rendezvous routing daemon (rvrd, rvsrd)

  Running rvrd as a fault-tolerant pair consumes two rvrd licenses—one for each member of the fault tolerance group.

- Rendezvous agent (rva)
- Rendezvous current value cache (rvcache)
- Rendezvous protocol monitor (rvtrace)

License Ticket Types

Standard licenses assign a separate license ticket to each host computer.

A site license assigns a single license ticket for a very large site; each host checks its own ticket file, which contains a copy of the site ticket.

Temporary licenses permit customer evaluation; they expire, either on a fixed calendar date, or a specific number of minutes after starting.
License Ticket File

Each host computer must keep a license ticket file. The file `tibrv.tkt` must contain a ticket for each daemon component that runs on that computer.

To put new license tickets into effect, stop and restart the licensed components.

Ticket File Syntax

- Each line in the ticket file represents one license ticket.
- A license ticket consists of four fields, delimited by whitespace.
- The first field is the component or product name, such as `rvd`, or `rvrd`.
- The second field is the serial number of the license.
- The third field specifies the expiration of the license, in one of these formats:
  - `never` (license never expires)
  - `2005-11-01` (license expires on a specific date; in this example, November 1, 2005)
  - `start+60` (license expires 60 minutes after start time)
- The fourth field is a digital signature that validates the license.
- Programs that read the license ticket file ignore empty lines, extra whitespace, and anything to the right of a `#` character (comments).
- All fields are case sensitive.

Finding the License Ticket File

The license ticket file must be accessible by all Rendezvous daemon components. Ensure that the execution path of each user includes the directory that contains the license ticket file.

If a component cannot find an appropriate license ticket, the component runs for no more than 10 minutes.

UNIX Platforms

On UNIX platforms, the Rendezvous daemon searches for the license ticket file (`tibrv.tkt`) in the directories listed in the `PATH` environment variable. You must ensure that the `PATH` variable includes the `bin` subdirectory (of the Rendezvous directory).
In releases 8.0.0 and later, the Rendezvous installation procedure no longer installs a file containing temporary license tickets.

**Windows Platforms**

On Microsoft Windows platforms, the Rendezvous daemon searches for the license ticket file (`tibrv.tkt`) in the directories listed in the `PATH` environment variable. You must ensure that the `PATH` variable includes the location of the ticket file.

In releases 8.0.0 and later, the Rendezvous installation procedure no longer installs a file containing temporary license tickets.

**VMS Platforms**

On VMS platforms, the Rendezvous daemon searches for the license ticket file (`tibrv.tkt`) in two ways.

- If you do not define a `PATH` specifier, then the daemon searches for the ticket file in the current default directory. When a Rendezvous client program starts the daemon automatically, or when the system startup file starts the daemon, then the default directory is `TIBRV:[BIN]`.
- If you do define a `PATH` specifier, then the daemon searches in that directory.

VMS searches for a `PATH` specifier in this order—first as a logical name, then as a local DCL symbol, then as a global DCL symbol. Consider these examples.

```
define PATH tibrv:[bin]          As a logical name.
PATH := tibrv:[bin]              As a local DCL symbol.
PATH := dka300:[tibco.tibrv.bin] As a global DCL symbol.
PATH := "/tibrv/bin/"            Using UNIX syntax.
PATH := "/dka300/tibco/tibrv/bin/" Using UNIX syntax.
```

Installing Rendezvous creates the logical name `TIBRV`; the first and fourth of these examples use that logical name.

When using UNIX syntax, the path must terminate with a slash, and must be enclosed in quotes, as in the fourth and fifth examples.

In releases 8.0.0 and later, the Rendezvous installation procedure no longer installs a sample file containing temporary license tickets.

**IBM i Platforms**

See step 7 of Post-Installation Instructions on page 43 in TIBCO Rendezvous Installation.
Purchasing License Tickets

To purchase license tickets, contact TIBCO Rendezvous Licensing (via web, email, phone or fax) and provide details of your purchase order.

Company and Contact Information

Information about your company and a contact person helps us keep in touch. We use it to send you license tickets, documentation, notices of software updates, announcements of new features, and other useful items.

- Company name.
- Your name (or the contact person to whom we should send all Rendezvous correspondence).
- Your phone number (or the number of the contact person).
- Your fax number (or the fax number of the contact person).
- Your postal address (or the mailing address of the contact person; please include the country and postal codes as applicable).
- Your e-mail address (or the address of the contact person).

License Information

License information helps us create the correct number of licenses, and issue appropriate license tickets.

- Total Rendezvous licenses for each licensed Rendezvous component (for example, rvd, rvrd, rva, rvcache, rvtracer).
Chapter 3  Network Details

Rendezvous software hides most networking details from applications programmers. In some cases (in cooperation with network and system administrators), programmers may supply three optional networking parameters—network, service and daemon—to the transport creation calls. This chapter describes those parameters for the administrator.

The book *TIBCO Rendezvous Concepts* describes these parameters in even greater detail than this chapter. See also these sections:

- Service Parameter on page 103 in *TIBCO Rendezvous Concepts*
- Network Parameter on page 107 in *TIBCO Rendezvous Concepts*
- Daemon Parameter on page 110 in *TIBCO Rendezvous Concepts*

Topics

- Transport Parameters, page 18
- Service Selection, page 20
- Network Selection, page 23
- Daemon Client Socket—Establishing Connections, page 28
- Default Port and Service Numbers, page 32
- Reliability and Message Retention Time, page 34
- Disabling Multicast, page 38
Transport Parameters

Network transport creation calls accept three parameters that govern the behavior of the Rendezvous daemon: service, network and daemon. In simple networking environments, the default values of these parameters are sufficient (in C, the program can supply NULL for all three).

Most programmers will use default values for these parameters unless advised otherwise by their network administrator. To determine whether your environment requires special treatment, consider whether any of these conditions apply:

- Several independent distributed applications run on the same network, and you must isolate them from one another (service parameter).
- Programs use the Rendezvous routing daemon, rvard, to cooperate across a WAN with programs that belong to a particular service group, and the local programs must join the same service group (service parameter).
- A Rendezvous program runs on a computer with more than one network interface, and you must choose a specific network for Rendezvous communications (network parameter).
- Computers on the network use multicast addressing to achieve even higher efficiency, and programs must specify a set of multicast groups to join (Network parameter).
- A program runs on one computer, but connects with a Rendezvous daemon process running on a different computer, and you must specify the remote daemon to support network communications (daemon parameter).
- Two programs use direct communication. Both programs must enable this feature and specify its service (service parameter).

If none of these conditions apply, then programmers can use default values for the transport parameters.

If your network environment requires special treatment for any these parameters, please notify applications programmers developing software for your environment. If your organization runs Rendezvous programs developed by a third party, consult the third-party documentation for information about network and service configuration.

In addition, certain components of Rendezvous software, local programs and third-party programs may also require special configuration:
- The Rendezvous routing daemon, \texttt{rvrd}, must specify the service and network for each local network. Exchange this information with the network administrators at each remote site.

- The Rendezvous secure daemon limits clients to communication on a set of authorized network and service pairs.

- The current value cache, \texttt{rvcache}, accepts all three transport parameters. When you configure this program, include any special values as needed.

- Many Rendezvous programs accept transport parameters on their command lines. Inform all users of any special values that apply.
Service Selection

Rendezvous daemon (rvd) processes communicate using UDP or PGM services. The service parameter instructs the Rendezvous daemon to use this service whenever it does network communications on behalf of this transport.

As a direct result, services divide the network into logical partitions. Each transport communicates over a single service; a transport can communicate only with other transports on the same service. To communicate with more than one service, a program must create more than one transport.

Interaction between Service and Network Parameters

Within each Rendezvous daemon, all the transports that use a specific service must also use the same network parameter. That is, if the service parameters resolve to the same UDP or PGM port, then the network parameter must also be identical. (This restriction extends also to routing daemons.)

For example, suppose that the program foo, on the computer named orange, has a transport that communicates on the service svc1 over the network lan1. It is illegal for any program to subsequently create a transport to that rvd on orange to communicate on svc1 over any other network—such as lan2. Once rvd binds svc1 to lan1, that service cannot send outbound broadcast messages to any other network. Attempting to illegally rebind a service to a new network fails; the transport creation call produces the status code TIBRV_INIT_FAILURE.

To work around this limitation, use a separate service for each network.

The limitation is not as severe as it might seem at first, because it only affects outbound broadcast messages.

- Point-to-point messages on the transport’s service travel on the appropriate network (as determined by the operating system) irrespective of the transport’s network parameter.
- Inbound broadcast messages on the transport’s service can arrive from any network, irrespective of the transport’s network parameter.
Specifying the UDP or PGM Service

Programs can specify the service in one of several ways, listed in order of preference in Table 3.

Table 3  Specify UDP or PGM Service (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Port number</th>
<th>When a program specifies a UDP or PGM port number, it must be a string representing a decimal integer. For example:</th>
<th>&quot;7890&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service name</td>
<td>When a program specifies a service name, the transport creation function searches the network database using getservbyname(), which searches a network database such as NIS, DNS or a flat file such as /etc/services (in some versions of UNIX).</td>
<td></td>
</tr>
<tr>
<td>Default (Non-Secure Daemons)</td>
<td>If a program does not specify a service, or it specifies null, the transport creation function searches for the service name rendezvous. If getservbyname() does not find rendezvous, the Rendezvous daemon instructs the transport creation function to use a hard default:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The TRDP daemon offers the default service 7500.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The PGM daemon offers the default service 7550.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>We strongly recommend that administrators define rendezvous as a service, especially if either the of ports 7500 or 7550 is already in use.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For example, network administrators might add the following service entry to the network database (where 7500 is the port number): rendezvous 7500/udp</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Once this entry is in the network database, programmers can conveniently specify NULL or the empty string as the service argument to create a transport that uses the default Rendezvous service.</td>
<td></td>
</tr>
<tr>
<td>Default (Secure Daemons)</td>
<td>Secure daemons use internal defaults, which must be set explicitly by the administrator; see Default Network and Service on page 197.</td>
<td></td>
</tr>
</tbody>
</table>
The TRDP and PGM variants of rvd interpret the service specification differently:

- The TRDP variant interprets it as a UDP service.
- The PGM variant interprets it as a pair of services with the same port number—a PGM service for multicast communication, and a UDP service for point-to-point communication. Even though these twin services share the same port number, data does not cross from one to the other.

You may specify both parts either as a service name or a port number.

Direct communication is not available when connecting to a remote daemon. (However, it is available when connecting to a TIBCO Messaging Appliance™ P-7500.)

For a general overview, see Direct Communication on page 116 in TIBCO Rendezvous Concepts.
Network Selection

Every network transport object communicates with other transport objects over a network. On computers with only one network interface, the Rendezvous daemon communicates on that network without further instruction from the program.

On computers with more than one network interface, the network parameter instructs the Rendezvous daemon to use a particular network for all communications involving this transport. To communicate over more than one network, a program must create a separate transport object for each network. For further details, see Limitation on Computers with Multiple Network Interfaces on page 25.

The network parameter also specifies multicast addressing details (for a brief introduction, see Multicast Addressing on page 25).

To connect to a remote daemon, the network parameter must refer to the network from the perspective of the remote computer that hosts the daemon process.

Constructing the Network Parameter

The network parameter consists of up to three parts, separated by semicolons—network, multicast groups, send address—as in these examples:

- "lan0" network only
- "lan0;225.1.1.1" one multicast group
- "lan0;225.1.1.1,225.1.1.5;225.1.1.6" two multicast groups, send address
- "lan0;;225.1.1.6" no multicast group, send address

Part One—Network

Part one identifies the network, which you can specify in several ways:

<table>
<thead>
<tr>
<th>Host name</th>
<th>When a program specifies a host name, the transport creation function calls gethostbyname(), which searches a network database to obtain the IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host IP address</td>
<td>When a program specifies an IP address, it must be a string representing a multi-part address. For example: &quot;101.120.115.111&quot;</td>
</tr>
</tbody>
</table>

Table 4 Specify Network (Sheet 1 of 2)
The use of the UDP broadcast protocol has generally been superseded by the IP multicast protocol. To use broadcast protocols without multicast addressing, specify only part one of the network parameter, and omit the remaining parts.

### Part Two—Multicast Groups

Part two is a list of zero or more multicast groups to join, specified as IP addresses, separated by commas. Each address in part two must denote a valid multicast address. Joining a multicast group enables listeners on the resulting transport to receive data sent to that multicast group.

For a brief introduction to multicasting, see Multicast Addressing on page 25.
Part Three—Send Address

Part three is a single send address. When a program sends broadcast data on the resulting transport, it is sent to this address. (Point-to-point data is not affected.) If present, this item must be an IP address—not a host name or network name. The send address need not be among the list of multicast groups joined in part two.

If you join one or more multicast groups in part two, but do not specify a send address in part three, the send address defaults to the first multicast group listed in part two.

Multicast Addressing

Multicast addressing is a focused broadcast capability implemented at the operating system level. In the same way that the Rendezvous daemon filters out unwanted messages based on service groups, multicast hardware and operating system features filter out unwanted messages based on multicast addresses.

When no broadcast messages are present on the service, multicast filtering (implemented in network interface hardware) can be more efficient than service group filtering (implemented in software). However, transports that specify multicast addressing still receive broadcast messages, so combining broadcast and multicast traffic on the same service can defeat the efficiency gain of multicast addressing.

Rendezvous software supports multicast addressing only when the operating system supports it. If the operating system does not support it, and you specify a multicast address in the network argument, then transport creation calls produce an error status (TIBRV_NETWORK_NOT_FOUND).

Limitation on Computers with Multiple Network Interfaces

On computers with more than one network interface, Rendezvous programs must not attempt to combine communications over different network interfaces using the same UDP or PGM service. To understand this limitation, consider these examples of incorrect usage in the context of multiple network interfaces.

Erroneous Examples

- A program, mylistener, creates a transport using service 7500 and network lan0; it listens to broadcast subjects using that transport. Other program processes on both lan0 and lan1 send broadcast messages using service 7500. As a result, mylistener might unexpectedly receive inbound messages from lan1.
An administrator configures the Rendezvous routing daemon on a computer with two network interfaces (lan0 and lan1) using service 7500. Since the administrator does not specify a \texttt{-network} parameter, the routing daemon uses the default network interface.

As a result, the routing daemon forwards messages from its neighbor only to the default network interface; however, it might forward messages from both lan0 and lan1 to its neighbor.

A program creates two network transports. Both use service 7500, but one uses network lan0, while the other uses network lan1.

As a result, the call to create the second transport produces an error.

Two programs on the same computer each create a transport. Both use service 7500, but one uses network lan0, while the other uses network lan1. Even though these transports are in different processes, both transports connect to the same Rendezvous daemon—which is subject to the limitation.

As a result, the program fails to create the second transport.

We recommend \textit{caution} when you deploy Rendezvous programs or Rendezvous routing daemons on any computer with multiple network interfaces.

\textbf{Source of the Limitation}

The roots of this limitation are in the underlying IP broadcast protocols. Consider this asymmetry:

- When sending an \textit{outbound} broadcast packet, IP software sends the packet on exactly one network.

  Rendezvous programs can specify this network with the transport creation function’s \texttt{network} parameter.

- In contrast, IP software collects \textit{inbound} broadcast packets from all network interfaces.

  Furthermore, when IP software presents an inbound packet to a client program (such as \texttt{rvd}) it does not include any indication of the network on which that packet arrived.

Because of this asymmetry, the actual behavior of IP broadcast protocols can be different than one might expect.

\textbf{Avoiding the Limitation}

You can use two strategies to avoid problems associated with this limitation:

- Use a separate service for Rendezvous messages on each network.
• Use multicast addressing to precisely define a range of reachable transports.

Using a separate service can rectify all four of the erroneous examples. Multicast addressing can rectify the first two erroneous examples, but not the latter two. In all four examples, a single Rendezvous daemon is sufficient.

For example, consider these two approaches to rectifying the first erroneous example:

• Separate Service

  A program, mylistener, creates a transport using service 7500 and network 1an0; it listens to broadcast subjects using that transport. Other processes on 1an0 send messages using service 7500, but all processes on 1an1 send messages using service 7510. Separating by service prevents the transport from receiving interference from 1an1.

• Multicast Addressing

  A program, mylistener, creates a transport using service 7500 and multicast network 1an0; 225.1.1.1. This transport selectively receives only those inbound multicast messages that are sent on network 1an0, to multicast address 225.1.1.1, on service 7500. Multicast addressing (where available) filters out messages sent on other networks using any other multicast address.

  However, multicast addressing does not filter out IP broadcast messages sent on the same UDP service. Once again, the roots of this limitation are in the underlying IP broadcast protocols.
Daemon Client Socket—Establishing Connections

The Rendezvous daemon (rvd) and the Rendezvous routing daemon (rvrd) both open a client socket to establish communication with their clients (Rendezvous programs). The -listen option to rvd and rvrd lets you specify where the daemon should listen for new client connections. Conversely, Rendezvous programs request connections with the daemon at that client socket. The daemon parameter of the transport creation function determines how and where to find the Rendezvous daemon and establish communication.

Each transport object establishes a communication conduit with a Rendezvous daemon, as described in these steps:

1. The daemon process opens a (TCP) client socket, and waits for a client to request a connection.

   The -listen option of the Rendezvous daemon specifies where the daemon listens for new client transports.

2. The program calls the transport creation function, which contacts the daemon at the client socket specified in its daemon parameter.

   The daemon parameter of the transport creation function must correspond to the -listen option of the daemon process; that is, they must specify the same communication type and socket number.

   If no daemon process is listening on the specified client socket, then the transport creation call automatically starts a new daemon process (which listens on the specified client socket) and then attempts to connect to it.

3. The daemon process opens a conduit for private communication with the new transport. All future communication uses that private conduit.

   The request socket is now free for additional requests from other client transports.

Specifying a Local Daemon

Specify the daemon’s client socket as a character string.

For local daemons, specify a TCP socket number; for example: "6555"

If you omit the daemon parameter of the transport creation function (in C, supply NULL), it uses 7500 as the default. Similarly, to start a daemon process using the default socket, omit the -listen option to the daemon command line.
In all cases, the communication type and socket number in the `daemon` parameter of the transport creation call must match those given to `rvd` through its `-listen` parameter.

**See Also**  
POSIX Local IPC Sockets, page 30

### Remote Daemon

In most cases, programs use a local daemon, running on the same host as the program. Certain situations require a *remote* daemon, for example:

- The program runs on a laptop computer that is not directly connected to the network. Instead, the laptop connects to a workstation on the network, and the daemon runs on that workstation.
- The program connects to a network at a remote site.

For *remote* daemons, specify two parts (introducing the remote host name as the first part):

- Remote host name.
- TCP socket number.

For example: "purple_host:6555".

Once again the communication type and socket number in the `daemon` parameter of the transport creation call must match those given to `rvd` through its `-listen` parameter. However, the `-listen` parameter still receives only a two-part argument—without a remote host name.

Direct communication is not available when connecting to a remote daemon. (However, it is available when connecting to a TIBCO Messaging Appliance P-7500.)

For a general overview, see **Direct Communication on page 116** in *TIBCO Rendezvous Concepts*.

### Barring Remote Connections

A Rendezvous daemon or routing daemon can prohibit connections from remote programs by specifying `-listen "127.0.0.1"`. The special network address `127.0.0.1` represents the local host, so this parameter specifies that only local programs may connect.

This configuration is especially important when a routing daemon runs on a firewall computer.
**POSIX Local IPC Sockets**

POSIX local IPC sockets, also known as UNIX domain sockets, are an alternative to the TCP/IP sockets that Rendezvous normally uses for communication between client and daemon. When available, IPC sockets yield faster performance than TCP/IP sockets. All other Rendezvous behavior is transparent to this choice of socket protocol.

**Availability**

IPC sockets are available only on UNIX platforms that support `AF_UNIX` or `AF_LOCAL` socket types.

**Behavior**

On UNIX platforms where IPC is available, they are the default behavior of Rendezvous (release 8.3 and later). That is, Rendezvous automatically uses IPC sockets for communication between client and daemon processes on the same host computer. To override (or force) this default behavior, you can explicitly configure either the daemon or the client (see below).

On platforms where IPC is not available, requests to use IPC always fail.

**Daemon Configuration**

The daemon’s `-listen` parameter accepts values that specify the available socket protocols.

**Table 5  Socket Type for Client-Daemon Communication—Daemon**

<table>
<thead>
<tr>
<th>Value</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tcp:port</code></td>
<td>The daemon listens only for TCP connection requests. If the operating system prevents the daemon from listening for TCP connections, the daemon exits immediately.</td>
</tr>
<tr>
<td><code>ipc:port</code></td>
<td>The daemon listens for both IPC and TCP connection requests. If the operating system prevents the daemon from listening for either TCP or IPC connections, the daemon exits immediately.</td>
</tr>
<tr>
<td><code>port</code></td>
<td>The daemon listens for both IPC and TCP connection requests. However, if the operating system prevents the daemon from listening for IPC connections, the daemon does not exit; instead, it degrades gracefully, listening only for TCP connection requests.</td>
</tr>
</tbody>
</table>

**More than One Daemon**

When starting two (or more) daemons on the same host computer, you must specify distinct `port` numbers. If one daemon is already using a `port` number, and another daemon attempts to reuse it, the second daemon exits immediately.
Client Configuration

Client programs can specify the preferred socket protocol for connecting to the daemon; specify the socket preference in the daemon parameter of the transport creation call.

Table 6  Socket Type for Client-Daemon Communication—Client

<table>
<thead>
<tr>
<th>Value</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcp:port</td>
<td>The client connects to the daemon using a TCP/IP socket.</td>
</tr>
<tr>
<td>ipc:port</td>
<td>The client connects to the daemon using an IPC socket. If the connection request fails, then the transport creation call fails too.</td>
</tr>
<tr>
<td>port</td>
<td>The client first attempts to connect to the daemon using an IPC socket; if that attempt fails, then the client attempts to connect using a TCP/IP socket.</td>
</tr>
</tbody>
</table>

IPC sockets are available only if the client and the daemon reside on the same host computer.

Daemon Auto-Start

If no daemon is listening for client connections on port, then the transport creation call attempts to start one automatically. The transport creation call replicates its own daemon argument as the new daemon’s -listen argument.
Default Port and Service Numbers

For convenient reference, these tables list port and service numbers with special meaning to Rendezvous components.

**Browser Administration Interface**

Rendezvous components use HTTP ports for browser administration interfaces.

Configurable daemons distributed with Rendezvous also open an ephemeral HTTPS port (to keep the daemon configuration secure against unauthorized modification). To find the actual HTTPS port that the operating system has assigned, check the daemon’s start banner or log file. The configurable daemons are rvrd, rvsd, rvsrd, rva and rvcache—but not rvd.

**Table 7  Default HTTP Port Numbers**

<table>
<thead>
<tr>
<th>HTTP Port</th>
<th>Rendezvous Component</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Daemons Distributed with Rendezvous</strong></td>
<td></td>
</tr>
<tr>
<td>7580</td>
<td>rvd, rvrd, rvsd, rvsrd</td>
</tr>
<tr>
<td>7581</td>
<td>rvcache</td>
</tr>
<tr>
<td>7680</td>
<td>rva</td>
</tr>
<tr>
<td><strong>Daemons Distributed with Related Products</strong></td>
<td></td>
</tr>
<tr>
<td>7880</td>
<td>rvacld</td>
</tr>
<tr>
<td>7590</td>
<td>rvtxd</td>
</tr>
</tbody>
</table>

To use the browser administration interface, point your browser at an address like this template:

```
http://host:port
```

- *host* can be the host name or IP address of the host computer where the daemon is running. In some networks, a fully qualified host name is required (for example, myhost.tibco.com).

- *port* is either the default port from *Table 7*, or the port supplied as the `-http` command line argument when starting the daemon; see Command Line Parameters on page 43.
### Service Ports

**Table 8  Default UDP or PGM Service Numbers**

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rendezvous</td>
<td>Program transport objects use these UDP or PGM services as defaults.</td>
<td></td>
</tr>
<tr>
<td>7500 (TRDP)</td>
<td>The component programs <code>rva</code> and <code>rvrad</code> follow this convention.</td>
<td></td>
</tr>
<tr>
<td>7550 (PGM)</td>
<td>For more detail, see Specifying the UDP or PGM Service on page 21.</td>
<td></td>
</tr>
<tr>
<td>rendezvous-ft</td>
<td>Fault tolerance members use these UDP or PGM services as defaults for fault-tolerance protocol communications.</td>
<td></td>
</tr>
<tr>
<td>7504</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Listen Ports

**Table 9  Default TCP Port Numbers**

<table>
<thead>
<tr>
<th>TCP Port</th>
<th>Rendezvous Component</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7500</td>
<td><code>rvd</code> and <code>rvrd</code> use this TCP port as the default to listen for new connections from program transports. Program transport objects use this TCP port as the default to establish connections to <code>rvd</code> or <code>rvrd</code>. The component programs <code>rva</code> and <code>rvrad</code> follow this convention.</td>
<td></td>
</tr>
<tr>
<td>7600</td>
<td><code>rva</code> uses this TCP port as the default to listen for new connections from Java applets. See Listen Port on page 269. <code>TibrvRvaTransport</code> objects in Java applets use this TCP port as the default to establish connections with <code>rva</code>.</td>
<td></td>
</tr>
</tbody>
</table>
Reliability and Message Retention Time

The reliability interval (or retention time) is the time interval during which a sending daemon retains outbound messages. Retaining message data allows the sending daemon to retransmit message packets upon request from another daemon process (which did not correctly receive the data). Conversely, the reliability interval also specifies the time for which a receiving daemon can request retransmission of missing data from a sender.

A related concept, the reliability window, refers to the set of outbound message data that a sending daemon has retained (and not yet discarded). Data within the reliability window is available for the sending daemon to retransmit upon request.

You can specify the reliability interval in several ways, depending on the needs and complexity of your enterprise:

- For all services of a daemon, using a factory default (60 seconds)
- For all services of a daemon, using a daemon command-line argument
- For a specific service, using a client API call

See Also For additional background information, see also Reliable Message Delivery on page 58 in TIBCO Rendezvous Concepts.

Factory Default

The factory default reliability interval is 60 seconds. In environments with low data rates, this default is sufficient for reliability and does not impose high memory requirements on the daemon process. Since it is a hard-coded default, it is simple to use, requiring no further administrative action.

Using a Non-Default Reliability Interval

Decreasing Some situations require a shorter reliability interval, in order to decrease memory requirements. For example:

- High-Speed Sender

Consider a program that sends messages at very high data rates. Using the factory default retention time, the daemon must retain 60 seconds worth of data in its process memory; this volume of data might overwhelm the host computer’s available memory. To reduce the rvd process memory requirement, consider shortening the retention time.
- Time-Sensitive Data

In some programs, data becomes obsolete before 60 seconds elapse. For example, in real-time multi-player game networks, data might become obsolete in less than one second. Retaining and retransmitting obsolete data burdens the daemons and the network without any benefit. A shorter retention time would be more consistent with application requirements, and can improve overall performance.

In some situations, it is not unreasonable to reduce retention time to zero.

**Consequences**

Decreasing retention time decreases reliability, and increases the probability of lost data. **DATALOSS** advisory messages indicate message data lost because the sending daemon no longer retains it.

**Zero**

Zero is a special value for retention time. When the retention time is zero, the sending daemon does not store message data for retransmission (nor does it retransmit packets). Conversely, when the retention time is zero, the receiving daemon does not request that sending daemons retransmit data packets.

Zero is a legal value for a daemon’s `-reliability` parameter (see below). However, zero is not a legal value when a client program requests reliability for a specific service (see below).

**Increasing**

We strongly discourage increasing the retention time beyond 60 seconds. Memory requirements increase in direct proportion to the retention time. If greater reliability is required, consider using certified delivery features instead (see **Certified Message Delivery on page 139** in *TIBCO Rendezvous Concepts*).

**Lower Value Rule**

It is *not necessary* that all daemons in a network specify the same reliability time. If a sending daemon and a receiving daemon have different reliability intervals, the lower value governs retransmission interactions between the two.

- The sender’s actual retention time follows the sender’s reliability interval.
- The receiver requests retransmissions only until the lower value of the reliability interval (either the receiver’s or the sender’s) has elapsed.

Contrast this rule with the **Service Reliability Rule** that applies among transports that connect to the *same* daemon; see below.

**Routing Daemons are Exempt**

The reliability interval affects only the messages within a local network. It does not affect the operation of `rvrd` as it transfers messages across network boundaries.
Changing the Reliability Interval at a Daemon

You can override the factory default reliability interval for a specific daemon by supplying the `-reliability` parameter on the command line that starts the daemon.

The value (measured in seconds) that you supply replaces the 60-second factory default, becoming the new default reliability interval for the daemon. It applies to all services on which the daemon communicates—except when a lower value in turn overrides it for a specific service or multicast group (see below).

Changing the Reliability Interval within an Application Program

Application programs can override both the factory default and the daemon’s `-reliability` parameter. An API call can request reliability on a specific service. Any transport can request a reliability interval. The daemon uses the requested value as one of several inputs when it computes the effective reliability interval (see below).

Each request pertains to an individual transport, and is independent of other transports on the service or within an application process. A transport that does not request a specific reliability interval implicitly requests the daemon’s governing value.

Programs can request reliability only from daemons of release 8.2 or later.

An application can request a shorter retention time than the value that governs the daemon as a whole (either the factory default or the daemons `-reliability` parameter). The daemon silently overrides calls that request a retention time longer than the daemon’s governing value.

Service Reliability Rule

Client transport objects that connect to the same daemon could specify different reliability intervals on the same service—whether by requesting a reliability value, or by using the daemon’s governing value. In this situation, the daemon resolves the difference using a method that favors more stringent reliability requirements, yet limits the maximum value to the daemon’s governing value.

1. The daemon begins by selecting the largest potential value from among all the transports on that service.

2. The daemon then compares that maximum value to the daemon’s governing value, and uses the smaller of the two as the effective reliability interval for the service (that is, for all the transports on the service). That is, the daemon’s governing value limits the maximum requested value.
Contrast this rule with the Lower Value Rule that applies between two daemons; see above.

Example 1  Service Reliability Rule

Consider a situation in which the daemon’s command line specifies 40 seconds as the \texttt{-reliability} value. Two client transports on service 7500 request reliability values of 30 seconds and 50 seconds. The daemon selects the largest value, 50, and then limits it to the daemon’s governing value of 40 seconds.

Now consider a separate situation, in which the daemon uses the factory default reliability (60 seconds) as its governing value. A client transport requests 75 seconds. The daemon limits that request to 60 seconds.

Recomputing the Reliability

Whenever a transport connects, requests reliability, or disconnects from the daemon, the daemon recalculates the reliability interval for the corresponding service, by selecting the largest value of all transports communicating on that service.

When recomputing the reliability interval would result in a shorter effective retention time, the daemon delays using the new value until after an interval equivalent to the older (longer) retention time. This delay ensures that the daemon retains message data at least as long as the effective reliability interval at the time the message is sent.
Disabling Multicast

When the command line for any of the daemon components includes \(-no-multicast\), the daemon disables multicast (and broadcast) communication. This section describes the behavior of the daemons (\texttt{rvd, rvrd, rvsd, rvsrd}) when multicasting is disabled.

API
All changes in behavior occur within the daemon. These behavior changes are transparent to Rendezvous API calls. Client programs can create transports that specify multicast addressing or PGM service, send messages to any subjects, and listen to any subjects. No changes to client programs are required.

Daemon Behavior
Disabling multicast communication changes daemon behavior in these ways:

- When a client sends a message to a public subject, the daemon does not multicast it (nor broadcast it) to the network.
- When a \textit{routing} daemon receives multicast or broadcast messages from the network, it does not forward them to other daemons within the local network.

When multicast communication is disabled, daemons continue to operate in these ways:

- Point-to-point messages continue to flow in both directions between clients and the network.
- All messages (including public subjects) flow among all the clients of the daemon.

For example, in Figure 1 on page 39, even though they connect to a daemon that has disabled multicast, clients A, B and C can still exchange public subjects among themselves. However, they can exchange only point-to-point messages with D, E and F (clients of another daemon).

- All messages (including public subjects) flow in both directions between local clients of a \textit{routing} daemon and the daemon’s neighbors.

For example, in Figure 2 on page 39, J and K are local clients of a routing daemon that has disabled multicast. Nonetheless, they can exchange public subjects with transports on net Z. In contrast, L (a client of another daemon on the same network cannot exchange public subjects with transports on net Z, nor with J and K on net Y. (All clients on nets Y and Z can exchange point-to-point messages.)

For another example, see \texttt{rvsrd on page 169}.
Figure 1  Disabling Multicast: Public Subjects Still Flow Among Local Clients

Figure 2  Disabling Multicast: Routing Daemons
Chapter 4  Rendezvous Daemon (rvd)

The Rendezvous daemon (rvd) is the background process that supports all Rendezvous communications. Distributed processes depend on it for reliable and efficient network communication. All information that travels between and among processes passes through a Rendezvous daemon when it enters or exits a host computer.

The Rendezvous daemon fills these roles:

- Route messages to program processes.
- Deliver messages reliably.
- Filter subject-addressed messages.
- Shield programs from operating system idiosyncrasies, such as low-level sockets and file descriptor limits.

The Rendezvous daemon process, rvd, starts automatically when needed, runs continuously and may exit after a period of inactivity.

For further general information about the Rendezvous daemon and reliable broadcast delivery, see The Rendezvous Daemon on page 55 in TIBCO Rendezvous Concepts.

Topics

- rvd, page 42
- Log Destination, page 54
- Browser Administration Interface—rvd, page 56
rvd

Command

Syntax

rvd  
[-http [ip_address:]http_port]
[-no-http]
[-listen [socket_protocol:]ip_address:]tcp_port]
[-no-permanent]
[-no-lead-wc | -lead-wc]
[-no-multicast]
[-reliability time]
[-max-consumer-buffer size]
[-rxc-max-loss loss]
[-rxc-recv-threshold bps]
[-rxc-send-threshold bps]
[-rvdm control_channel]
[-rvdm-reverse-asym]
[-reuse-port inbox_port]
[-logfile log_filename]
[-log-max-size size]
[-log-max-rotations n]
[-foreground]
[-udp-buffer-size size]
[-transport-batch-size size]

Purpose

The command rvd starts the Rendezvous daemon process. The Rendezvous daemon is the network I/O handler for all Rendezvous programs on a computer.

Remarks

Usually, the Rendezvous daemon (rvd) process starts automatically. When a Rendezvous program creates a transport, Rendezvous software determines whether a daemon is already listening for connections (as specified by the daemon parameter). If so, the new transport connects to that daemon. If not, it automatically starts a new daemon and connects to it.

However, when the daemon parameter of the transport creation call specifies a remote daemon, the daemon does not start automatically—you must start it manually on the remote computer.

The rvd command starts the Rendezvous daemon manually. You might do this to specify optional parameters, or a start a daemon that will accept connections from programs running on remote computers.

When started automatically by a client, rvd can also exit automatically. If rvd is not connected to any valid client transports for 2 minutes, then rvd automatically exits. However, when started manually, rvd does not exit automatically. To override this behavior, start it manually with the -no-permanent option.

The Rendezvous routing daemon (rvrd) subsumes the behavior of rvd, so it is usually not necessary to run rvd on computers that already run rvrd.

Running duplicate daemons on one computer yields no benefit, and can cause errors or decreased efficiency.
**Licenses**

To put new licenses into effect, stop and restart `rvd`. Restarting `rvd` can be manual or automatic as described above.

The first time a program transport successfully connects to the Rendezvous daemon process, `rvd` starts the clock ticking against license tickets. When a license expires, all programs receive an advisory message, and `rvd` stops delivering messages.

**IPM**

TIBCO Rendezvous® Server In-Process Module (IPM) uses many of the same parameters as `rvd`, with parallel behavior. The table of parameters below notes exceptions to this rule.

### Command Line Parameters

(Sheet 1 of 7)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `-http ip_address:http_port` | The browser administration interface accepts connections on this HTTP port. Permit administration access only through the network interface specified by this IP address.  
To limit access to a browser on the `rvd` host computer, specify `127.0.0.1` (the local host address).  
When the IP address is absent, the daemon accepts connections through any network interface on the specified HTTP port.  
If the explicitly specified port is already occupied, the program exits.  
When this parameter is entirely absent, the default behavior is to accept connections from any computer on HTTP port 7580.  
If this default port is unavailable, the operating system assigns an ephemeral port number.  
In all cases, the program prints the actual HTTP port where it accepts connections.  
This parameter is not available with IPM. |
| `-no-http` | Disable all HTTP connections, overriding `-http`.  
This parameter is not available with IPM. |
rvd (and by extension, rvd operating within the local network) opens a TCP client socket to establish communication between itself and its client programs. The `-listen` parameter specifies the TCP port where the Rendezvous daemon listens for connection requests from client programs. This `-listen` parameter of `rvd` corresponds to the `daemon` parameter of the transport creation call (they must specify the same TCP port number).

The IP address specifies the network interface through which this daemon accepts TCP connections.

To bar connections from remote programs, specify IP address 127.0.0.1 (the loopback interface).

When the IP address is absent, the daemon accepts connections from any computer on the specified TCP port.

When this parameter is entirely absent, the default behavior is to accept connections from any computer on TCP port 7500.

For more detail about the choreography that establishes conduits, see Daemon Client Socket—Establishing Connections on page 28.

This parameter does not correspond to the `service` parameter of the transport creation call—but rather to the `daemon` parameter.

This parameter is not available with IPM.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-listen tcp_port</code></td>
<td><code>rvd</code> (and by extension, <code>rvrd</code> operating within the local network) opens a TCP client socket to establish communication between itself and its client programs. The <code>-listen</code> parameter specifies the TCP port where the Rendezvous daemon listens for connection requests from client programs. This <code>-listen</code> parameter of <code>rvd</code> corresponds to the <code>daemon</code> parameter of the transport creation call (they must specify the same TCP port number).</td>
</tr>
<tr>
<td><code>-listen ip_address:tcp_port</code></td>
<td></td>
</tr>
<tr>
<td><code>-listen socket_protocol:tcp_port</code></td>
<td></td>
</tr>
<tr>
<td><code>-no-permanent</code></td>
<td>If present (or when <code>rvd</code> starts automatically), <code>rvd</code> exits after 2 minutes during which no transports are connected to it. If not present, <code>rvd</code> runs indefinitely until terminated.</td>
</tr>
<tr>
<td><code>-permanent</code></td>
<td>This flag is deprecated in release 7.0 and later. To preserve backward compatibility with existing scripts, <code>rvd</code> ignores this flag, rather than rejecting it.</td>
</tr>
</tbody>
</table>
Sending to subjects with lead wildcards (for example, > or *.foo) can cause unexpected behavior in some applications, and cause network instability in some configurations. This option lets you selectively screen wildcard sending.

When `-no-lead-wc` is present, `rvd` quietly rejects client requests to send outbound messages to subjects that contain wildcards in the lead element. `rvd` does not report excluded messages as errors.

When `-lead-wc` is present (or when neither flag is present), `rvd` allows sending messages to subjects with lead wildcards.

This parameter is not available with IPM.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-no-multicast</code></td>
<td>When present, the daemon disables multicast (and broadcast) communication. For details, see Disabling Multicast on page 38. This parameter is not available with IPM.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-reliability</code></td>
<td>Rendezvous daemons compensate for brief network failures by retaining outbound messages, and retransmitting them upon request. This parameter is one of several ways to control the message reliability interval. For a complete discussion the concept of reliability, the various ways to control it, the interaction among those ways, and reasonable values, see Reliability and Message Retention Time on page 34.</td>
</tr>
</tbody>
</table>

If this parameter is absent, `rvd` uses the factory default (60 seconds).

If this parameter is present, `rvd` (and by extension, `rvrd` operating within the local network) retains messages for `time` (in seconds). The value must be a non-negative integer.
When present, the daemon enforces this upper bound (in bytes) on each consumer buffer (the queue of messages for a client transport). When data arrives faster than the client consumes it, the buffer overflows this size limit, and the daemon discards the oldest messages to make space for new messages. The client transport receives a `CLIENT.SLOWCONSUMER` advisory.

When absent or zero, the daemon does not enforce a size limit on the consumer buffer. (However, a 60-second time limit on messages still limits buffer growth, independently of this parameter.)

This parameter is not available with IPM.

These three parameters configure the retransmission control (RXC) feature, which suppresses retransmission requests from chronically-lossy receivers.

If `-rxc-max-loss` is absent or zero, then RXC is disabled. If it is an integer in the range \([1, 100]\), it determines the maximum percentage acceptable loss rates above which a receiver is considered chronically-lossy.

- `-rxc-receive-threshold bps` configures the threshold receive rate (in bits per second) above which a chronically-lossy receiver censors its own retransmission requests. When absent, the default value is zero (always censor a chronically-lossy receiver).

- `-rxc-send-threshold bps` configures the threshold send rate (in bits per second) above which the daemon suppresses (that is, ignores requests from) chronically-lossy receivers. When absent, the default value is zero (always suppress retransmissions to chronically-lossy receivers).

For a complete explanation, see Retransmission Control on page 50.
The control_channel argument specifies the dedicated control channel over which the RVDM server coordinates with managed Rendezvous daemons. This value must denote the same control channel as configured in the RVDM server; see Control Channel on page 257. The form of the control_channel argument is:

```
port_number : [network_interface] ; multicast_address
```

- `port_number` (required) specifies a service port.
- `network_interface` (optional) specifies the network interface (for host computers with multiple interfaces). When this element is absent, the default value is the host’s primary interface.
- `multicast_address` (required) is a dedicated multicast address for RVDM control messages.

When this entire parameter is absent, the daemon operates as a non-managed daemon with respect to RVDM.

When present, the daemon operates as a managed daemon. (For a complete explanation of this feature, see Daemon Manager on page 209.)

When present, this daemon reverses the direction of the RVDM multicast groups. If RVDM instructs it to listen on group 225.1.1.1 and send on 225.2.2.2, it reverses direction and instead listens on 225.2.2.2 and sends on 225.1.1.1. For a usage scenario, see Asymmetric Multicast on page 218.

When absent, this daemon obeys multicast group designations from RVDM without alteration.
When present, other daemons on the same host computer can reuse service ports.

When absent, other daemons cannot reuse a service port that is in use by this daemon.

For correct operation, all the daemons that use a common service port on a host computer must specify this option. For background and details, see Reusing Service Ports on page 52.

The `inbox_port` argument (required) specifies the UDP port that this daemon uses for point-to-point communications. This value must be unique for each daemon (that reuses service ports) on a common host computer.

Furthermore, you must not use the `inbox_port` in any transport specification on the same host computer. When using RVDM, use care to avoid violating this restriction inadvertently. You must not use the `inbox_port` for the RVDM control channel. You must not define a subject map for the `inbox_port`. You must not specify the `inbox_port` as an effective port in the RVDM port map.

```
-logfile log_filename
```

Send log output to this file.

When absent, the default is `stderr`.

```
-log-max-size size
-log-max-rotations n
```

When present, activate the log rotation regimen (see Log Rotation on page 54).

When you specify these options, you must also specify `-logfile`.

`size` is in kilobytes. If `size` is non-zero, it must be in the range [100, 2097152]. Values outside this range are automatically adjusted to the nearest acceptable value. Zero is a special value, which disables log rotation. When `-log-max-size` is zero or absent, a single log file may grow without limit (other than the limit of available storage).

`n` indicates the maximum number of files in the rotation. When `-log-max-rotations` is absent, the default value is 10.
 Utility Scripts

You can create utilities to start Rendezvous daemons with specific command line arguments. For models, see the sample rvd scripts (or the sample Windows program rvd.c) in the Rendezvous subdirectory src/examples/utilities/.

You can use such utilities to customize daemon behavior; for example, your utilities can select 64-bit daemons, specify managed daemons, logfile parameters or reliability parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-foreground</td>
<td>Available only on UNIX platforms.</td>
</tr>
<tr>
<td></td>
<td>When present, rvd runs as a foreground process.</td>
</tr>
<tr>
<td></td>
<td>When absent, rvd runs as a background process.</td>
</tr>
<tr>
<td></td>
<td>This parameter is not available with IPM.</td>
</tr>
<tr>
<td>-udp-buffer-size size</td>
<td>UDP Buffer Size</td>
</tr>
<tr>
<td></td>
<td>When present, the daemon requests a buffer of this size (in bytes) for outbound UDP multicast. (Operating system constraints can limit this request.) The value of size must be a non-negative integer.</td>
</tr>
<tr>
<td></td>
<td>When absent or zero, the default buffer size is 16000000 bytes (16MB).</td>
</tr>
<tr>
<td></td>
<td>In most situations we recommend the default buffer size. In some situations larger buffers can yield higher throughput, at the cost of longer latency (waiting for the operating system to flush the buffer). You can use rvlat on page 333 to empirically test the effect on latency.</td>
</tr>
<tr>
<td>-transport-batch-size size</td>
<td>IPM Transport Batch Size</td>
</tr>
<tr>
<td></td>
<td>When present, enable outbound batching of data from IPM, and set the batch size (in bytes).</td>
</tr>
<tr>
<td></td>
<td>When the batch size is greater than zero, IPM transfers data to the network in batches. This option can increase throughput, at the cost of higher latency.</td>
</tr>
<tr>
<td></td>
<td>When absent, the batch size is zero, and IPM transfers data to the network immediately, for lowest latency.</td>
</tr>
<tr>
<td></td>
<td>This parameter is available only with IPM.</td>
</tr>
</tbody>
</table>
Retransmission Control

The retransmission control feature addresses the issue of excessive retransmissions, which can occur in some deployments. Unless your deployment exhibits this specific behavior, this feature can degrade performance. We discourage use of this feature except in consultation with a TIBCO professional.

Motivation

When a receiving daemon consistently misses many packets, we categorize it as a chronically-lossy receiver. This condition could indicate a host computer that is slower than other computers in the network; an overloaded host computer; a hardware problem with the NIC, connectors or cables; mismatched NIC capacity; or a network problem involving routing or switching hardware.

The effects of a chronically-lossy receiver can include wasted network bandwidth, wasted CPU resources, and decreased performance for the entire distributed application system. When a sender retransmits excessively, its data send rate can increase dramatically, which in turn can exhaust network capacity.

Retransmission control (RXC) is a feature of the daemon that can help ameliorate the adverse effects of chronically-lossy receivers, conserve network and CPU resources, and locate problem hosts. When RXC is enabled, it lets sending daemons suppress retransmissions to chronically-lossy receivers, and it lets chronically-lossy receiving daemons censor their own retransmission requests.

Identifying Excessive Loss

When RXC is enabled for a receiving daemon, the daemon tracks inbound packet loss for each service. When RXC is enabled for a sending daemon, the daemon tracks packet loss and retransmission statistics for each receiving daemon and service. To distinguish chronic loss from temporary loss, the daemons compute a set of related measurements that pinpoint receivers for which retransmission is an ineffective solution.

Max Loss

When starting a daemon, an administrator can set the \texttt{-rxc-max-loss} command line parameter (a percentage, expressed as an integer between zero and 100, inclusive). If greater than zero, then RXC is enabled; the sending daemon measures loss statistics, and compares them against the configured threshold, and against other thresholds derived from it. Using several metrics lets the daemon distinguish between temporary loss and chronic loss. If measured rates exceed their corresponding threshold values, then a receiver is categorized as chronically-lossy.

Remediation at Sender

When a sender identifies a chronically-lossy receiver, it can suppress retransmission to that receiver, with two main effects:

- The sending daemon ignores retransmission requests from that receiver; that is, the daemon does not retransmit the requested data.
The sending daemon produces an INFO advisory, indicating the chronically-lossy receiver. For details, see RETRANSMISSION.OUTBOUND.SUPPRESSED on page 281 in TIBCO Rendezvous Concepts.

Remediation at Receiver

When a receiver identifies itself as a chronically-lossy receiver, it can censor its own retransmission requests with two main effects:

- The receiving daemon does not send retransmission requests to the network.
- The receiving daemon produces an INFO advisory, indicating that it is a chronically-lossy receiver. For details, see RETRANSMISSION.INBOUND.REQUEST_NOT_SENT on page 278 in TIBCO Rendezvous Concepts.

Bandwidth Usage & Thresholds

In some deployments it might be acceptable to retransmit to chronically-lossy receivers while both the receiver’s and the sender’s bandwidth usage remain low. To configure these thresholds, set the command line parameters -rxc-receive-threshold and -rxc-send-threshold.

Send

For each service, the receiving daemon measures its inbound bandwidth usage (in bits per second). The daemon does not censor retransmission requests until its inbound bandwidth usage exceeds -rxc-receive-threshold.

Receive

For each service, the sending daemon measures its outbound bandwidth usage (in bits per second). The daemon does not suppress retransmissions to chronically-lossy receivers until its outbound bandwidth usage exceeds -rxc-send-threshold.

The default value of both threshold parameters is zero, a special value specifying that the daemons always suppress retransmissions and requests whenever RXC is enabled and chronically-lossy receivers are identified.

Other Details

Daemons compute all statistics separately for each service (see Service Selection on page 20).

The RXC feature applies only to TRDP daemons; it does not apply to the PGM variant (see PGM and TRDP on page 18 in TIBCO Rendezvous Concepts).

Routing daemons include rvd functionality; in this capacity, RXC does apply to routing daemons. However, when a routing daemon forwards messages to another routing daemon, it uses TCP protocols, and RXC does not apply.
Reusing Service Ports

In Rendezvous release 8.1.x and earlier, a service port was available only to the first daemon (on a particular host computer) that bound it. That is, client transports would fail when requesting that same service from another daemon on the same host computer.

In release 8.2 and later, you can allow daemons to reuse service ports that are already bound by another daemon (or IPM) on the same host computer. The daemon parameter `-reuse-port` enables this feature.

HP Tru64 UNIX does not support this feature.

Motivation

This section presents two situations in which reusing a service port would be advantageous.

RVDM

If managed daemons run on the same host computer (for example, on different CPU cores of a multi-core computer), and the same RVDM instance manages both daemons, then the daemons necessarily use the same control channel to communicate with RVDM. In this situation, the daemons necessarily reuse the control channel’s service port, so the daemons must each specify `-reuse-port`.

IPM

Application processes that communicate using IPM can use this feature. When several such processes must communicate on the same service, and run on one host computer, then they necessarily reuse that service, because each such process acts as its own daemon. To allow this reuse, each IPM must specify `-reuse-port`.

Enabling Service Reuse

This feature is available only for TRDP daemons (not for PGM daemons).

To enable a set of daemons on the same host computer to reuse service ports, you must explicitly enable this feature on all the daemons in the set (otherwise the behavior is undefined).
Inbox Port

While daemons on the same host can reuse service ports for broadcast or multicast messages, they cannot reuse ports for point-to-point (_INBOX) messages. For each daemon and IPM on that common host, you must designate a unique UDP port to carry point-to-point communications. (That is, you must not assign the same inbox port number to two daemons on the same host.) Supply that port as the *inbox_port* argument to the `-reuse-port` option.

Migration from Earlier Releases

⚠️ Before using this feature, you must first upgrade all the interoperating daemons and IPM libraries to release 8.2 or later.

Furthermore, all application programs that connect to daemons that actually reuse service ports must relink to Rendezvous libraries from release 8.2 or later.

Incorrect migration can result in dataloss.

RVDM

If you use RVDM to administer managed daemons that reuse service ports, you must also upgrade the daemon that serves RVDM (to release 8.2 or later).

Reusing Service Ports in Routing Daemons

Release 8.2 updates *rvrd* (and *rvsrd*) with the ability to reuse service ports, and to correctly interoperate with daemons that reuse service ports. However, note the following restriction:

⚠️ Two routing daemons on the same host cannot serve the same local network.

If two routing daemons violate this restriction, the second router to start detects the conflict, reports a configuration error, and could exit (depending on other parameters).

In contrast:

- Two routing daemons can serve the same local network, if the routing daemons run on different host computers.
- Two routing daemons can run on the same host computer, if they do not serve any local networks in common.
Log Destination

Each Rendezvous daemon and component process—rvd, rvrd, rvsd, rvsrd, rva, rvtrace—produces log output. The content of log output varies, but the semantics of command line options that affect logging are identical for all of these components:

- When all of the command line options that affect logging are absent, daemons send log output to stderr.
- When -logfile log_filename is present, daemons send log output to the file you specify, namely log_filename.
- When -log-max-size size is present and non-zero, daemons use a log rotation regimen. For details, see Log Rotation below. The parameter -log-max-rotations n determines the number files in the rotation.
- When -log-config config_log_filename is present, daemons log duplicate copies of configuration changes to the file you specify, namely config_log_filename. Daemons never rotate nor remove this file, so a permanent record of this important information remains. (This parameter is available only for rvrd, rvsd and rvsrd.)

Log Rotation

To enable log rotation, you must specify a non-zero value for -log-max-size. (When absent, the default value is zero.)

The command line option -log-max-size size limits the growth of log files. The size parameter specifies the maximum disk space (in kilobytes) that a log file can occupy (approximately) before it is rotated.

The command line option -log-max-rotations n specifies the number of log files in the rotation sequence. Notice that the storage devoted to log files can grow at most to approximately size \( \times (n+1) \).

The daemon rotates the log files according to this renaming plan:

- The parameter -logfile log_filename specifies the name of the current log file, which receives log output. For example, for rvd one might specify rvd.log (without any numerical suffix). This name also becomes the base for a sequence of rotation files.
• When the current log file reaches its maximum size, the daemon rotates log files:
  — It closes the current log file (in our example, rvd.log).
  — It appends the next available numerical suffix to the base name, and renames the full log file with that name; for example, rvd.log1. Suffixes begin with 1, and continue through n before rotating back to 1. This renaming operation overwrites any existing file from a previous rotation.
  — It opens a new current log file (once again, rvd.log).

• When the daemon terminates and restarts, logging resumes by appending to the current log file, but the rotation state is reset (that is, the first rotation overwrites the file with suffix 1).

You can determine the most recent file by comparing either packet time stamps within the files, or file modification times.

Log Rotation Backward Compatibility

The command line option -log-rotate total_size is deprecated in release 7.5, and will become obsolete in a subsequent release. We recommend migrating to the new log rotation parameters at your earliest convenience.

In the meantime, we preserve backward compatibility by converting the value of this deprecated parameter to corresponding values of the new parameters:

• -log-rotate total_size retains its old meaning—specifying the total size for all log files. The maximum size for each individual file in the rotation is total_size/10.

• If both old and new parameters are present, the new parameters take precedence (overriding the old parameter).

Current Log Page

The browser interfaces for all daemon components include a Current Log page, which displays the most recent 4 kilobytes of log output (for convenience).
Browser Administration Interface—rvd

The browser administration interface lets you control rvd from a web browser. Although rvd does not have any configurable operating parameters, you can view internal data structures that reflect network conditions.

Topics

- Navigation, page 57
- General Information, page 59
- Clients, page 61
- Services, page 64
Navigation

All browser administration interface pages display a navigation panel at the left side of the page. Use these links to display other pages.

*Figure 3  rvd Navigation Panel*
<table>
<thead>
<tr>
<th>Category</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>General Information</td>
<td>This page displays information about an <code>rvd</code> process; see <em>General Information on page 59</em>.</td>
</tr>
<tr>
<td></td>
<td>Clients</td>
<td>This page summarizes the client transports; see <em>Clients on page 61</em>.</td>
</tr>
<tr>
<td></td>
<td>Services</td>
<td>This page summarizes network services activity; see <em>Services on page 64</em>.</td>
</tr>
<tr>
<td>Daemon Manager</td>
<td>Subject Map</td>
<td>This page summarizes RVDM subject maps by service number.; see <em>Subject Map Summary on page 71</em>.</td>
</tr>
<tr>
<td></td>
<td>Port Map</td>
<td>This page displays the RVDM port map; see <em>Port Map on page 74</em>.</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Current Log</td>
<td>This page displays the most recent 4 kilobytes from the log file.</td>
</tr>
<tr>
<td></td>
<td>TIBCO Rendezvous Web Page</td>
<td>The product page from the TIBCO web site.</td>
</tr>
<tr>
<td></td>
<td>Copyright</td>
<td>The Rendezvous copyright page.</td>
</tr>
</tbody>
</table>
**General Information**

rvd (like all Rendezvous components) displays information about itself on this page.

To display this page, click **General Information** in the left margin of any page of the rvd browser administration interface.

*Figure 4  rvd General Information Page*

(Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component</td>
<td>The name of the program—rvd (or rvsd).</td>
</tr>
<tr>
<td>version</td>
<td>Version number of the program.</td>
</tr>
<tr>
<td>license ticket</td>
<td>The license ticket that validates this process.</td>
</tr>
<tr>
<td>host name</td>
<td>The hostname of the computer where the daemon process runs.</td>
</tr>
<tr>
<td>user name</td>
<td>The user who started the daemon process.</td>
</tr>
<tr>
<td>IP address</td>
<td>The IP address of the computer where the daemon process runs.</td>
</tr>
<tr>
<td>client port</td>
<td>The TCP port where the daemon listens for client connections.</td>
</tr>
<tr>
<td>network services</td>
<td></td>
</tr>
<tr>
<td>process ID</td>
<td></td>
</tr>
<tr>
<td>managed</td>
<td></td>
</tr>
<tr>
<td>control channel</td>
<td></td>
</tr>
<tr>
<td>inbox port</td>
<td></td>
</tr>
</tbody>
</table>

Notice that the daemon process can run on one computer, while you access its browser interface from another computer.
<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>network services</td>
<td>The number of network services on which this daemon’s clients communicate.</td>
</tr>
<tr>
<td>process ID</td>
<td>The operating system’s process ID number for the component.</td>
</tr>
<tr>
<td>managed</td>
<td>The value yes indicates that this daemon is managed (see Chapter 7, Daemon Manager, on page 209). The link is the name of an RVDM instance; to view or interact with the RVDM instance, click this link. The value no indicates that this daemon is a non-managed daemon.</td>
</tr>
<tr>
<td>control channel</td>
<td>The RVDM control channel specification of a managed daemon; see -rvdm on page 47. For non-managed daemons, this field displays the string disabled.</td>
</tr>
<tr>
<td>inbox port</td>
<td>When the daemon reuses service ports, this field displays the unique inbox port. When the daemon does not reuse service ports, this field displays zero.</td>
</tr>
</tbody>
</table>
Clients

rvd displays information about its clients on this page. To display this page, click Clients in the left margin of any page of the rvd browser administration interface.

Figure 5  rvd Clients Page

<table>
<thead>
<tr>
<th>Description</th>
<th>User</th>
<th>Service</th>
<th>Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>./tibrvlisten</td>
<td>dinitz</td>
<td>5238</td>
<td>0A650223.DE83CB32FF08088B40</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table rows</td>
<td>Each row of the table describes one client transport.</td>
</tr>
<tr>
<td>Description</td>
<td>The description string of the transport object. Client programs set this string using an API call.</td>
</tr>
<tr>
<td>User</td>
<td>The user name of the user that started the client program process.</td>
</tr>
<tr>
<td>Service</td>
<td>The UDP or PGM service on which the client transport communicates.</td>
</tr>
<tr>
<td>Identifier</td>
<td>A globally unique identifier for the transport object. Click this identifier to view Client Information page.</td>
</tr>
</tbody>
</table>

Client Information

This page displays additional detail about a particular client transport. To display this page, click any transport identifier in the Clients page.
### Figure 6  rvd Client Information Page

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The description string of the transport object. Client programs set this string using an API call.</td>
</tr>
<tr>
<td>User</td>
<td>The user name string of the user that started the client program process.</td>
</tr>
<tr>
<td>Service</td>
<td>The UDP or PGM service on which the client transport communicates.</td>
</tr>
<tr>
<td>Original Service</td>
<td>On a managed daemon, if the RVDM port map shunts the service port, then this number is the effective port (see Port Map on page 223).</td>
</tr>
<tr>
<td>Host</td>
<td>IP address of the client’s host computer.</td>
</tr>
<tr>
<td>Port</td>
<td>TCP port number that the daemon uses to communicate with this client.</td>
</tr>
<tr>
<td>Pid</td>
<td>Process ID of the client (on its host computer).</td>
</tr>
<tr>
<td>Serial Number</td>
<td>Serial number of the Rendezvous license ticket that validates this client connection.</td>
</tr>
<tr>
<td>Expiration</td>
<td>Expiration date of the Rendezvous license ticket.</td>
</tr>
</tbody>
</table>
This page displays additional detail about the subscriptions of a particular client transport. Each row displays the subject name of one subscription.

To display this page, click **Clients** in the **Client Information** page.

**Figure 7  rvd Subscription List Page**

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>A globally unique identifier for the transport object.</td>
</tr>
<tr>
<td>Version</td>
<td>Version number of the Rendezvous API library that this client uses.</td>
</tr>
<tr>
<td>Subscriptions</td>
<td>Number of subscriptions that this client transport has registered with rvd.</td>
</tr>
</tbody>
</table>

Click this link to view a list of the subscription subjects; see **Subscription List on page 63**.

The number of subscriptions in this field need not match the number of subjects in the subscription list. Subject mapping can expand one listener with a wildcard subject into several subscriptions in the daemon (since the wildcard can match several rows of the subject map). The subscriptions total on the Client Information page counts each match as a separate subscription, summing the subscription counts for each row of the subject map.

---

**Subscription List**

This page displays additional detail about the subscriptions of a particular client transport. Each row displays the subject name of one subscription.

To display this page, click **Clients** in the **Client Information** page.
Services

On this page `rvd` displays information about the network services it mediates for its clients.

To display this page, click Services in the left margin of any page of the `rvd` browser administration interface.

Figure 8  `rvd` Services Page

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table rows</td>
<td>Each row of the table describes one network service—that is, a UDP or PGM service on a particular network interface.</td>
</tr>
<tr>
<td>Service</td>
<td>The UDP or PGM service number. On a managed daemon, if the RVDM port map shunts the service port, then this number is the effective port (see Port Map on page 223). Clicking this number displays more detail about the service:</td>
</tr>
<tr>
<td></td>
<td>• In a non-managed daemon, clicking this number displays the Service Information page.</td>
</tr>
<tr>
<td></td>
<td>• In a managed daemon, if the service does not have a subject map, then clicking this number displays the Service Information page.</td>
</tr>
<tr>
<td></td>
<td>• In a managed daemon, if the service has a subject map, then clicking the Subject Map link in the Network column displays the Subject Map Detail page.</td>
</tr>
<tr>
<td>Network</td>
<td>The network number or multicast specification. In a managed daemon, if the service has a subject map, then this column displays the string Subject Map; clicking this link displays the Subject Map Detail page.</td>
</tr>
<tr>
<td>Hosts</td>
<td>The number of other host computers with Rendezvous daemons that communicate on this network and service.</td>
</tr>
</tbody>
</table>
(Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clients</td>
<td>The number of client transports that use this network and service.</td>
</tr>
</tbody>
</table>

**Service Information**

**Service**  
This page (see Figure 9 on page 66) displays additional detail and operational statistics about a particular network service.

To display this page, click any service number in the Services page.

**Multicast Group**  
The Group Information page displays the same details and statistics about particular multicast group.

To display the Group Information page, click any group number in the Subject Map Detail page.
## Figure 9  rvd Service Information Page

### Service Information [20000]

<table>
<thead>
<tr>
<th>Service</th>
<th>20000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>10.101.2.141;224.1.1.12;224.1.1.12</td>
</tr>
<tr>
<td>Reliability</td>
<td>5 seconds</td>
</tr>
<tr>
<td>Creation</td>
<td>2009-08-11 (13:52:14)</td>
</tr>
<tr>
<td>Clients</td>
<td>1</td>
</tr>
<tr>
<td>Hosts</td>
<td>0</td>
</tr>
<tr>
<td>Subscriptions</td>
<td>2</td>
</tr>
<tr>
<td>Communication ID</td>
<td>47171</td>
</tr>
</tbody>
</table>

### Inbound Rates (per second)

<table>
<thead>
<tr>
<th>Inbound Rates (per second)</th>
<th>Outbound Rates (per second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>msgs</td>
<td>bytes</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

### Inbound Counts

<table>
<thead>
<tr>
<th>Inbound Counts</th>
<th>Outbound Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>msgs</td>
<td>bytes</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Inbound Packet Totals

<table>
<thead>
<tr>
<th>Inbound Packet Totals</th>
<th>Outbound Packet Totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>pkts</td>
<td>missed</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Information Alerts

<table>
<thead>
<tr>
<th>Information Alerts</th>
<th>none</th>
</tr>
</thead>
</table>

TIBCO Rendezvous Administration
<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>The UDP or PGM service number.</td>
</tr>
<tr>
<td></td>
<td>On a managed daemon, if the RVDM port map shunts the service port, then this</td>
</tr>
<tr>
<td></td>
<td>number is the effective port (see Port Map on page 223).</td>
</tr>
<tr>
<td>Network</td>
<td>The network number or multicast specification.</td>
</tr>
<tr>
<td>Reliability</td>
<td>On this service, \textit{rvd} retains outbound message data for retransmission. After this interval, it discards the data. For complete details, see Reliability and Message Retention Time on page 34.</td>
</tr>
<tr>
<td>Creation</td>
<td>Date and time that this service became active.</td>
</tr>
<tr>
<td>Clients</td>
<td>The number of client transports that use this network service. To view the</td>
</tr>
<tr>
<td></td>
<td>\textit{Clients} page, click this item.</td>
</tr>
<tr>
<td>Hosts</td>
<td>The number of other host computers with Rendezvous daemons that communicate on this network and service. To view the \textit{Host List} page, click this item.</td>
</tr>
<tr>
<td>Subscriptions</td>
<td>The number of subscriptions registered with the daemon on this network service. To view the list of subscriptions, click this item.</td>
</tr>
<tr>
<td>Communication ID</td>
<td>Identifies the daemon and service. TIBCO support staff may request this value for diagnostic purposes.</td>
</tr>
<tr>
<td>Inbound Rates</td>
<td>The rate (per second) at which inbound messages, bytes and packets arrived on this network service during the most recent sampling period.</td>
</tr>
<tr>
<td>Outbound Rates</td>
<td>The rate (per second) at which the daemon sent outbound messages, bytes and packets on this network service during the most recent sampling period.</td>
</tr>
<tr>
<td>Inbound Counts</td>
<td>Cumulative statistics about inbound data messages; running totals since the start of the daemon process:</td>
</tr>
</tbody>
</table>

- \textit{msgs}—number of messages
- \textit{bytes}—number of bytes
### Outbound Counts

Cumulative statistics about outbound data messages; running totals since the start of the daemon process:

- **msgs**—number of messages
- **bytes**—number of bytes

### Inbound Packet Totals

Cumulative statistics about inbound packets; running totals since the start of the daemon process:

- **pkts**—number of data packets
- **missed**—number of missed packets (detected as a packet sequence gap)
- **lost MC**—number of multicast packets lost because the sending daemon could not retransmit them
- **lost PTP**—number of point-to-point packets lost because the sending daemon could not retransmit them
- **suppressed MC**—number of multicast packets for which the daemon suppressed the sending of retransmission requests (see Retransmission Control on page 50)
- **bad pkts**—number of unreadable data packets

These bad packets correspond to DATALOSS advisories in which the error description string includes the words multicast destination. For information about non-zero values see Mixed Environment—Subject Maps on page 220.
Services | 69

Host List

This page displays Rendezvous daemon process instances on other host computers that communicate on the same network and service. From this page, you can view the service pages of those other daemons.

To display this page, click the word hosts in the Service Information page.

This page lists any Rendezvous communications daemon host, whether the process is rvd, rvsd, rvrd, or rvsrd.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outbound Packet Totals</td>
<td>Cumulative statistics about outbound packets; running totals since the start of the daemon process:</td>
</tr>
<tr>
<td></td>
<td>• pkts—number of data packets</td>
</tr>
<tr>
<td></td>
<td>• retrans—number of packets retransmitted (multicast and point-to-point)</td>
</tr>
<tr>
<td></td>
<td>• lost MC—number of multicast packets the daemon could not retransmit (too old)</td>
</tr>
<tr>
<td></td>
<td>• lost PTP—number of point-to-point packets the daemon could not retransmit (too old)</td>
</tr>
<tr>
<td></td>
<td>• shed MC—number of multicast packets for which the daemon ignored retransmission requests from a chronically-lossy receiver, and did not retransmit the data (see Retransmission Control on page 50)</td>
</tr>
<tr>
<td></td>
<td>• bad retreqs—number of unreadable retransmission request packets</td>
</tr>
<tr>
<td>Information Alerts</td>
<td>This panel displays the 50 most recent DATALOSS advisories.</td>
</tr>
</tbody>
</table>
**Figure 10  rvd Host List Page**

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>host</strong></td>
<td>Each row of this table represents one Rendezvous daemon process and its host computer.</td>
</tr>
<tr>
<td>Hostname</td>
<td>The name of the computer where the other daemon is running.</td>
</tr>
<tr>
<td>IP Address</td>
<td>The IP address of the computer where the other daemon is running.</td>
</tr>
<tr>
<td>Version</td>
<td>The version of the Rendezvous daemon on the other host.</td>
</tr>
<tr>
<td>Serial</td>
<td>The serial number of the license for that daemon.</td>
</tr>
<tr>
<td>Uptime</td>
<td>The elapsed time that the daemon has been using the common UDP or PGM service.</td>
</tr>
</tbody>
</table>
Subject Map Summary

On this page managed daemons display a summary table of subject maps. To display this page, click Subject Map in the left margin of any page of the daemon’s browser administration interface.

For information about subject maps, see Example on page 215

Figure 11  Managed Daemon Subject Map Summary Page

<table>
<thead>
<tr>
<th>Service</th>
<th>Last Update</th>
<th>Subscriptions</th>
<th>Clients</th>
</tr>
</thead>
<tbody>
<tr>
<td>20004</td>
<td>2008-03-11 13:18:03</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20005</td>
<td>2008-03-11 13:18:03</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20006</td>
<td>2008-03-11 13:18:03</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

**Item** | **Description**
--- | ---
*table rows* | Each row of the table denotes the subject map for one service.
Service | The service number of a subject map. (If the port map is enabled, this value represents the effective service.)
Click this number to view Subject Map Detail.
Last Update | Time stamp of the most recent update.
Subscriptions | Number of subscriptions using this service. If a wildcard subscription matches several rows of the subject map, this total counts each match as a separate subscription.
Clients | Number of client transports using this service.

Subject Map Detail

This page displays the details of a particular subject map. To display this page, click any service number in the Subject Map Summary page.
Figure 12  Managed Daemon Subject Map Detail Page

Subject Map for Service 20006

| service: | 20006 |
| last update: | 2008-03-11 13:18:03 |
| clients: | 1 |

Subject-to-Multicast-Group Map

<table>
<thead>
<tr>
<th>Subject</th>
<th>Group</th>
<th>Subscribers</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOO.1</td>
<td>224.4.5.6</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Default System Groups</td>
<td>239.88.88.88</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Default User Groups</td>
<td>239.99.99.99</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

(Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service</td>
<td>The service number of this subject map. (If the port map is enabled, this value represents the effective service.)</td>
</tr>
<tr>
<td>last update</td>
<td>Time stamp of the most recent update.</td>
</tr>
<tr>
<td>clients</td>
<td>The number of client transports using this service. To view the Clients page, click this item.</td>
</tr>
<tr>
<td>table rows</td>
<td>Each row of the table describes the mapping from a subject specification to a multicast group.</td>
</tr>
<tr>
<td>Subject</td>
<td>The subject for this row of the mapping.</td>
</tr>
<tr>
<td>Group</td>
<td>The target multicast group for this row of the mapping.</td>
</tr>
<tr>
<td></td>
<td>If statistics are available for this multicast group, clicking this link displays those statistics (see Service Information on page 65).</td>
</tr>
<tr>
<td>Subscribers</td>
<td>The number of client transports that have at least one matching listener (on the service). A matching listener is a listener with a subscription subject that matches the subject of this subject mapping row.</td>
</tr>
<tr>
<td>Rank</td>
<td>Rank (relative) serves as a tie-breaker when the subject of an outbound message matches more than one subject in the mapping.</td>
</tr>
<tr>
<td>Item</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Default System Groups</td>
<td>This row describes the mapping for Rendezvous system subjects; see Subject Map on page 214.</td>
</tr>
<tr>
<td>Default User Groups</td>
<td>This row describes the default mapping for subjects that do not match the subject in any other row; see Subject Map on page 214.</td>
</tr>
</tbody>
</table>
Port Map

On this page managed daemons display the port map.

To display this page, click Port Map in the left margin of any page of the daemon’s browser administration interface.

For information about the port map, see Port Map on page 223

Figure 13  Managed Daemon Port Map Page

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>last update</td>
<td>Time stamp of the most recent update.</td>
</tr>
<tr>
<td>table rows</td>
<td>Each row of the table describes a mapping from an original port to an effective port.</td>
</tr>
<tr>
<td>Original Port</td>
<td>The original port number, as specified in a client transport.</td>
</tr>
<tr>
<td>Effective Port</td>
<td>The effective port that the daemon uses for network communication on behalf of client transports that specify the corresponding original port.</td>
</tr>
<tr>
<td></td>
<td>If the effective port number has a subject map, clicking this number displays the subject map details page (see Subject Map Detail on page 71).</td>
</tr>
<tr>
<td>Clients</td>
<td>Number of client transports affected by this mapping.</td>
</tr>
</tbody>
</table>
Proper configuration of rvrd is one of the more complex administration tasks, and it is critically important for enterprises that use Rendezvous routing daemons. We recommend that network administrators give special attention to this chapter.

Topics

- Routing Daemon Overview, page 77
- Concepts, page 78
- Requirements, page 80
- Restricting Message Flow, page 81
- rvrd Process, page 82
- Routing Table Entry, page 83
- Local Network, page 84
- Neighbors, page 89
- Adding Neighbor Interfaces, page 91
- Redundant Routing Daemons for Fault Tolerance, page 94
- Independent Routing Table Entries in One Process, page 98
- Common Topology Errors, page 103
- Security and Firewalls, page 106
- Connecting PGM and TRDP Networks with Routing Daemons, page 108
- Backlog Protection, page 120
- Idle, page 121
- Routing Daemon Logging, page 122
- rvrd, page 124
• Browser Administration Interface—rvrd, page 132
Routing Daemon Overview

Rendezvous daemons (rvd) deliver messages to programs on computers within a single network. Delivering messages beyond network boundaries requires an additional software component—Rendezvous routing daemons (rvrd).

Routing daemons efficiently connect Rendezvous programs on separate IP networks, so that messages flow between them as if on a single network. Communicating programs remain decoupled from internetwork addresses and other details.

The routing daemons forward Rendezvous messages between networks. When routing daemons are present, Rendezvous programs on one network can listen for subject names and receive messages from other networks transparently—neither the sending nor the receiving programs require any code changes. Administrators retain control over the subject names that can flow in or out of each network.

Situations

Use the routing daemon in situations where one or more of these conditions apply:

- Participating networks lie in distant geographic areas.
- Participating networks lie in nearby geographic areas, but are not connected by multicast routing hardware.
- Participating networks are separated by a firewall.
- Messages must traverse expensive or slow WAN links.

Subsumes Rendezvous Daemon

In addition to routing Rendezvous messages to and from other networks, a routing daemon process also serves its host computer as a Rendezvous daemon (rvd). It is not necessary to run a separate rvd on a computer that is already running an rvrd process.
Concepts

This section compares routing daemon software to a hardware router, using an extended analogy to introduce the operational concepts of Rendezvous routing daemons.

Goal

The goal of routing daemon software is to take Rendezvous messages from one network, and make them available on other networks. The effect is to connect a set of networks into a larger network.

Compare this goal to the goal of routing hardware—to take packets from one network, and make them available on other networks. Once again, the effect is to connect a set of networks into a larger network.

Connections

Routing daemon software uses a routing table to define connections to local networks, and to other routing daemons.

Compare this tool to a hardware router, which uses a routing table to define the connections between the router and its interfaces.

Each entry in the routing table describes one routing daemon and its connections. Although each routing daemon specifies only its own routing table entry, all the routing daemons in a WAN cooperate to share this information, so that every routing daemon builds a copy of the complete global routing table.

Local Network

A routing daemon serves a set of local networks by forwarding messages between those networks and other networks (usually, by way of other routing daemons).

While routing hardware specifies its local networks primarily in terms of network interfaces, routing daemon software specifies each local network as a pair combining network and UDP or PGM service. UDP or PGM services effectively divide the physical network into separate logical networks—even though they use the same hardware.
A routing daemon filters messages by subject name, restricting the subjects that its local networks can import and export. Filtering messages by subject in routing daemon software yields a finer granularity of control than filtering packets in a hardware router. Routing daemons control the set of subjects that each network can export to other networks, and import from other networks. For more information, see Subject Gating on page 85.

Neighbor

To achieve the goal of forwarding message between networks, routing daemons connect to other routing daemons. A routing daemon declares its potential neighbors—the other routing daemons to which it can directly connect. Two potential neighbors become actual neighbors when they establish a TCP connection.

Figure 14   Routing Daemons

Route

The set of connections through which a message travels between its originating network and its destination network is called a route. Several potential routes can exist between the originating and destination networks; routing daemons select the actual route for each message.
Requirements

These four conditions enable delivery of Rendezvous messages between networks:

Routing Daemons

A routing daemon must exist on at least one computer of each local network that participates by sending or receiving Rendezvous messages.

Neighbor Connections

The network administrator must allow the routing daemons to establish TCP or SSL connections, so the routing daemons can become neighbors.

Subject Gating

Each routing daemon must export the relevant subject names from its local network, and import the relevant subject names from other networks.

For details, see Subject Gating on page 85, and Subject Filtering with Wildcards on page 85.

Subject Interest

Import and export gating is not sufficient to start the flow of messages. To receive forwarded messages, programs within the local network must express interest in the relevant subject names, by listening for those subjects.

Whenever a routing daemon detects interest in a subject within one of its local networks, it cooperates with other routing daemons to forward that subject to that local network. When programs in the network no longer retain interest in a subject, the routing daemons stop forwarding it.

For more details, see Routing Daemons Filter Interest to Permitted Subjects on page 86.
Restricting Message Flow

Routing daemons can be very selective in allowing messages to flow between networks. Network administrators can use this selectivity in several important ways:

- Restrict sensitive information to particular networks.
- Limit the volume of messages between networks.
- Constrain information to flow in only one direction between two networks.

Restricting Messages by Service or Port

For coarse-grained control over information flow, limit communication between networks to particular UDP or PGM services.

Recall that Rendezvous programs can segregate messages by specifying the service parameter to the transport creation function. The UDP or PGM service is part of the definition of a local network; the routing daemon exchanges with its neighbors only information that arrives on the designated service.

For example, if your organization adopts a convention to send sensitive information via particular UDP or PGM services, then you can use the routing daemon to regulate (or even completely disable) the import and export of messages sent via those services.

Restricting Messages by Subject Name

For fine-grained control over all the information flowing in or out of your networks, limit communication by subject name.

Subject names specify exactly which messages may enter and leave a local network—the routing daemon blocks all other Rendezvous messages. For details, see Subject Gating on page 85.
rvrd Process

Initial State

In its initial state an rvrd process operates identically to an rvd process; it does not route messages yet.

Administrators use the browser administration interface to configure the routing daemon, and to start its operation as a software router.

Administrative Store File

An rvrd process can store its routing table entry and parameters in a file. When the process restarts, it can read that file to resume its previous operating configuration.

The format of administrative store files is not human-readable. To examine or change the routing table entry or parameters, use the browser administration interface.

Administrative store files require physical security safeguards and operating system protection. Store these files in a location that is accessible only to the system administrators who maintain it.

HTTP Administration Interface

You can configure rvrd using the browser administration interface. For more information, see Browser Administration Interface—rvrd on page 132, and the command line parameter -http for rvrd on page 124.

Logging

An rvrd process can output a log of its activity. For details, see Routing Daemon Logging on page 122.

Routing Table Entry

Each rvrd process specifies its routing table entry. For details, see Routing Table Entry on page 83.
Routing Table Entry

Routing table entries are the basic building blocks of a Rendezvous routing system. In most situations, each routing daemon process embodies a single routing table entry, which denotes that daemon throughout the WAN, and describes its operation.

In rare situations one routing daemon process can embody several routing table entries. Each entry defines a separate and independent software router, but without the cost associated with process switching. For more information, see Independent Routing Table Entries in One Process on page 98.

Combining all the routing table entries of all the routing daemons produces the global routing table. Each routing daemon uses its copy of the global routing table to forward messages efficiently to other routing daemons and their networks.

Router Name

Each routing table entry has a name. Routing daemons use these names to identify one another—so names must be unique throughout the entire WAN.

One convenient way to ensure unique names is to use the fully-qualified DNS names of the rvrd host computers; for example, frobitz.yellowNet.baz.com. (When one process embodies several routing table entries, you can use a prefix to create unique names; for example, 1.frobitz.yellowNet.baz.com).

Other naming conventions are acceptable, as long as the names are unique.

The name is a string of alphanumeric, dot, and dash characters. The maximum total length of the string is 64 characters (including the dot separators).

Local Networks

Each routing daemon can serve zero or more local networks. For details, see Local Network on page 84.

Notice that a routing daemon need not serve any local networks. In this configuration, it operates as a way station, forwarding message traffic between other routing daemons—for example, to cross a firewall. For an illustration of this role, see Security and Firewalls on page 106.

Neighbors

Each routing daemon can connect to zero or more neighbors (routing daemons on other networks). For details, see Neighbors on page 89.
Local Network

Each routing daemon can serve zero or more local networks.

Network and Service

Two parameters together define the local network:

- UDP or PGM Service
  For details, see Specifying the UDP or PGM Service on page 21.
- Network Specification
  For details, see Constructing the Network Parameter on page 23.

Local Network Name

Each local network must have a *globally unique* network name.

One convenient way to generate globally unique network names is to concatenate the UDP or PGM service, the network specification, and the DNS domain name. For example, 7500.fooNet.baz.com could refer to a local network using service 7500; in contrast, the name 7522.fooNet.baz.com would refer to the local network using service 7522 on the same physical network.

Although that naming scheme is convenient, it can sometimes adversely affect network bandwidth use. Consider using shorter unique names in these situations:

- When WAN bandwidth is severely limited.
- When the average message is very small (smaller than 50 bytes).

Like router names, each local network name is a string of alphanumeric, dot, and dash characters. The maximum total length of the string is 64 characters (including the dot separators).

⚠️ When several routing daemons serve one local network, each routing daemon *must* specify the same name for that network.

That is, if two local networks use the same physical network and the same service, then they are really the same local network. It is an error to refer to that local network with two or more different names.
Subject Gating

The router configuration determines the set of public subjects that can potentially pass between the routing daemon and the local network:

- **Export** subjects can flow out from the local network to the routing daemon, and from there to other networks.
- **Import** subjects can flow into the local network from the routing daemon.

Gating of System Subjects

As a rule, routing daemons do not forward Rendezvous system subjects (such as _RV.> and _RVRD.>). If you specify these subjects for export or import, the router ignores them.

However, system subjects required for feature operation (such as _RVCM.> and _RVFT.>) are the exception to this rule. Routing daemons do forward these subjects, and you can specify them for export and import.

Point-to-Point Gating

Routing daemons automatically transmit point-to-point messages as appropriate:

- When a routing daemon receives a point-to-point message whose destination is elsewhere in the global routing table, it forwards that message to the routing daemon that serves the destination network.
- When a routing daemon receives a point-to-point message whose destination is in one of its local networks, it forwards that message directly to rvd on the destination computer.
- Administrators do not need to explicitly import or export inbox subject names.

Subject Filtering with Wildcards

The wildcard characters, * and >, have the same semantics in import, export and exchange parameters as they do in listening calls:

- * matches any single element.
- > in the last (rightmost) position matches one or more trailing elements.
Recall that these rules of import parameter behavior apply to routing daemons, and also to the Rendezvous agent (rva).

Table 10  Importing Wildcard Subjects

<table>
<thead>
<tr>
<th>Importing this wildcard name</th>
<th>Matches messages with names like these</th>
<th>But not names like these (reason)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOO.*</td>
<td>FOO.BAR</td>
<td>FOO.BAR.BAZ (extra element)</td>
</tr>
<tr>
<td>FOO.&gt;</td>
<td>FOO.BAR</td>
<td>FOO (missing element)</td>
</tr>
<tr>
<td></td>
<td>FOO.BAR.BAZ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FOO.BAR.BAZ.BOX</td>
<td></td>
</tr>
<tr>
<td>FOO.*.&gt;</td>
<td>FOO.BAR.BAZ</td>
<td>FOO.BAR (missing element)</td>
</tr>
<tr>
<td></td>
<td>FOO.BAR.BAZ.BOX</td>
<td>FOO (missing elements)</td>
</tr>
<tr>
<td>FOO.*.STOP</td>
<td>FOO.BAR.STOP</td>
<td>FOO.STOP (missing element)</td>
</tr>
<tr>
<td></td>
<td>FOO.FOZ.STOP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FOO.BAR.BAZ (unmatched 3rd element)</td>
<td></td>
</tr>
</tbody>
</table>

Routing Daemons Filter Interest to Permitted Subjects

Routing daemons filter local listening interest according to the subjects that the local networks can import and export. The general rule is that routing daemons disregard listening interest that would include subjects in either of these categories:

- Subjects that the listener’s local network cannot import.
- Subjects that the sender’s local network cannot export.

Customers frequently deploy application programs that listen to wildcard subjects that are more inclusive than the wildcard subjects that rvrd imports or exports. As a result, the routing daemon filters this application subject interest, and the listeners do not receive any messages.

For example, consider a situation in which the local network imports FOO.> (that is, it does not permit any other subjects to enter from the WAN). When a process, L1, within the local network listens to the subject > (that is, the wildcard that matches any subject), the routing daemon first compares it to the permitted import subjects; since > is not a subset of FOO.>, the routing daemon does not forward any messages into the local network, so L1 does not receive any messages.
When a second process, L2, in the same local network, listens to the subject FOO.BAR, the routing daemon begins importing messages (because the subject matches a subject for which import is permitted); both L1 and L2 receive the imported messages.

When a third process, L3, listens to the subject FOO.>, the routing daemon widens the set of messages it imports; both L1 and L3 receive the additional message subjects.

Table 11 on page 87 summarizes this example, and presents further examples.

### Table 11  Correctly Importing Wildcard Subjects

<table>
<thead>
<tr>
<th>Importing this wildcard name</th>
<th>While listening to this subject</th>
<th>Imports these subjects</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOO.&gt;</td>
<td>&gt;</td>
<td>none</td>
<td>&gt; is more inclusive than FOO.&gt; rvrd filters out &gt; because it isn’t imported.</td>
</tr>
<tr>
<td>FOO.&gt;</td>
<td>FOO.BAR</td>
<td>FOO.BAR</td>
<td>FOO.BAR is included within FOO.&gt; rvrd filters out everything else because (for example, FOO.BAZ) no listener is requesting it.</td>
</tr>
<tr>
<td>FOO.&gt;</td>
<td>FOO.&gt;</td>
<td>FOO.&gt;</td>
<td>FOO.&gt; is identical to FOO.&gt;</td>
</tr>
<tr>
<td>A.B.&gt;</td>
<td>A.B.*</td>
<td>A.B.*</td>
<td>A.B.* is included within A.B.&gt; rvrd filters out everything else (for example, A.B.C.D) because no listener is requesting it.</td>
</tr>
</tbody>
</table>
See Also  Using Wildcards to Receive Related Subjects on page 66 in TIBCO Rendezvous Concepts

Fixed Subject Interest

The concept of fixed subject interest is obsolete in release 6 (and later). Instead, subject interest dynamically determines the set of subjects that actually flow to and from a network.

Restriction on Local Networks

Two routing daemons on the same host cannot serve the same local network. For further explanation, see Reusing Service Ports in Routing Daemons on page 53.
Neighbors

Neighbor links connect routing daemons. A routing daemon declares its potential neighbors in its routing table entry. Two routing daemons become actual neighbors when they establish a TCP connection.

To declare potential neighbors, see Neighbor Interfaces on page 157. To examine actual neighbors, see Connected Neighbors on page 140.

Neighbor Pairs

Neighbors operate in pairs—one router at each end of a neighbor connection. Administrators can specify the pairs in four ways; see Adding Neighbor Interfaces on page 91.

Local Connection Information

These parameters specify the local end of a neighbor connection.

<table>
<thead>
<tr>
<th>Local Host</th>
<th>The default value denotes the host computer’s default interface. You may override this default by specifying another network interface on the local host computer—either as a resolvable hostname, or as the IP address of the interface.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Connect Port</td>
<td>In each neighbor declaration, a routing daemon designates a TCP port number where the routing daemon accepts connection requests from that neighbor. When a routing daemon declares several neighbors, it can designate a unique local connect port for each neighbor, or some of its neighbors can share a local connect port. However, when a routing daemon process operates several routing table entries, the routing entries may not share any local connect ports.</td>
</tr>
</tbody>
</table>

Remote Connection Information

These parameters specify the remote end of a neighbor connection.

<table>
<thead>
<tr>
<th>Remote Router Name</th>
<th>In most situations, a routing daemon identifies a neighbor using its unique router name (see Router Name on page 83). (For a counterexample, see Seek Neighbor with Any Name on page 92.)</th>
</tr>
</thead>
</table>
Remote Host

Specify the location of a neighbor either as a resolvable hostname, or as the IP address of the computer in a remote network where the neighboring daemon is running.

Remote Connect Port

The remote port is the TCP port number where the remote neighbor listens for a connection request from this routing daemon.

This parameter must match the local connect port of a routing table entry within the rvrd process on the neighboring host computer.

Remote Public Certificate

When neighbors communicate using SSL, you must enter the public certificate of the authorized neighbor. For background information, see Certificates and Security on page 52 in TIBCO Rendezvous Concepts.

Network Administration

Neighboring routing daemon processes must be able to establish a TCP connection. The network administrator (at each site) must configure the hardware (or software) routers and firewalls to permit this TCP connection between the two routing daemon host computers.

Data Compression

Routing daemons can compress data to reduce the network volume that travels between neighbors.

- Compression is most useful when you pay for WAN transmission by volume.
- Compression reduces volume at the cost of speed. Compression and decompression slows rvrd processing at both ends of a neighbor link.

To enable data compression, select an appropriate option on the neighbor interface forms of both neighbors; see Add New Neighbor Interface on page 159.
Adding Neighbor Interfaces

Routing daemons can declare neighbors in four ways:

- **Active Neighbor, page 91**
- **Passive Neighbor, page 91**
- **Accept Any as Neighbor, page 92**
- **Seek Neighbor with Any Name, page 92**

To specify a potential neighbor connection, see Add New Neighbor Interface on page 159.

**Active Neighbor**

A routing daemon can declare another routing daemon as its neighbor, and actively initiate a connection to it. If the connection is broken, the routing daemon actively attempts to restore it.

Consider an example situation in which a routing daemons link several networks within an enterprise. Each routing daemon within the enterprise declares every other routing daemon as an active neighbor.

To specify an active neighbor, you must supply this information:

- **Remote Router Name**
- **Remote Host**
- **Remote Connect Port**
- **Local Connect Port**

**Passive Neighbor**

A routing daemon can declare that it passively accepts connections from its neighbor, but does not actively initiate the connection itself.

Consider these example situations:

- **Unidirectional firewall.**

Routing daemon abc.homeNet.myDom.com is located behind a firewall that allows connection requests in only one direction—outward. Active connection attempts by its neighbor, mno.lyonNet.myDom.com, would invariably fail, marking each attempt as a potential security violation at the firewall. When
mno.lyonNet.myDom.com declares abc.homeNet.myDom.com as a neighbor, it can specify passive connect, reflecting its inability to initiate a connection to abc.homeNet.myDom.com. To become actual neighbors, abc.homeNet.myDom.com must initiate the connection to mno.lyonNet.myDom.com.

- Modem restriction.

Routing daemon fgh.oshkoshNet.myDom.com is located on a host that depends on a modem for network access; the modem settings permit fgh.oshkoshNet.myDom.com to dial out, but the modem does not accept incoming calls. Active connection attempts by its neighbor, klm.chicagoNet.myDom.com, would invariably fail, while wasting resources. When klm.chicagoNet.myDom.com declares fgh.oshkoshNet.myDom.com as a neighbor, it can specify passive connect, reflecting its inability to initiate a connection to fgh.oshkoshNet.myDom.com. To become actual neighbors, fgh.oshkoshNet.myDom.com must initiate the connection to klm.chicagoNet.myDom.com.

**Specifying Passive Neighbors**

To specify a passive neighbor, you must supply this information:

- Remote Router Name
- Local Connect Port

**Accept Any as Neighbor**

Instead of declaring a specific set of neighbors, a routing daemon can declare that it accepts neighbor connections from any routing daemon.

This configuration is especially useful for hub topologies, dial-in connections, and any situation in which a routing daemon might operate with a large number of potential neighbors.

Specify the local connect port where this routing daemon accepts TCP connections from any (remote) routing daemon.

A routing daemon can simultaneously specify individual neighbors and declare that it accepts any other routing daemons as neighbors.

**Seek Neighbor with Any Name**

Instead of declaring a neighbor with a specific name, a routing daemon can seek out any available member from a set of routing daemons, without regard to its name.
This configuration is especially useful for load balancing among a set of potential neighbors with identical routes.

Specify the potential neighbors with two pieces of information:

- **Remote Host**, which must be either a DNS hostname that can resolve to more than one IP address, or a virtual IP address.
- **Remote Connect Port**—all potential neighbors must listen for connection requests on this port).

Each potential neighbor must accept connections from the seeking routing daemon, without actively attempting to connect to it. The potential neighbors can specify this in either of two ways:

- Accept connections from any neighbor, including the seeking routing daemon (see **Accept Any as Neighbor** on page 92).
- Passively accept connections specifically from the seeking routing daemon (see **Passive Neighbor** on page 91).
Redundant Routing Daemons for Fault Tolerance

Rendezvous routing daemons can cooperate for fault-tolerant service. Fault tolerance protects routing daemons against hardware failures, process failures and network segmentation.

In Figure 15, two routing daemon processes, E.Anet.moo.com and F.Anet.moo.com, run on separate host computers, and serve the local client network Anet.moo.com; similarly, routing daemons G.Bnet.moo.com and H.Bnet.moo.com both serve local client network Bnet.moo.com. Neighbor links connect E with G and H, and also F with G and H. Although these neighbor links offer redundant paths from Anet to Bnet, the routing daemons cooperate to forward each message only once. In failure situations, the routing daemons automatically readjust to continue service smoothly.

The concepts of primary and secondary do not apply to redundant routing daemons. Instead load balancing parameters govern fault-tolerant behavior (see Load Balancing on page 95).

In groups of redundant routers (such as E and F), the router names must be distinct, the local network configurations must be identical, while the load balancing parameters along neighbor links may differ.

Notice that E and F are not neighbors, nor are G and H. It would be an error for neighbors to serve the same local network (see Common Topology Errors on page 103).

Figure 15 Fault Tolerance among Routing Daemons
LoadBalancing

You can balance network load by directing messages along preferred routes. Routing daemons let you specify preferred routes using two cooperating mechanisms:

- **Path costs** determine a preference to route messages along specific neighbor links.
- **Subject import weights** determine a preference to import particular subjects into a network through a specific routing daemon.

**Example** Figure 16 repeats the fault-tolerant configuration from Figure 15—messages generally travel downward from Anet to Bnet. In this variation, the administrator specifies parameters to balance the load during normal operation—divide the message volume by subject, and direct messages along the outer links in preference to the crossover links. (In failure situations, messages continue to flow along the alternate routes.)

*Figure 16  Path Cost and Subject Import Weight*

- Path costs direct the message flow through the two outer links.
- Import weights split the traffic by subject:
  - Messages with subjects bar.> enter Bnet through routing daemon G.
  - Messages with subjects foo.> enter Bnet through routing daemon H.
Cooperating Mechanisms

Notice that effective load balancing depends on both mechanisms together. With path costs alone, all messages might flow only through F and H, while E and G have idle capacity. With subject import weights alone, all messages might flow only through F, while E has idle capacity. When both mechanisms cooperate, subject import weights divide the message volume between G and H, and path costs propagate that division back to E and F.

Path Cost

You can specify the path cost of each neighbor link. Routing protocols seek the route with the lowest cost.

For example, in Figure 16 on page 95, the outer links—between E and G, and between F and H—each specify a cost of 1. In contrast, the inner crossover links—between F and G, and between E and H—each specify a cost of 5. When all the components operate normally, messages flow across the lower cost (outer) links. When components fail, messages flow across the lowest cost link that remains operational.

Symmetric Path Costs

You must specify symmetric path costs. That is, if you specify a path cost of n at G’s neighbor link to E, the you must also specify the same path cost, n, at E’s neighbor link to G. This rule applies even when you intend that messages flow only in one direction (for example, from top to bottom in Figure 16 on page 95). Asymmetric path costs can result in unpredictable and inefficient routing behavior.

Backward Compatibility

For routing daemons from release 6, the cost of every path is always 1, and you cannot change this value. You can set a higher value for path costs only when configuring routers from release 7 or later.

See Also

To configure path cost between neighbors, see Neighbor Interfaces on page 157.

To configure path cost from a router instance to a local network, see Local Network Interfaces Configuration on page 151.

Subject Import Weight

You can specify weight values as annotations on import subject gating. When a message could travel two paths with equal cost, import weights break the tie. Routing protocols seek the path with the greatest weight.
For example, in Figure 16 on page 95, the administrator has specified that G imports foo.> with weight 1, and bar.> with weight 10. Conversely, H imports foo.> with weight 10, and bar.> with weight 1. When all the components operate properly, messages with subjects foo.> travel through H (which draws them through F), while messages with subjects bar.> travel through G (which draws them through E). If E were to fail, all messages would travel through F and H (because that route has the lowest path cost).

**See Also**
To configure subject import weight, see Subject Gating on page 153.

**Border Routing**
When rvrd is configured as a border router, then path cost and subject import weight affect only first-tier routing decisions—that is, routing within a zone. They do not affect second-tier routing decisions—that is, routing across a border between two zones. For background information about these concepts, see Border Routing on page 110.
Independent Routing Table Entries in One Process

In most situations, each routing daemon process embodies a single routing table entry. Nonetheless, in rare situations one routing daemon process can embody several routing table entries. Each entry defines a separate and independent software router, but without the cost associated with process switching.

This section explores two situations in which multiple routing table entries are appropriate:

- Overlapping Subject Space
- Bandwidth Contention on page 100

Overlapping Subject Space

Consider two distinct distributed programs that use overlapping subject spaces—that is, they use some of the same subjects for their messages. When the two programs are deployed on the same physical network, each one receives messages from the other, which is inappropriate. To eliminate interference within the network, isolate each program to a separate UDP or PGM service.

Yet this solution within one network does not ordinarily keep the subject spaces separate when routing daemons connect two or more networks, because the routing daemon merges the subject spaces of its local networks.
For example, on the left side of Figure 17, the two UDP or PGM services 7500 and 7502 effectively separate one physical network (k.foo.com) into two disjoint subject spaces; that is, program L2 cannot receive messages from program S1. Similarly, on the right side of Figure 17, two UDP or PGM services 7577 and 7588 effectively separate one physical network (J.foo.com) into two disjoint subject spaces. However, the routing daemons in this configuration merge the subject spaces of their local networks—effectively canceling the separation; that is, program L2 does receive messages from program S1.

To restore the separation, configure an independent routing table entry for each local network, as in Figure 18 on page 100.
In Figure 18, each rvrd process contains two independent routers, which act as parts of two disjoint routes—keeping the data and subject spaces separate:

- Routing table entries $A$ and $F$ form a route connecting network 2 with network 3.
- Routing table entries $B$ and $G$ form a route connecting network 1 with network 4.

Notice that once again, program L2 cannot receive messages from S1.

**Bandwidth Contention**

Bandwidth contention is the second reason to separate programs using disjoint routes.
Consider two programs that are deployed on the same physical network—a program S2 that sends messages at a moderate data rate, and a program S1 that sends messages at a relatively high data rate. However, messages from S2 are much more important to the enterprise as a whole than messages from S1.

When forwarding these messages across a WAN, routing daemons would ordinarily send them across the same TCP connection. The many unimportant messages from S1 could delay the more important messages from S2.

To solve this throughput problem, configure an independent route for each set of messages, as in Figure 18 on page 100. On the left side of Figure 18, S1 and S2 use distinct UDP or PGM services within the same physical network, effectively separating their messages into two logical network spaces. Disjoint routes carry the two sets of messages:

- Important messages from S2 travel through routing entries A and F.
- Messages from S1 travel through routing entries B and G.

The heavy volume on this route does not interfere with crucial message throughput on the S2 route, because a separate TCP connection carries each route.

**Defeating Independence**

The routing table entries within an rvrd process operate as independent pathways; that is, data does not flow directly between routing table entries within a routing daemon process instance.

Nonetheless, data can flow indirectly by way of a mutual neighbor. In Figure 19 on page 102, notice that adding a neighbor link between M and T would merge the route connecting networks A, B and C, with the otherwise disjoint route connecting X and Y (defeating their independence). Use caution when altering a network of routing daemons.
Figure 19  Mutual Neighbors Merge Routes

Legend

- Local Network
- Neighbor Link
- Routing Table Entry
- rvrd Process

Diagram:

- A
  - A.A.foo.com
  - B.X.foo.com
  - X

- B
  - S.B.foo.com
  - T.Y.foo.com
  - Y

- C
  - M.C.foo.com

Connections:

- A.A.foo.com to S.B.foo.com
- B.X.foo.com to T.Y.foo.com
- M.C.foo.com to A.A.foo.com
Common Topology Errors

This section describes two variants of an erroneous routing configuration.

Neighbors on the Same Network

It is an error to configure two neighbors to serve the same logical local network (network and service).

Since no gain could possibly result from forwarding messages from a network to the same network, it might seem that this error is rather rare. Nonetheless, in actual practice this error occurs rather frequently as an oversight.

Consider the situation in Figure 20 on page 104. In the desired outcome, neighbors on computers gemini and taurus exchange messages on UDP or PGM service 7500 between the two networks, Castor.star.com and Pollux.star.com. Because computer gemini has two network interfaces, the administrator attempts to limit rvrd operation to only Castor.star.com. Nonetheless, the routing daemon on gemini still receives messages from Pollux.star.com through its other interface (to understand the reason for this behavior, see Limitation on Computers with Multiple Network Interfaces on page 25). Because the two neighbors both serve the same network, Pollux.star.com, erroneous behavior results.

If gemini had only one network interface, Castor.star.com, the routing daemons would operate correctly.

When the routing daemon detects a topology error of this kind, it outputs an error message. Administrators must correct this situation immediately.
It is an error to use routing daemons to duplicate the effort of another forwarding mechanism (for example, a hardware router, or another pair of routing daemon neighbors. (This error is actually a variation of the error described in Neighbors on the Same Network on page 103.)

Consider the situation in Figure 21 on page 105. Two mechanisms forward messages between the two networks—the hardware router and a pair of routing daemons (A.a.bad.com and B.b.bad.com). When a program on network a.bad.com sends a message, routing daemon A forwards it to its neighbor B, which redistributes it on network b.bad.com. When the hardware router receives the redistributed message, it forwards it back to network a.bad.com, where A detects the duplication and produces an error message.

This kind of error can occur in either broadcast or multicast situations. However, it is especially common in environments where hardware routers enable multicast routing. Upgrading a hardware router can trigger this error.

Upgrading rvrd from release 5 to release 6 (or later) provides another fertile environment for this error. When both routing daemons run concurrently in the same network, be careful to avoid duplicate service.
To repair the situation, remove one of the routing daemons, or disable hardware multicast routing.

**Figure 21  Routing Daemons and Duplication**
Routing daemons offer security controls based on UDP or PGM service groups and subject names (see Restricting Message Flow on page 81). In addition, the routing daemon works in concert with firewalls to constrain information flow.

The WAN in Figure 22 connects two enterprises across the Internet. Each enterprise protects its networks with firewalls. Notice that the routing daemon within the DMZ does not serve any local network; instead that routing daemon operates as a *way station*, forwarding messages across the firewalls on either side of it.

*Figure 22  Routing Daemon WAN with Firewalls*
Neighbors Across a Firewall

Firewalls restrict the flow of information across organizational boundaries. For Rendezvous messages to flow between routing daemons, the daemons must establish TCP connections between neighbors. Security administrators can permit this connection using any technique they prefer; for example:

- Configure the firewall to permit SSL connections on the routing daemon’s local port. Configure the routing daemons to connect with one another using SSL neighbor connections.
- Configure VPN connectivity between neighbor host computers.
- Configure the firewall to permit TCP connections on the routing daemon’s local port.
- Configure the neighbors to connect using an SSH tunnel through the firewall.
Connecting PGM and TRDP Networks with Routing Daemons

PGM and TRDP network protocols do not interoperate. Nor can Rendezvous components and programs from one variant interoperate with components and programs from the other variant. The only exception to this rule is the Rendezvous routing daemon.

You can deploy a pair of Rendezvous routing daemons to construct a bridge that connects a PGM network with a TRDP network. This bridge lets PGM programs in one network communicate with TRDP programs in the other network.

Figure 23 depicts an example. A routing daemon from the TRDP variant runs in the TRDP network, while a routing daemon from the PGM variant runs in the PGM network. The two routers specify each another as neighbors. They forward both multicast and point-to-point messages.

Figure 23  Bridge PGM and TRDP Networks with rvrd

The two networks need not be physically distinct. For example, you can run PGM and TRDP variants on the same physical LAN—as long as they use non-overlapping services (that is, port numbers).
Retransmission

Within its local networks, a routing daemon is the source (that is, the sending daemon) of all the forwarded messages that it rebroadcasts. That routing daemon handles retransmission requests (for example, if a listening application in the local network misses a packet).

Figure 24 illustrates this concept:

1. An application in network A sends a message.
2. Routing daemon A forwards the message to routing daemon B.
3. Routing daemon B rebroadcasts the message on network B.
4. A receiving application in network B misses a packet, and its rvd requests retransmission.
5. If the packet is within the reliability window of routing daemon B, then it retransmits the packet.
   Otherwise, it denies the retransmission request; it does not attempt to get a new copy of the message from routing daemon A.

Figure 24  Retransmission and rvd
Border Routing

Border routing introduces a second tier of organization to rvrd routing networks, connecting several networks into a larger grouping while preserving their independence.

To configure border routing, see Border Routing on page 149 and Border Policy on page 155.

Border routing is an advanced feature. We recommend that you consult with TIBCO Software Inc. before deploying this feature.

Advantages

Border routing can be advantageous in some situations:

- In networks with many routers, border routers can limit the spread of routing-related information, resulting in increased network stability.
- When routing messages between separate enterprises—for example, to a business partner outside your intranet—border routers can isolate intranet topology information.
- Border router policies enforce subject gating restrictions pairwise between routing table entries (that is, neighbors and local networks). You can specify a separate policy for each ordered pair.

Concepts

First-Tier Router

In the context of border routing, the term first-tier router denotes the kind of router that is already familiar from the preceding sections of this chapter.

First-tier routers share a global routing table. Every first-tier router has its own copy of the entire routing table, spanning the entire routing network; any change in the routing network propagates to every router. In networks with many routers, the resulting overhead can be noticeable.

Border Router

A border router or second-tier router is an rvrd process that can serve as a border, dividing a routing network into separate zones (see Zone, below).

You can configure a border router with neighbors and local networks (in the same way as you would configure a first-tier router). The border router connects these elements, and forwards messages among them.
Policy

A policy defines the set of subjects that a border router forwards from one of its neighbors or local networks (called the From interface) to another of its neighbors or local networks (the To interface). To configure policy, see Border Policy on page 155.

A border router can restrict a subject, forwarding only those messages that have not yet crossed a border; see First Border on page 156.

Adding an interface to a border router automatically creates a default policy for all pairings of existing interfaces with the new interface. The default policy allows forwarding of _INBOX.> (that is, all point-to-point messages) in both directions (that is, from the new interface to each existing interface, and to the new interface from each existing interface). This default behavior mimics the behavior of first-tier routers (namely, inbox messages flow automatically, without explicit configuration).

This behavior is automatic when you add a border router interface using either the browser administration interface, or these Java configuration API methods:

- `Router.addActiveInterface()`
- `Router.addLocalNetworkInterface()`
- `Router.addPassiveInterface()`

Nonetheless, you may explicitly remove this subject (_INBOX.>) from the border policy to disable forwarding of inbox messages.

In contrast, specifying a new interface by editing an XML configuration overrides this default border policy. An XML document engenders a router configuration that matches the XML specification exactly; an interface will not have any border policy unless the XML document explicitly specifies one.

Zone

A zone or first-tier routing network is a collection of routers and local networks, in which every pair in the collection is connected by a route that does not cross through a border router.

Administrators do not explicitly configure zones. Instead, border routers periodically examine the network, and dynamically partition it into zones based on network connectivity.

In Figure 25, border router BR1 has two first-tier neighbors (K and L), and a route connects those neighbors without crossing through BR1 nor any other border router; so K and L are in the same zone.
The **effective policy** of a zone is the union of the policies of all its constituents. In other words, a message that can enter or leave through any constituent can enter or leave the zone as a whole. For example, if BR1 allows `foo.*` to cross from K to BR2, then `foo.*` can cross from anywhere in zone 2 to anywhere in zone 3.

Each border router process embodies several *implicit internal first-tier routers*. When a border router automatically groups its neighbors and local networks into zones, it tacitly instantiates one first-tier router (within itself) for each zone that it serves.

When we say that a border router participates in a zone, we really mean that one of the implicit first-tier routers within the border router participates in that zone. **Figure 26** expands a portion of **Figure 25**; it illustrates that border router BR1 contains three implicit internal first-tier routers, which serve zones 1, 2 and 3. Each one participates in one zone, as a representative of BR1.
Implicit first-tier routers are similar to the embodied routers described in Independent Routing Table Entries in One Process on page 98, except that you cannot configure them—the border router creates and configures them automatically. (Border routers can embody only implicit first-tier routers; they cannot configure explicit internal first-tier routers.)

Internal representatives of a border router are invisible to the external first-tier interfaces that they serve. All representatives of a border router present the same routing name, which is identical to the name of the border router (in our example, all three are named BR1). As a result, all external interfaces appear to communicate with BR1.

Within a border router, a border separates every pair of implicit internal first-tier routers (see Figure 26 on page 113). Border routers dynamically determine borders, just as they dynamically determine zones.

A message can cross a border when a policy allows the message subject.

First-tier routing table information cannot cross a border.

Borders are not directly accessible to administrators; they remain internal to border routers.
A second-tier routing network is a collection of border routers in which every pair in the collection is connected by a route.

First-tier routing information includes information about first-tier routers, local networks, and all the subjects that can flow among them (within a zone).

Second-tier-routing information includes information about border routers, zones, and all the subjects that can flow among them; it specifically excludes all first-tier routing information.

All border routers in a second-tier network share second-tier routing information, but not first-tier information. Conversely, first-tier constituents of zones cannot access second-tier information—except for information about subjects available through a participating border router.

For example, Figure 27 on page 114 illustrates the view of the second-tier network shared by BR1 and BR2 (based on the example of Figure 25). BR1 and BR2 share second-tier information so that both can create an internal routing table that includes both border routers, all five zones, and all the subjects that can flow among them. However, BR1 cannot access first-tier information about the constituents of zones 4 and 5, and BR2 cannot access first-tier information about the constituents of zones 1 and 2.

**Figure 27  Border Router: Second-Tier Routing Network**

---

**High-Fanout Second-Tier Networks**

You can use border routers to construct a high-fanout network, like the standard reference architecture in Figure 28 on page 115.
Within an enterprise, this architecture can promote network stability, use network bandwidth efficiently, and effectively control the flow of data.

When this architecture spans several enterprises, distribution-level border routers can isolate each enterprise network within a separate zone. With appropriate policy configuration, this architecture addresses privacy issues among partners in business-to-business applications.

**Figure 28  Border Router: High-Fanout Network**

### Best Practice: Zone Stability in Second-Tier Networks

In some topologies, a communication link failure can remove an entire zone from a second-tier network. For example, in the straightforward topology (left in **Figure 29**), a WAN failure disconnects the entire Remote Zone from border router BR. The border router automatically assesses the situation and rebuilds its zone map, however, this process can disrupt message flow for several minutes. A similar disruption can occur when the WAN resumes normal operation.
Contrast the better topology (right in Figure 29). BR and the first-tier router R1 both reside on the same side of the WAN link (possibly even on the same host computer). In this topology, WAN failure does not disconnect the entire Remote Zone, since BR remains in contact with R1. BR need not reconfigure its zone map, so it avoids associated delays.

**Figure 29  Border Router: Zone Stability**

![Legend](image)

**Legend**
- Local Network
- Neighbor Link
- Routing Table Entry (First-Tier Router)
- Border Router (Second-Tier Router)

**Best Practice: Fault Tolerance in Second-Tier Networks**

Figure 30 illustrates fault tolerance in a second-tier network that crosses a WAN link. Notice that this topology addresses two aspects of fault-tolerance:

- Redundant border routers—BR1 and BR2
- Redundant routing across WAN communications—the X pattern spanning the first-tier routers L1, L2, M1 and M2 within zone B
Best Practice: Isolating Enterprise Zones in Second-Tier Networks

Business-to-business (B2B) networks often require strict isolation of each partner’s data at the same time as they require data to flow in two directions between the hub and each partner. Figure 31 on page 118 illustrates a situation in which the main network in zone A must exchange data freely with partner P and partner Q—but data from either partner must not flow to the other partner. The border policy configures this separation as required.
(Incidentally, notice that Figure 31 features zone stability, co-locating routers P1 and Q1 near border router BR; see Best Practice: Zone Stability in Second-Tier Networks on page 115.)

Figure 31   Border Router: Isolating Enterprise Zones

Fault Tolerance

Adding fault tolerance to the topology of Figure 31 would seem straightforward, but actually adds an unexpected complication. Figure 32 on page 119 depicts the resulting topology. Notice that we address two aspects of fault tolerance (as we did in Figure 30):

- Redundant border routers (BR1 and BR2) guard against failure of either member of this pair. BR1 and BR2 each connect the hub to both partner zones (zone P and zone Q).
- Redundant routing across WAN communications guards against WAN link failure. The familiar X pattern is repeated within each partner zone.

Yet this topology introduces a prohibited behavior—messages can flow between the two partners. BR1 routes messages from partner P to hub A; BR2 forwards the messages from hub A to partner Q. To block this unintended data flow, administrators must properly configure border policy on BR1 and BR2; it is crucial to restrict the flow to those messages which have not yet crossed a border (see First Border on page 156). Configure this restriction separately for each subject that these border routers forward.
Figure 32  Border Router: Isolating Enterprise Zones and Fault Tolerance
Backlog Protection

Every WAN connection has a maximum capacity. Routing daemons cannot exceed this physical limitation. When the volume of routed data is greater than WAN capacity, rvrld buffers the outbound data.

Data backlog can occur in several scenarios; for example:

- An unexpected burst of data exceeds WAN capacity.
- A temporary problem with the WAN sharply decreases its capacity.
- WAN capacity is insufficient for the required volume of data.
- WAN capacity is generally sufficient, but rvrld is misconfigured to route more data than expected. The total data volume exceeds WAN capacity.

The Connected Neighbors page displays the peak backlog for each neighbor; see Connected Neighbors on page 140.

Maximum Backlog

An extremely large backlog can cause severe problems for rvrld and its host computer. Administrators can configure rvrld to protect against this possibility.

When enabling this feature, the administrator specifies the maximum permissible backlog (in kilobytes). When an outbound backlog of this size accumulates for any neighbor connection, rvrld automatically disconnects from that neighbor, clears the corresponding outbound data buffer, and attempts to reconnect to the neighbor.

To obtain a reasonable estimate for the threshold value that triggers this action, calculate the process storage available to rvrld, divided by the number of neighbor connections it serves.

You can configure this feature separately for each routing table entry. The router applies that maximum to all of its neighbor connections.

To configure this feature, see Routers on page 148.

Notice that enabling this feature represents a deliberate decision to discard data in certain extreme circumstances. When this feature is disabled (the default), the routing daemon does not protect against backlog. The decision to use this feature must be based on the business requirements of the enterprise.
rvrd can run in either of two states—running or idle.

When *running*, rvrd establishes neighbor connections and routes messages.

When *idle*, rvrd does no routing operations. However, the browser administration interface is available for configuring parameters. The process still behaves as a Rendezvous daemon (rvd).

While rvrd is in idle state, you can configure the routing table and other parameters without affecting the network in any way, without binding local resources (such as UDP or PGM services or TCP ports), and without resolving any names in the routing table. After saving the configuration in a store file (and terminating the rvrd process), you can restart rvrd using the stored configuration. Alternatively, you can move the store file to another host computer, and start rvrd there.
Routing Daemon Logging

A routing daemon process can output a running log of its activity. System administrators can use the resulting log files to monitor neighbor connections, subject interest and message flow.

To configure the kinds of normal activity to log, see Logging on page 147.
To configure the destination of log output, see Log Destination on page 54.

The command line parameter -log is obsolete in release 6 (and later). Use the browser administration interface to configure rvrd logging categories.

Interpreting Log Output

Each line in the log file describes a significant event in the operation of a routing daemon. A time stamp indicates the date and time of the interaction. The remainder of the line is a string describing the event.

The log file begins with events in the routing daemon’s start sequence. First, it discovers its hardware and software operating environment:

```
2004-09-08 14:03:13 rvrd: Hostname: optimist
2004-09-08 14:03:13 rvrd: Hostname IP address: 10.101.2.140
2004-09-08 14:03:13 rvrd: Detected IP interface: 127.0.0.1 (lo)
2004-09-08 14:03:13 rvrd: Detected IP interface: 10.101.2.140 (eth0)
2004-09-08 14:03:13 rvrd: Using ticket file /usr/local/tibco/bin/tibrv.tkt
2004-09-08 14:03:13 rvrd: Using store file /tmp/1.admin
2004-09-08 14:03:13 rvrd: Warning: zlib compression not supported in SSL initialization.
```

Next, the routing daemon reads its configuration from its store file. In this example, it defines a router (routing table entry) named optimist. That router has an accept any neighbor interface, and a local network interface.

```
2004-09-08 14:03:13 rvrd: [optimist]: Defined.
2004-09-08 14:03:13 rvrd: [optimist]: Any neighbor is allowed to connect to local port 9666. Link cost: 1.
```
2004-09-08 14:03:13 rvrd: [optimist]: Local network 7505.RV.TIBCO defined.
Interface: 10.101.2.140. Service UDP port: 7505, Service spec: 7505, Network spec:
10.101.2. Link cost: 1.

The routing daemon finishes its start sequence by reporting the URL bindings of
its browser administration interfaces.


The administrator sets the logging parameters for normal activity.

2004-09-08 14:08:35 rvrd: Logging: [Connections - On], [Subject Interest - On],
[Subject Data - On].

Now the routing daemon begins normal operations. Log items reflect neighbor
connections to other routers (viggen-r1), exchange of subscription interest
information, and forwarding of message data.

2004-09-08 14:13:26 rvrd: [optimist]: Connected to viggen-r1.

2004-09-08 14:15:40 rvrd: [optimist]: Sending subscription for [TEST] to viggen-r1
for source 0A65023F/hpux11-viggen-n1.
2004-09-08 14:16:30 rvrd: [optimist]: Received data on [TEST] from neighbor
viggen-r1.

2004-09-08 14:18:03 rvrd: [optimist]: Sending cancel for [TEST] to viggen-r1 for
source 0A65023F/hpux11-viggen-n1.


Then optimist discovers another routing daemon (fanox) serving the same local
network and service (redundancy for fault tolerance). Then it loses contact with
fanox.

2004-09-08 14:23:16 rvrd: [optimist]: Found fanox in 7505.RV.TIBCO.
2004-09-08 14:24:24 rvrd: [optimist]: Lost fanox in 7505.RV.TIBCO.
**rvrd**

**Command**

**Syntax**

rvrd  
- store _filename_
[-http [ip_address:]http_port]  
[-https [ip_address:]https_port]  
[-http-only]  
[-https-only]  
[-no-http]  
[-idle]  
[-listen [socket_protocol:]tcp_port]  
[-no-multicast]  
[-reliability _time_]  
[-max-consumer-buffer _size_]  
[-rxc-max-loss _loss_]  
[-rxc-recv-threshold _bps_]  
[-rxc-send-threshold _bps_]  
[-compress-level _level_]  
[-rvdm _control_channel_]  
[-rvdm-reverse-asym]  
[-reuse-port _inbox_port]  
[-logfile _log_filename]  
[-log-max-size _size_]  
[-log-max-rotations _n_]  
[-log-config _log_config_filename]  
[-foreground]

**Purpose**

The routing daemon efficiently connects Rendezvous programs on distant IP networks, so that messages flow between them as if within a single network. Nonetheless, communicating programs remain decoupled from internetwork addresses and other details.

**Remarks**

The **rvrd** process subsumes the behavior of **rvd**, so it is not necessary to run a separate **rvd** process on computers that run **rvrd**. We recommend against running both components on the same computer.

**rvrd** must run on a host computer with a **permanent** IP address. For example, a temporary address assigned by DHCP is invalid.

**Command Line Parameters**
-store filename

This file contains the routing table entry and parameters that configure `rvrd`. `rvrd` reads this file when the process starts, and writes this file each time you change the configuration using the browser administration interface.

⚠️

The store file requires physical security safeguards and operating system protection. Keep it in a location that is accessible only to the system administrators who maintain it.

See also Appendix B, Store Files, on page 419.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-http ip_address: http_port</td>
<td>The browser administration interface accepts connections on this HTTP or HTTPS port. Permit administration access only through the network interface specified by this IP address.</td>
</tr>
<tr>
<td>-http http_port</td>
<td>To limit access to a browser on the <code>rvrd</code> host computer, specify 127.0.0.1 (the local host address).</td>
</tr>
<tr>
<td>-https ip_address: https_port</td>
<td>When the IP address is absent, the daemon accepts connections through any network interface on the specified HTTP or HTTPS port.</td>
</tr>
<tr>
<td>-https https_port</td>
<td>If the explicitly specified HTTP port is already occupied, the program exits.</td>
</tr>
<tr>
<td></td>
<td>If the explicitly specified HTTPS port is already occupied, the program selects an ephemeral port.</td>
</tr>
<tr>
<td></td>
<td>When the <code>-http</code> parameter is entirely absent, the default behavior is to accept connections from any computer on HTTP port 7580; If this default port is unavailable, the operating system assigns an ephemeral port number.</td>
</tr>
<tr>
<td></td>
<td>When the <code>-https</code> parameter is entirely absent, the default behavior is to accept secure connections from any computer on an ephemeral HTTPS port.</td>
</tr>
<tr>
<td></td>
<td>In all cases, the program prints (in its start banner and log file) the actual HTTP and HTTPS ports where it accepts browser administration interface connections.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>-http-only</td>
<td>Disable HTTPS (secure) connections, leaving only an HTTP (non-secure) connection.</td>
</tr>
<tr>
<td>-https-only</td>
<td>Disable HTTP (non-secure) connections, leaving only an HTTPS (secure) connection.</td>
</tr>
<tr>
<td>-no-http</td>
<td>Disable all HTTP and HTTPS connections, overriding -http and -https.</td>
</tr>
<tr>
<td>-idle</td>
<td>When present, start rvrd in its idle state. When absent, start rvrd in its running state—routing messages.</td>
</tr>
<tr>
<td>-listen tcp_port</td>
<td>rvd (and by extension, rvrd operating within the local network) opens a TCP client socket to establish communication between itself and its client programs. The -listen parameter specifies the TCP port where the Rendezvous daemon listens for connection requests from client programs. This -listen parameter of rvd corresponds to the daemon parameter of the transport creation call (they must specify the same TCP port number). The IP address specifies the network interface through which this daemon accepts TCP connections. To bar connections from remote programs, specify IP address 127.0.0.1 (the loopback interface). When the IP address is absent, the daemon accepts connections from any computer on the specified TCP port. When this parameter is entirely absent, the default behavior is to accept connections from any computer on TCP port 7500. For more detail about the choreography that establishes conduits, see Daemon Client Socket—Establishing Connections on page 28.</td>
</tr>
<tr>
<td>-listen ip_address:tcp_port</td>
<td></td>
</tr>
<tr>
<td>-listen socket_protocol:tcp_port</td>
<td></td>
</tr>
</tbody>
</table>

This parameter does not correspond to the service parameter of the transport creation call—but rather to the daemon parameter.
Sending to subjects with lead wildcards (for example, > or *.foo) can cause unexpected behavior in some applications, and cause network instability in some configurations. This option lets you selectively screen wildcard sending.

When `-no-lead-wc` is present, the daemon quietly rejects client requests to send outbound messages to subjects that contain wildcards in the lead element. The daemon does not report excluded messages as errors.

When `-lead-wc` is present (or when neither flag is present), the daemon allows sending messages to subjects with lead wildcards.

When present, the daemon disables multicast (and broadcast) communication. For details, see Disabling Multicast on page 38.

Rendezvous daemons compensate for brief network failures by retaining outbound messages, and retransmitting them upon request.

This parameter is one of several ways to control the message reliability interval. For a complete discussion the concept of reliability, the various ways to control it, the interaction among those ways, and reasonable values, see Reliability and Message Retention Time on page 34.

If this parameter is absent, `rvd` uses the factory default (60 seconds).

If this parameter is present, `rvd` (and by extension, `rvrd` operating within the local network) retains messages for `time` (in seconds). The value must be a non-negative integer.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-no-lead-wc</code></td>
<td>Sending to subjects with lead wildcards (for example, &gt; or *.foo) can cause unexpected behavior in some applications, and cause network instability in some configurations. This option lets you selectively screen wildcard sending.</td>
</tr>
<tr>
<td><code>-lead-wc</code></td>
<td>When <code>-no-lead-wc</code> is present, the daemon quietly rejects client requests to send outbound messages to subjects that contain wildcards in the lead element. The daemon does not report excluded messages as errors. When <code>-lead-wc</code> is present (or when neither flag is present), the daemon allows sending messages to subjects with lead wildcards.</td>
</tr>
<tr>
<td><code>-no-multicast</code></td>
<td>When present, the daemon disables multicast (and broadcast) communication. For details, see Disabling Multicast on page 38.</td>
</tr>
<tr>
<td><code>-reliability time</code></td>
<td>Rendezvous daemons compensate for brief network failures by retaining outbound messages, and retransmitting them upon request. This parameter is one of several ways to control the message reliability interval. For a complete discussion the concept of reliability, the various ways to control it, the interaction among those ways, and reasonable values, see Reliability and Message Retention Time on page 34. If this parameter is absent, <code>rvd</code> uses the factory default (60 seconds). If this parameter is present, <code>rvd</code> (and by extension, <code>rvrd</code> operating within the local network) retains messages for <code>time</code> (in seconds). The value must be a non-negative integer.</td>
</tr>
</tbody>
</table>
When present, the daemon enforces this upper bound (in bytes) on each consumer buffer (the queue of messages for a client transport). When data arrives faster than the client consumes it, the buffer overflows this size limit, and the daemon discards the oldest messages to make space for new messages. The client transport receives a \texttt{CLIENT.SLOWCONSUMER} advisory.

When absent or zero, the daemon does not enforce a size limit on the consumer buffer. (However, a 60-second time limit on messages still limits buffer growth, independently of this parameter.)

These three parameters configure the retransmission control (RXC) feature, which suppresses retransmission requests from chronically-lossy receivers. (This feature applies to the \texttt{rvd} behavior within \texttt{rvrd}, but not to routing behavior.)

If \texttt{-rxc-max-loss} is absent or zero, then RXC is disabled. If it is an integer in the range \([1, 100]\), it determines the maximum percentage acceptable loss rates above which a receiver is considered chronically-lossy.

\texttt{-rxc-recv-threshold} configures the threshold receive rate (in bits per second) above which a chronically-lossy receiver censors its own retransmission requests. When absent, the default value is zero (always censor a chronically-lossy receiver).

\texttt{-rxc-send-threshold} configures the threshold send rate (in bits per second) above which the daemon suppresses (that is, ignores requests from) chronically-lossy receivers. When absent, the default value is zero (always suppress retransmissions to chronically-lossy receivers).

For a complete explanation, see Retransmission Control on page 50.
When present, this option guides the trade-off between data compression and data latency. Acceptable values are integers in the range $[1, 10]$.

- 1 favors minimum latency, sacrificing compression efficiency.
- 10 favors maximal compression, accepting the concomitant cost of latency.

This option applies across all neighbor interfaces (it is not possible to specify different values for each neighbor). Furthermore, it applies only to neighbor interfaces that are configured for data compression without SSL.

When absent, the default behavior is equivalent to 10—favoring compression over latency.

When present, the daemon operates as a managed daemon. (For a complete explanation of this feature, see Daemon Manager on page 209.)

The $control\_channel$ argument specifies the dedicated control channel over which the RVDM server coordinates with managed Rendezvous daemons. This value must denote the same control channel as configured in the RVDM server; see Control Channel on page 257. The form of the $control\_channel$ argument is:

$$port\_number : [ network\_interface ] ; multicast\_address$$

- $port\_number$ (required) specifies a service port.
- $network\_interface$ (optional) specifies the network interface (for host computers with multiple interfaces). When this element is absent, the default value is the host’s primary interface.
- $multicast\_address$ (required) is a dedicated multicast address for RVDM control messages.

When this entire parameter is absent, the daemon operates as a non-managed daemon with respect to RVDM.
When present, this daemon reverses the direction of the RVDM multicast groups. If RVDM instructs it to listen on group 225.1.1.1 and send on 225.2.2.2, it reverses direction and instead listens on 225.2.2.2 and sends on 225.1.1.1. For a usage scenario, see Asymmetric Multicast on page 218.

When absent, this daemon obeys multicast group designations from RVDM without alteration.

When present, other daemons on the same host computer can reuse service ports.

When absent, other daemons cannot reuse a service port that is in use by this daemon.

For correct operation, all the daemons that use a common service port on a host computer must specify this option. For background and details, see Reusing Service Ports on page 52.

The `inbox_port` argument (required) specifies the UDP port that this daemon uses for point-to-point communications. This value must be unique for each daemon (that reuses service ports) on a common host computer.

Furthermore, you must not use the `inbox_port` in any transport specification on the same host computer. When using RVDM, use care to avoid violating this restriction inadvertently. You must not use the `inbox_port` for the RVDM control channel. You must not define a subject map for the `inbox_port`. You must not specify the `inbox_port` as an effective port in the RVDM port map.

Send log output to this file.

When absent, the default is `stderr`.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-rvdm-reverse-asym</td>
<td>When present, this daemon reverses the direction of the RVDM multicast groups. If RVDM instructs it to listen on group 225.1.1.1 and send on 225.2.2.2, it reverses direction and instead listens on 225.2.2.2 and sends on 225.1.1.1. For a usage scenario, see Asymmetric Multicast on page 218. When absent, this daemon obeys multicast group designations from RVDM without alteration.</td>
</tr>
<tr>
<td>-reuse-port inbox_port</td>
<td>When present, other daemons on the same host computer can reuse service ports. When absent, other daemons cannot reuse a service port that is in use by this daemon. For correct operation, all the daemons that use a common service port on a host computer must specify this option. For background and details, see Reusing Service Ports on page 52. The <code>inbox_port</code> argument (required) specifies the UDP port that this daemon uses for point-to-point communications. This value must be unique for each daemon (that reuses service ports) on a common host computer. Furthermore, you must not use the <code>inbox_port</code> in any transport specification on the same host computer. When using RVDM, use care to avoid violating this restriction inadvertently. You must not use the <code>inbox_port</code> for the RVDM control channel. You must not define a subject map for the <code>inbox_port</code>. You must not specify the <code>inbox_port</code> as an effective port in the RVDM port map.</td>
</tr>
<tr>
<td>-logfile log_filename</td>
<td>Send log output to this file. When absent, the default is <code>stderr</code>.</td>
</tr>
</tbody>
</table>
When present, activate the log rotation regimen (see Log Rotation on page 54).

When you specify these options, you must also specify -logfile.

*size* is in kilobytes. If *size* is non-zero, it must be in the range \([100, 2097152]\). Values outside this range are automatically adjusted to the nearest acceptable value. Zero is a special value, which disables log rotation. When -log-max-size is zero or absent, a single log file may grow without limit (other than the limit of available storage).

*n* indicates the maximum number of files in the rotation. When -log-max-rotations is absent, the default value is 10.

Send duplicate log output to this file for log items that record configuration changes. The daemon never rotates nor removes this special log file. Instead, this file remains as a record of all configuration changes.

When absent, the default is *stderr*.

Available only on UNIX platforms.

When present, *rvrd* runs as a foreground process.

When absent, *rvrd* runs as a background process.
Browser Administration Interface—rvrd

The browser administration interface lets you control rvrd from a web browser. You can configure its operating parameters, and view operating statistics.

Topics

- Navigation, page 133
- General Information, page 136
- Local Networks, page 138
- Connected Neighbors, page 140
- Daemon Parameters, page 145
- Routers, page 148
- Local Network Interfaces Configuration, page 151
- Subject Gating, page 153
- Neighbor Interfaces, page 157
- Certificates, page 163
Navigation

All browser administration interface pages display a navigation panel at the left side of the page. Use these links to display other pages.

Figure 33  rvrd Navigation Panel

State:
- General Information
- Clients
- Local Networks
- Connected Neighbors
- Services

Daemon Manager:
- Subject Map
- Port Map

Configuration:
- Daemon Parameters
- Routers
- XML Configuration
- Certificates

Miscellaneous:
- Current Log
- Copyright
- TIBCO Rendezvous Web Page
<table>
<thead>
<tr>
<th>Category</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>General Information</td>
<td>This page displays information about an rvrd process; see General Information on page 136.</td>
</tr>
<tr>
<td></td>
<td>Clients</td>
<td>This page summarizes the client transports; see Clients on page 61.</td>
</tr>
<tr>
<td></td>
<td>Local Networks</td>
<td>This page summarizes the local networks of a router; see Local Networks on page 138.</td>
</tr>
<tr>
<td></td>
<td>Connected Neighbors</td>
<td>This page summarizes the actual neighbor connections of a router; see Connected Neighbors on page 140.</td>
</tr>
<tr>
<td></td>
<td>Services</td>
<td>This page summarizes network services activity; see Services on page 64.</td>
</tr>
<tr>
<td>Daemon Manager</td>
<td>Subject Map</td>
<td>This page summarizes RVDM subject maps by service number.; see Subject Map Summary on page 71.</td>
</tr>
<tr>
<td></td>
<td>Port Map</td>
<td>This page displays the RVDM port map; see Port Map on page 74.</td>
</tr>
<tr>
<td>Configuration</td>
<td>Daemon Parameters</td>
<td>This page lets you configure parameters that control configuration access and router logging; see Daemon Parameters on page 145.</td>
</tr>
<tr>
<td></td>
<td>Routers</td>
<td>This page lets you configure routers. You can access additional configuration pages through links on this page. See Routers on page 148, and the sections that follow it.</td>
</tr>
<tr>
<td></td>
<td>XML Configuration</td>
<td>This page lets you view the current configuration as an XML document, and reconfigure the component by submitting an edited XML document.</td>
</tr>
<tr>
<td></td>
<td>Certificates</td>
<td>This page lets you configure certificates that the daemon uses to identify itself in secure protocols. See Certificates on page 163.</td>
</tr>
<tr>
<td></td>
<td>Log Out</td>
<td>This item logs out the current user or Administrator. See Log Out on page 147.</td>
</tr>
<tr>
<td>Category</td>
<td>Item</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Current Log</td>
<td>This page displays the most recent 4 kilobytes from the log file.</td>
</tr>
<tr>
<td></td>
<td>Copyright</td>
<td>The Rendezvous copyright page.</td>
</tr>
<tr>
<td></td>
<td>TIBCO Rendezvous Web Page</td>
<td>The product page from the TIBCO web site.</td>
</tr>
</tbody>
</table>
General Information

rvrd (like all Rendezvous components) displays information about itself on this page.

To display this page, click General Information in the left margin of any page of the rvrd browser administration interface.

Figure 34  rvrd General Information Page

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component</td>
<td>The name of the program—rvrd (or rvsrd).</td>
</tr>
<tr>
<td>version</td>
<td>Version number of the program.</td>
</tr>
<tr>
<td>license ticket</td>
<td>The license ticket that validates this process.</td>
</tr>
<tr>
<td>host name</td>
<td>The hostname of the computer where the daemon process runs. Notice that the daemon process can run on one computer, while you access its browser interface from another computer.</td>
</tr>
<tr>
<td>user name</td>
<td>The user who started the daemon process.</td>
</tr>
</tbody>
</table>
### Item Description

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP address</td>
<td>The IP address of the computer where the daemon process runs.</td>
</tr>
<tr>
<td>client port</td>
<td>The TCP port where the daemon listens for client connections.</td>
</tr>
<tr>
<td>network services</td>
<td>The number of network services on which this daemon’s clients communicate.</td>
</tr>
<tr>
<td>routing names</td>
<td>The number of router names that this daemon embodies; see <a href="#">Routing Table Entry</a> on page 83.</td>
</tr>
<tr>
<td>store file</td>
<td>File name of the daemon’s store file; see the command line parameter <code>-store</code> for <code>rvsd</code> on page 180, and for <code>rvsrd</code> on page 184.</td>
</tr>
<tr>
<td>process ID</td>
<td>The operating system’s process ID number for the component.</td>
</tr>
<tr>
<td>managed</td>
<td>A link indicates that this daemon is managed (see <a href="#">Chapter 7, Daemon Manager, on page 209</a>). The link is the name of an RVDM instance; to view or interact with the instance, click this link. Otherwise, this daemon is a non-managed daemon.</td>
</tr>
<tr>
<td>control channel</td>
<td>The RVDM control channel specification of a managed daemon; see <code>-rvdm</code> on page 47. For non-managed daemons, this field displays the string <code>disabled</code>.</td>
</tr>
</tbody>
</table>
## Local Networks

rvrd displays information about its local networks on this page.

To display this page, click **Local Networks** in the left margin of any page of the rvrd browser administration interface.

*Figure 35  rvrd Local Networks Page*

<table>
<thead>
<tr>
<th>Router Name</th>
<th>Local Network Name</th>
<th>Service</th>
<th>Network Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>router.1</td>
<td>lan.1</td>
<td>7502</td>
<td>;224.1.1.12</td>
</tr>
</tbody>
</table>

**Local Routers**

<table>
<thead>
<tr>
<th>Hostname</th>
<th>IP Address</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Subscriptions**

0

(Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Router Name</strong></td>
<td>This page groups local networks by router name (routing table entry). A box in this column indicates the name of the routing table entry that serves the local networks shown to its right. See also Routing Table Entry on page 83.</td>
</tr>
<tr>
<td><strong>Local Network Name</strong></td>
<td>The name of a local network.</td>
</tr>
<tr>
<td><strong>Service</strong></td>
<td>The UDP or PGM service for communication on the local network. On a managed daemon, if the RVDM port map shunts the service port, then this number is the effective port (see Port Map on page 223).</td>
</tr>
<tr>
<td><strong>Network Specification</strong></td>
<td>The network specification (as specified by the routing table entry).</td>
</tr>
<tr>
<td><strong>Local Routers</strong></td>
<td>This subtable lists other routers that serve this local network.</td>
</tr>
<tr>
<td>Item</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Hostname</td>
<td>The name of the host computer where the other routing daemon runs. Click here to view the browser administration interface for the other routing daemon process.</td>
</tr>
<tr>
<td>IP Address</td>
<td>The IP address of the host computer where the other routing daemon runs.</td>
</tr>
<tr>
<td>Version</td>
<td>The version of the other routing daemon executable.</td>
</tr>
</tbody>
</table>
| Subscriptions | Total number of subscriptions over all transports within the local network, for which clients have registered interest.  
Click this link to view a list of the subscription subjects; see Subscription List on page 63. |


Connected Neighbors

rvrd displays information about its (actual) neighbor connections on this page. To display this page, click Connected Neighbors in the left margin of any page of the rvrd browser administration interface.

This page is related to—but not the same as—the page described in Neighbor Interfaces on page 157.

Figure 36  rvrd Connected Neighbors Page

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table rows</td>
<td>Each row in this table describes one neighbor connection.</td>
</tr>
<tr>
<td>Router Name</td>
<td>This page groups neighbors by local router name (routing table entry). A box in this column indicates the name of the local router that connects to the neighbors show to its right. See also Routing Table Entry on page 83.</td>
</tr>
<tr>
<td>Neighbor Name</td>
<td>The name of a remote router with which the local router has a neighbor connection. Click here to view the browser administration interface for the neighbor routing daemon process.</td>
</tr>
<tr>
<td>Link Stats</td>
<td>The name of the (local) neighbor interface that specifies this neighbor connection. rvrd generated this name automatically when you configured the neighbor interface. Click this name to view the Router Connection Statistics page.</td>
</tr>
</tbody>
</table>
Router Connection Statistics

rvrd displays statistics about the performance of a neighbor connection on this page.

To display this page, click the Link Stats column of the Connected Neighbors page.

Connection statistics are not available when neighbors connect using SSL. See also SSL Connection with Compression on page 162.
Figure 37  rvrd Router Connection Statistics Page

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>summary</strong></td>
<td>This list presents static information about the neighbor connection.</td>
</tr>
<tr>
<td>Router Name</td>
<td>The name of the local router. (See also Routing Table Entry on page 83.)</td>
</tr>
<tr>
<td>Neighbor Name</td>
<td>The name of the neighbor (remote router).</td>
</tr>
<tr>
<td>Interface Number</td>
<td>The name of the (local) neighbor interface that specifies the neighbor connection. rvrd generated this number automatically when you configured the neighbor interface, and incorporates it into the neighbor ID.</td>
</tr>
<tr>
<td>Local Port</td>
<td>TCP port that this router uses to communicate with the neighbor.</td>
</tr>
</tbody>
</table>

**Router Connection Statistics**

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Router Name</td>
<td>shen03</td>
</tr>
<tr>
<td>Neighbor Name</td>
<td>shen01</td>
</tr>
<tr>
<td>Interface No.</td>
<td>2</td>
</tr>
<tr>
<td>Local Port</td>
<td>9475</td>
</tr>
<tr>
<td>Remote Port</td>
<td>9475</td>
</tr>
<tr>
<td>Cost</td>
<td>1</td>
</tr>
<tr>
<td>SSL Connection</td>
<td>NO</td>
</tr>
<tr>
<td>Data Compression</td>
<td>YES</td>
</tr>
<tr>
<td>Backlog Limit (bytes)</td>
<td>256K</td>
</tr>
</tbody>
</table>

**Data Flow**

<table>
<thead>
<tr>
<th>Data Flow</th>
<th>Messages</th>
<th>Bytes</th>
<th>Bytes/Sec</th>
<th>Compr Bytes</th>
<th>Compr Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound</td>
<td>93</td>
<td>21853</td>
<td>22</td>
<td>15800</td>
<td>0.723</td>
</tr>
<tr>
<td>Outbound</td>
<td>10540</td>
<td>277352511</td>
<td>89917</td>
<td>64466839</td>
<td>0.232</td>
</tr>
</tbody>
</table>

**Miscellaneous Statistics**

<table>
<thead>
<tr>
<th>Peak Backlog (bytes)</th>
<th>Curr Backlog (bytes)</th>
<th>Reconnects (times)</th>
<th>Total Inbound (bytes)</th>
<th>Total Outbound (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>52.623 K</td>
<td>0</td>
<td>0</td>
<td>21.341 K</td>
<td>264.504 M</td>
</tr>
</tbody>
</table>

(Sheet 1 of 3)
<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote Port</td>
<td>TCP port that the neighbor (remote router) uses to communicate with the local router.</td>
</tr>
<tr>
<td>Cost</td>
<td>Path cost of the neighbor link.</td>
</tr>
<tr>
<td>SSL Connection</td>
<td>SSL security feature for the neighbor link.</td>
</tr>
<tr>
<td>Data Compression</td>
<td>Data compression feature for the neighbor link. When a connection uses SSL with data compression, then compression statistics are not available (because SSL does the compression). For background information, see Data Compression on page 90, and SSL Connection with Compression on page 162.</td>
</tr>
<tr>
<td>Backlog Limit</td>
<td>When backlog protection is enabled, this item displays the threshold for disconnect from the neighbor. See Backlog Protection on page 120.</td>
</tr>
<tr>
<td><strong>Data Flow</strong></td>
<td>This table displays statistics about the volume of data on the neighbor connection.</td>
</tr>
<tr>
<td></td>
<td>The Inbound row displays statistics about inbound data from the remote neighbor to the local router.</td>
</tr>
<tr>
<td></td>
<td>The Outbound row displays statistics about outbound data from the local router to the remote neighbor.</td>
</tr>
<tr>
<td>Messages</td>
<td>Cumulative count of messages.</td>
</tr>
<tr>
<td>Bytes</td>
<td>Cumulative count of bytes (without compression).</td>
</tr>
<tr>
<td>Bytes/Sec</td>
<td>Data transmission rate during the most recent interval.</td>
</tr>
<tr>
<td>Compr Bytes</td>
<td>Cumulative count of compressed bytes.</td>
</tr>
<tr>
<td></td>
<td>This item displays non-zero only when both of the neighbors specify data compression on the rvrd Neighbor Interface Configuration Form.</td>
</tr>
<tr>
<td>Compr Ratio</td>
<td>Compression ratio.</td>
</tr>
<tr>
<td></td>
<td>This item displays non-zero only when both of the neighbors specify data compression on the rvrd Neighbor Interface Configuration Form.</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>This table displays statistics not related to either inbound or outbound data transmission.</td>
</tr>
<tr>
<td>Statistics</td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Peak Backlog</td>
<td>Peak backlog of outbound data (in bytes) since the last reset of statistics. See also, Backlog Protection on page 120.</td>
</tr>
<tr>
<td>Curr Backlog</td>
<td>Current backlog of outbound data (in bytes). See also, Backlog Protection on page 120.</td>
</tr>
<tr>
<td>Reconnects</td>
<td>Cumulative count of times when the neighbor link became disconnected and subsequently reconnected. (For example, network failure or backlog protection could cause a disconnect.)</td>
</tr>
<tr>
<td>Total Inbound</td>
<td>Cumulative counts of inbound and outbound bytes (without compression) since the start of the neighbor connection. The Reset Statistics button does not affect these items.</td>
</tr>
<tr>
<td>Total Outbound</td>
<td></td>
</tr>
<tr>
<td>Reset Statistics</td>
<td>Click this button to reset statistical counters to zero.</td>
</tr>
</tbody>
</table>
### Daemon Parameters

This page lets you configure parameters that affect overall daemon security.

To display this page, click **Daemon Parameters** in the left margin of any page of the *rvrd* browser administration interface.

*Figure 38  rvrd Daemon Parameters Configuration Page*

#### Administrator and Password

Only authorized personnel have access to routing daemons.

When administrator identification information is *not* set, anyone who can connect to the browser administration interface can examine and reconfigure the daemon. This arrangement can be useful during initial configuration and testing phases. However, during regular operation we recommend limiting access.
Once administrator identification information is registered, the browser administration interface is locked against unauthorized access. The daemon prompts administrators to prove identity by typing a name and password. After providing proper identification, an authorized administrator is logged in, and has complete access to configure the daemon. If the administrator does not provide proper identification, the browser displays the General Information page and continues to prompt for a correct name and password.

Browsers remember administrator name and password information for the duration of the browser process. Merely closing the browser window does not erase this information. To guard against intruders you must terminate the browser process (all its windows).

The first administrator to register is called the primary administrator. In addition to configuring the daemon, the primary administrator can also add, delete and modify identification information pertaining to the other administrators.

Each daemon configuration can store up to 16 additional administrator name and password pairs (after the primary administrator).

Each daemon process permits only one administrator session at a time. When one administrator is logged in, other administrators are locked out; this prevents conflicts in which two administrators attempt to modify the configuration at the same time. To terminate a administrator session, see Log Out (below).

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type a name string.</td>
</tr>
<tr>
<td>Password</td>
<td>Type a password string.</td>
</tr>
<tr>
<td>Confirm Password</td>
<td>Type the password again.</td>
</tr>
<tr>
<td>Add/Update</td>
<td>Specify a name and password, then click this button to add a new administrator. The primary administrator can add other administrators and update their passwords. All other administrators can update only their own passwords.</td>
</tr>
</tbody>
</table>
Log Out

To end an administrative session, click Log Out in the left margin of the browser administration interface. This item appears only when you are logged in as an Administrator.

Daemons automatically log out administrator sessions that have been idle for 10 minutes.

Logging

This panel configures the kind of routing activity that the routing daemon routinely outputs to its log file.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connections</td>
<td>Log connection activity whenever this routing daemon establishes or closes a connection to a neighbor.</td>
</tr>
<tr>
<td>Subject Interest</td>
<td>Log all subscription requests (notification of listening) that this routing daemon sends to its neighbors or receives from its neighbors.</td>
</tr>
<tr>
<td>Subject Data</td>
<td>Log all messages that this routing daemon forwards to its neighbors or receives from its neighbors.</td>
</tr>
</tbody>
</table>

To configure the destination of log output, see Log Destination on page 54.
To interpret the content of log output, see Routing Daemon Logging on page 122.
This page lets you configure routing table entries (router names). For more information, see Routing Table Entry on page 83.

To display this page, click Routers in the left margin of any page of the rvrd browser administration interface.

Identify each routing table entry by a globally unique name.

You can add a new entry or remove an existing entry at any time. (However, border routing introduces restrictions; see Border Routing on page 149.)

For background information, see Routing Table Entry on page 83, and Independent Routing Table Entries in One Process on page 98.

**Figure 39  rvrd Routers Configuration Page**

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing Routers</td>
<td>This panel lists the routing table entries within this routing daemon process. Each row represents one routing table entry.</td>
</tr>
</tbody>
</table>
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.

To configure a process as a border router, type the new router name, and click the Add Border Router button.
You cannot remove a border router from an rvrd process.
Local Network Interfaces Configuration

This page lets you configure local networks for a routing table entry. To display this page, click the number of local networks in a row of the Routers page.

For background information, see Local Network on page 84.

This page is not the same as the page described in Local Networks on page 138.

*Figure 40  rورد Local Network Interfaces Configuration Page*

<table>
<thead>
<tr>
<th>Local Network Name</th>
<th>Service</th>
<th>Network Specification</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunfire-bigdog-n1</td>
<td>7666</td>
<td>224.9.9.9</td>
<td>1</td>
</tr>
</tbody>
</table>

(Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existing local networks</td>
<td>The upper table lists local networks. Each row represents one local network.</td>
</tr>
</tbody>
</table>
Local Network Name: The name of a local network. Local network names must be globally unique.

To configure subject gating for a local network, click its name in the table of existing local networks.

For more information, see Local Network on page 84.

Service: The UDP or PGM service for communication on a local network. Programs within the local network communicate using this service.

For more information, see Specifying the UDP or PGM Service on page 21.

Network Specification: The network specification for a local network.

For more information, see Constructing the Network Parameter on page 23.

Cost: Path cost for routing between a local network and the routing daemon.

For more information, see Load Balancing on page 95.

Add Local Network Interface: To add a new local network, type the specifications and click this button.
Subject Gating

This page lets you configure subject gating (import and export subjects) for a local network.

To display this page, click the name of a local network in a row of the Local Network Interfaces Configuration page.

For background information, see Subject Gating on page 85, and Subject Filtering with Wildcards on page 85.

Figure 41   rvrd Subject Configuration (Gating) Page

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import Subjects</td>
<td>This table lists import subjects. The local network can import subjects that match these names. You can remove a subject at any time.</td>
</tr>
</tbody>
</table>
Export Subjects

This table lists export subjects. The local network can export subjects that match these names. You can remove a subject at any time.

**adding subjects**

To add subjects, specify the subject string (which may contain wildcards) here, and click one of three buttons:

- Import
- Export
- Import and Export

See also Subject Import Weight on page 96.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export Subjects</td>
<td>This table lists export subjects. The local network can export subjects that match these names. You can remove a subject at any time.</td>
</tr>
<tr>
<td>adding subjects</td>
<td>To add subjects, specify the subject string (which may contain wildcards) here, and click one of three buttons:</td>
</tr>
<tr>
<td></td>
<td>• Import</td>
</tr>
<tr>
<td></td>
<td>• Export</td>
</tr>
<tr>
<td></td>
<td>• Import and Export</td>
</tr>
</tbody>
</table>

See also Subject Import Weight on page 96.
This page lets you configure policy for a border router—that is, the subjects that the router forwards between its interfaces.

To display this page, click the phrase (border policy) following the name of a border router in the Routers page.

For background information, see Policy on page 111 (and the items that follow it), and Subject Filtering with Wildcards on page 85.

Figure 42  rvrd Border Policy Configuration Page

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>amading.non-tibco</td>
<td>7515.lan1.tib.com</td>
</tr>
</tbody>
</table>

Show Configuration

Allowed Subjects from amading.non-tibco to 7515.lan1.tib.com

<table>
<thead>
<tr>
<th>Subject</th>
<th>First Border</th>
</tr>
</thead>
<tbody>
<tr>
<td>_INBOX, &gt;</td>
<td></td>
</tr>
</tbody>
</table>

Remove Allowed Subject

Configure Selected Border

<table>
<thead>
<tr>
<th>Subject</th>
<th>First Border</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Add Allowed Subject
<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td>Choose a From interface from this menu.</td>
</tr>
<tr>
<td>To</td>
<td>Choose a To interface from this menu.</td>
</tr>
<tr>
<td>Show Configuration</td>
<td>To display the current set of subjects that the border router forwards from the From interface to the To interface, click this button. The Allowed Subjects table (below) then displays the current list for that ordered pair.</td>
</tr>
<tr>
<td>Allowed Subjects</td>
<td>This table lists subjects that the border router allows for the current pair of From interface and To interface. To update this table, click the Show Configuration button. You can remove a subject at any time.</td>
</tr>
<tr>
<td>Remove Allowed Subject</td>
<td>To delete an allowed subject, check its select box, and click this button.</td>
</tr>
<tr>
<td>Add Allowed Subject</td>
<td>To add an allowed subject, choose the From interface and To interface, then specify the subject string (which may contain wildcards) and click this button.</td>
</tr>
<tr>
<td>First Border</td>
<td>A border router can restrict a subject, forwarding only those messages that have not yet crossed another border. To restrict the new subject in this way, check the First Border box before adding the subject.</td>
</tr>
</tbody>
</table>
Neighbor Interfaces

This page lets you configure the potential neighbor connections of a routing table entry.

To display this page, click the number of neighbors in a row of the Routers page. For background information, see Neighbors on page 89.

Existing Neighbor Interfaces

The first part of this page is a table of existing neighbor interfaces—that is, interface specifications for potential neighbor connections to other routers.

Figure 43  rvrd Neighbor Interfaces Page—Existing

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existing neighbor interfaces</td>
<td>The upper table lists configured neighbor interfaces. Each row represents one potential neighbor.</td>
</tr>
<tr>
<td>Interface ID</td>
<td>The name of this neighbor interface. rvrd generates this name automatically, incorporating the router name.</td>
</tr>
</tbody>
</table>
Local Endpoint

This three-part string denotes the local end of the potential neighbor link. It has the form:

```
router_name@host:TCP_connect_port
```

- `router_name` is the name of the local routing table entry.
- `host` is a fixed token, `local_host`, which denotes the local `rvrd` host computer. (Note that this token does not denote the LOCALHOST loopback network address.)
- `TCP_connect_port` is the TCP port where the local router accepts neighbor connection requests from remote routers.

Remote Endpoint

This three-part string denotes the remote end of the potential neighbor link. It has the form:

```
router_name@host:TCP_connect_port
```

- `router_name` is the name of the remote routing table entry.
- `host` is the hostname or IP address of the remote `rvrd` host computer.
- `TCP_connect_port` is the TCP port where the local router attempts to connect to remote routers.

The token `Any` can appear in these three parts. For the semantics of this notation see [Accept Any as Neighbor on page 92](#), and [Seek Neighbor with Any Name on page 92](#). See also, [Four Variations of the Form on page 160](#).

Features

This column lists optional features of this neighbor specification:

- Cost: the path cost of this neighbor link (see [Load Balancing on page 95](#))
- Compression: this flag indicates whether this interface specifies data compression (see [Data Compression on page 90](#))
- SSL: this flag indicates whether this interface requires an SSL connection (see [SSL Connection with Compression on page 162](#))

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Endpoint</td>
<td>This three-part string denotes the local end of the potential neighbor link. It has the form:</td>
</tr>
<tr>
<td></td>
<td><code>router_name@host:TCP_connect_port</code></td>
</tr>
<tr>
<td></td>
<td>- <code>router_name</code> is the name of the local routing table entry.</td>
</tr>
<tr>
<td></td>
<td>- <code>host</code> is a fixed token, <code>local_host</code>, which denotes the local <code>rvrd</code> host computer. (Note that this token does not denote the LOCALHOST</td>
</tr>
<tr>
<td></td>
<td>loopback network address.)</td>
</tr>
<tr>
<td></td>
<td>- <code>TCP_connect_port</code> is the TCP port where the local router accepts neighbor connection requests from remote routers.</td>
</tr>
<tr>
<td>Remote Endpoint</td>
<td>This three-part string denotes the remote end of the potential neighbor link. It has the form:</td>
</tr>
<tr>
<td></td>
<td><code>router_name@host:TCP_connect_port</code></td>
</tr>
<tr>
<td></td>
<td>- <code>router_name</code> is the name of the remote routing table entry.</td>
</tr>
<tr>
<td></td>
<td>- <code>host</code> is the hostname or IP address of the remote <code>rvrd</code> host computer.</td>
</tr>
<tr>
<td></td>
<td>- <code>TCP_connect_port</code> is the TCP port where the local router attempts to connect to remote routers.</td>
</tr>
<tr>
<td>Features</td>
<td>This column lists optional features of this neighbor specification:</td>
</tr>
<tr>
<td></td>
<td>- Cost: the path cost of this neighbor link (see Load Balancing on page 95)</td>
</tr>
<tr>
<td></td>
<td>- Compression: this flag indicates whether this interface specifies data compression (see Data Compression on page 90)</td>
</tr>
<tr>
<td></td>
<td>- SSL: this flag indicates whether this interface requires an SSL connection (see SSL Connection with Compression on page 162)</td>
</tr>
</tbody>
</table>
Add New Neighbor Interface

The remainder of this page lets you complete a form to specify a new neighbor interface.

Figure 44   rvrd Neighbor Interface Configuration Form

Please supply a local port, as well as a remote host, port and router name.

Local Endpoint:
Host: <local_host>
Port: 7501
Router Name: Rick

Remote Endpoint:
Host: 
Port: 7501
Router Name: 

Connection Type:
Normal Connection: 
Data Compression without SSL: 
SSL Connection with Compression: 

Certificate of Expected Peer: 

Other Parameters:
Cost: 1

Add Neighbor Interface   Reset
Four Variations of the Form

Four buttons rearrange the form into four variations, each with a different meaning (see Table 12). In each variation, rvrd automatically fills in some fields, and leaves others empty for you to fill.

Table 12  Four Neighbor Interface Configuration Forms (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
</table>
| Accept Any | Use this variation of the form to specify a neighbor interface in which this routing daemon accepts neighbor connections from any other routing daemon. A distinguishing characteristic of accept any neighbors is a remote endpoint string in which the router name, the host and the port are all Any. Restrictions:  
• It is not possible to configure more than one accept any neighbor interface.  
• Accept any interfaces cannot use SSL neighbor connections.  
• Border routers cannot configure an accept any neighbor interface.  
For more information, see Accept Any as Neighbor on page 92. |
| Passive    | Use this variation of the form 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.  
A distinguishing characteristic of passive neighbors is a remote endpoint string in which the router name is specified, but the host and port are Any.  
For more information, see Passive Neighbor on page 91. |
| Active     | Use this variation of the form to specify a neighbor interface in which the local router actively attempts to connect to the remote neighbor.  
A distinguishing characteristic of active neighbors is a remote endpoint string in which the router name, the host and the port are all specified.  
For an example, see Active Neighbor on page 91. |
Table 12  Four Neighbor Interface Configuration Forms (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seek Any</td>
<td>Use this variation of the form to specify a neighbor interface in which this routing daemon attempts to connect to any remote routing daemon that matches the specification.</td>
</tr>
<tr>
<td></td>
<td>A distinguishing characteristic of seek any neighbors is a remote endpoint string in which the router name is Any, but the host and the port are specified. In addition, the local endpoint port is Any.</td>
</tr>
<tr>
<td><strong>Restrictions:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• It is illegal to configure two or more seek any neighbor interfaces with the same host.</td>
</tr>
<tr>
<td></td>
<td>• Seek any interfaces cannot use SSL neighbor connections.</td>
</tr>
<tr>
<td></td>
<td>• Border routers cannot configure a seek any neighbor interface.</td>
</tr>
<tr>
<td></td>
<td>For more information, see Seek Neighbor with Any Name on page 92.</td>
</tr>
</tbody>
</table>

**Items in the Neighbor Interface Configuration Form**

This table describes the items in Figure 44 on page 159.

(Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Endpoint</td>
<td>This three-part specification denotes the local end of the potential neighbor link:</td>
</tr>
<tr>
<td></td>
<td>• <em>Router Name</em> is the name of the local routing table entry. <em>rvrd</em> always automatically fills in this name.</td>
</tr>
<tr>
<td></td>
<td>• <em>Host</em> is a hostname or IP address corresponding to a network interface in the local <em>rvrd</em> host computer. For convenience, <em>rvrd</em> automatically fills in this field with the fixed token, <em>local_host</em>, which denotes the default network interface of the local <em>rvrd</em> host computer. (Note that this token does <em>not</em> denote the LOCALHOST loopback network address.) You may override this default value by typing an alternate hostname or IP address.</td>
</tr>
<tr>
<td></td>
<td>• <em>Port</em> is the local TCP port where the local router accepts neighbor connection requests from remote routers. For more information, see Local Connect Port on page 89.</td>
</tr>
</tbody>
</table>
Remote Endpoint

This three-part specification denotes the remote end of the potential neighbor link:
- **Router Name** is the name of the remote routing table entry.
- **Host** is the hostname or IP address of the remote rvrd host computer.
- **Port** is the remote TCP port where the local router attempts to connect to remote routers.

For more information, see Remote Connection Information on page 89.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal Connection</td>
<td>With this option, the two neighbors neither compress data nor use SSL protocols for communication on the link between them.</td>
</tr>
<tr>
<td>Data Compression without SSL</td>
<td>With this option, the two neighbors compress data on the link between them. To enable compression, you must select this option on both neighbors. For more information, see Data Compression on page 90.</td>
</tr>
<tr>
<td>SSL Connection with Compression</td>
<td>With this option, the two neighbors communicate using both compression and SSL protocols. To enable SSL, you must select this option on both neighbors—otherwise they cannot establish a connection. This option appears only in the Passive and Active variations of the configuration form. Connection statistics are not available when neighbors connect using SSL. See also Router Connection Statistics on page 141. In older releases of the routing daemon, SSL and compression are mutually exclusive features. For backward compatibility with older neighbors, this feature degrades gracefully to SSL without compression.</td>
</tr>
<tr>
<td>Certificate of Expected Peer</td>
<td>In SSL protocols, the local router expects the remote router to present this certificate as evidence of its identity. Paste the text of the public certificate (in PEM encoding) in this field. This field appears only in the Passive and Active variations of the configuration form.</td>
</tr>
<tr>
<td>Cost</td>
<td>The path cost of this neighbor link (see Load Balancing on page 95).</td>
</tr>
</tbody>
</table>
Certificates

This page lets you configure the X.509 certificates that the routing daemon uses to identify itself.

To display this page, click **Certificates** in the left margin of any page of the `rvrd` browser administration interface.

For background information, see **Certificates and Security on page 52** in *TIBCO Rendezvous Concepts*.

Each daemon process keeps a list of certificates it can use to identify itself. These certificates are numbered for easy reference. The first panel on this page determines which of these certificates the daemon uses for particular tasks. The remainder of the page lets you enter the certificates.

### Certificate Uses

*Figure 45  rvrd Certificate Uses Form*

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTPS (between daemon and Web browser)</td>
<td>Set the certificate for the secure browser administration interface.</td>
</tr>
<tr>
<td></td>
<td>To avoid security warnings from the web browser, distribute this certificate to authorized administrators.</td>
</tr>
<tr>
<td>Routers to Routers (between the routers defined in this daemon and their neighbors)</td>
<td>Set the certificate for the secure browser administration interface.</td>
</tr>
<tr>
<td></td>
<td>To avoid security warnings from the web browser, distribute this certificate to authorized administrators.</td>
</tr>
<tr>
<td></td>
<td>For security information, see <strong>Level of Trust—CA-Signed versus Self-Signed Certificates on page 176</strong>.</td>
</tr>
<tr>
<td>Item</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Routers to Routers</td>
<td>Set the certificate for secure SSL neighbor connections.</td>
</tr>
<tr>
<td></td>
<td>Distribute this certificate to each applicable neighbor.</td>
</tr>
</tbody>
</table>
Certificate List

Figure 46  rurd Certificate List

<table>
<thead>
<tr>
<th>Certificate #1</th>
</tr>
</thead>
<tbody>
<tr>
<td>You may either specify the location of a certificate file OR copy and paste the text of a certificate.</td>
</tr>
</tbody>
</table>

**Add from File**

| Pathname: |  |
| Password: |  |

**Note:** The daemon reads this file only once, when adding the certificate. After that, the certificate is permanently kept in the store file.

**Add from Text**

<table>
<thead>
<tr>
<th>Text:</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----BEGIN CERTIFICATE-----</td>
</tr>
<tr>
<td>MIIC6DCCAlGwIBAwIBATANBgkqhkiG9wOBAQQQFAzARBgNVBAgTCkNhbgGlmb3JuaWExEjAQBgNVBAcTQChMUVElCQ0gU29mdhcmUsIEElYy4xJTAjBgNV</td>
</tr>
<tr>
<td>dyXMgRW5naW51ZXJpbmcxHDAaBgNVBAMTE2JpZ2RVIFBgqkhkiG9wOBCQFWDmluZm9AWE1CQ05uY29tMB4GMDQxNzAwMDMyNl0wgbkxCzAJBgNVBAAYTA1VTMRAwHQYDVR0O</td>
</tr>
<tr>
<td>EMIwEAYDVQQHEw1QYXwvIEFsdG8xHTABoBgNVBAoTBmMuMSUwI1wYDVQQLEExuUSUDTyB8ZW5kZ2p2b3ZvZVQDExNiaWdkb2cuncYudGliY28yY29tMR0wGwYJKQ</td>
</tr>
<tr>
<td>QkNPLmNvbTCBnzANBgkqhkiG9wOBAQFADA0BjJAoAwggDQwIBADAdB0IBAQCQFWDmluZm9AWE1CQ05uY29tMB4GMDQxNzAwMDMyNl0wgbkxCzAJBgNVBAAYTA1VTMRAwH</td>
</tr>
<tr>
<td>QkNPLmNvbTCBnzANBgkqhkiG9wOBAQFADA0BjJAoAwggDQwIBADAdB0IBAQCQFWDmluZm9AWE1CQ05uY29tMB4GMDQxNzAwMDMyNl0wgbkxCzAJBgNVBAAYTA1VTMRAwH</td>
</tr>
</tbody>
</table>

| Password: |  |

**Add from Text**

Add from Text  Reset
Each daemon process creates a self-signed certificate at start time, and registers it in the list as certificate #1. You may use that certificate as is, add other certificates to the list, or delete it and enter other certificates. For security information, see Level of Trust—CA-Signed versus Self-Signed Certificates on page 176.

This self-signed certificate expires one year after creation.

You can also supply certificates signed by a certificate authority (CA). To use a CA-signed certificate, you must supply not only the certificate and private key, but also the CA’s public certificate (or a chain of such certificates). Concatenate these items in one file or string. For more details, see CA-Signed Certificates on page 175.

CA-signed certificates expire at dates recorded within the certificate data.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-Signed</td>
<td>Each daemon process creates a self-signed certificate at start time, and registers it in the list as certificate #1. You may use that certificate as is, add other certificates to the list, or delete it and enter other certificates. For security information, see Level of Trust—CA-Signed versus Self-Signed Certificates on page 176.</td>
</tr>
<tr>
<td>CA-Signed</td>
<td>You can also supply certificates signed by a certificate authority (CA). To use a CA-signed certificate, you must supply not only the certificate and private key, but also the CA’s public certificate (or a chain of such certificates). Concatenate these items in one file or string. For more details, see CA-Signed Certificates on page 175.</td>
</tr>
<tr>
<td>Certificate number</td>
<td>Use this number to refer to the certificate in the Certificate Uses panel.</td>
</tr>
<tr>
<td>Add from File</td>
<td>Enter a file name and a private key password. When you click Add from File, the daemon reads the certificate with private key from the file. The file may be in either PEM encoding, or PKCS #12 format. See also Security Factors on page 175.</td>
</tr>
<tr>
<td>Add from Text</td>
<td>Paste the text of a certificate with private key. Enter a private key password. The certificate must be in PEM encoding. See also Security Factors on page 175.</td>
</tr>
</tbody>
</table>
Chapter 6  
Secure Daemons (rvsd and rvsrd)

These two new daemons use SSL for secure connections to client program transports:

- \texttt{rvsd}, the Rendezvous secure communications daemon, corresponds to \texttt{rvd}
- \texttt{rvsrd}, the Rendezvous secure routing daemon, corresponds to \texttt{rvrd}

This chapter describes the security features of these two daemons, and details the parameters that differentiate them from their non-secure counterparts.

Topics

- Secure Daemon Overview, page 168
- Motivation, page 169
- Users, page 171
- Limiting Access, page 173
- Security Factors, page 175
- Behavioral Differences, page 178
- \texttt{rvsd}, page 180
- \texttt{rvsrd}, page 184
- Browser Administration Interface—\texttt{rvsd} and \texttt{rvsrd}, page 188
Secure Daemon Overview

This chapter describes the two daemons that offer secure client connections:

- **rvsd**, the Rendezvous secure communications daemon, corresponds to **rvd**. Chapter 4 describes **rvd**, the Rendezvous communications daemon.
- **rvsrd**, the Rendezvous secure routing daemon, corresponds to **rvrd**. Chapter 5 describes **rvrd**, the Rendezvous routing daemon.

Secure Connections

The two ordinary Rendezvous daemons, **rvd** and **rvrd**, communicate with clients over non-secure TCP connections. In contrast, their secure counterparts, **rvsd** and **rvsrd**, communicate with clients over SSL connections, allowing secure client communication over non-secure networks.

Restricting Access

Secure daemons restrict client access in three ways:

- Only authorized clients can connect to a secure daemon.
- Secure daemons restrict the combinations of network and UDP or PGM service over which client transports can communicate.
- Secure daemons limit the subject space that its clients can access.

Plaintext Communication

Although they ensure secure client connections, both secure daemons transmit messages as plaintext. That is, when they publish messages from clients to local networks, those messages are not encrypted.
Motivation

Deploy secure daemons when clients must connect securely over a non-secure network. This section illustrates example situations involving remote clients.

rvsd

Figure 47 depicts a hub and spoke architecture. An rvsd hub runs on a firewall computer, and remote programs access the hub through secure SSL connections. This arrangement lets trusted remote programs communicate with servers and other programs inside the secure inner network. rvsd bars untrusted programs from connecting to it.

rvsrd

Figure 48 rvsrd—Secure Connections across Double Firewall
Figure 48 on page 169 depicts a situation with two Rendezvous routing daemons configured to cross a double firewall. Remote programs initiate secure SSL connections to a secure routing daemon hub (rvsd) within the outer firewall (DMZ network). A secure SSL neighbor link connects that secure routing daemon with an ordinary routing daemon (rvrd) in the secure inner network.

To configure secure neighbor links, see SSL Connection with Compression on page 162.

Preventing Multicast in the DMZ

To prevent rvsrd from multicasting client messages within the DMZ network, start rvsrd with the -no-multicast option. For background information, see Disabling Multicast on page 38.

-no-multicast is available starting with Rendezvous release 7.2. This feature replaces the following procedure, which was required in earlier releases:

- Configure rvsrd so that in all of its local networks, the network specification is the loopback address (IP address 127.0.0.1). To configure, see Local Network Interfaces Configuration on page 151.

- Similarly limit the access of client transports to network and service pairs in which the network is the loopback address (IP address 127.0.0.1). To configure, see Authorize Network and Service Pairs on page 201.
Users

Each secure daemon instance authorizes a set of trusted users:

- The secure daemon allows a client transport to connect only if the client presents valid identification as an authorized user.
- User identification can be either a certificate, or a user name and password.

To authorize a user, see Users on page 199.

To connect to a secure daemon as a user, see Secure Daemon on page 60 in TIBCO Rendezvous Concepts, and the appropriate functions or methods in each programming language API.

Certificate Identification

The secure daemon can register zero or more X.509 public key identity certificates per user. The secure daemon limits access to user programs that can sign SSL protocol messages with a corresponding private key.

The secure daemon accepts all certificates in either PEM encoding or PKCS #12 format.

For more details, see CA-Signed Certificates on page 175.

User Name and Password Identification

The secure daemon registers at most one password per user. The secure daemon limits access to user programs that supply a correct pair of user name and password strings.

For important information about password security, see Security Factors on page 175.

Syntax

User name and password strings must conform to these syntax specifications:

- The user name must be less than 128 characters. The combined length of the user name and password must be less than 250 characters.
- These strings must consist of printable characters only, from any character set. Dot (.), star (*), and greater-than (>)) characters are permitted. However, we recommend against using them except in legacy situations (for example, where such names are already in use in another security system).
• These strings cannot contain two adjacent space characters.
• The first and last characters must not be spaces.
• These strings must contain at least one non-space character.
• These strings cannot contain embedded newline characters (\n) or null characters.
• The null or empty string is not a legal user name nor password.
Limiting Access

A secure daemon controls user access to local communications. Administrators can limit access at two levels of granularity:

- Network and service
- Subject

Network and Service Authorization

Each secure daemon allows its users to communicate over a set of local networks. Two parameters together define a local network:

- Network Specification
  
  For details, see Constructing the Network Parameter on page 23.
- UDP or PGM Service
  
  For details, see Specifying the UDP or PGM Service on page 21.

You must explicitly authorize each local network by specifying these two parameters. To authorize a local network, see Authorize Network and Service Pairs on page 201.

Users can communicate only on the local networks that you authorize. A user program cannot create a client transport that specifies an unauthorized local network (the transport create call produces an error status code).

Default Local Network

As an administrator, you can designate a default local network. A client transport that does not specify particular network and service parameters automatically communicates over this default local network; see Default Network and Service on page 197.

Subject Authorization

Each secure daemon allows its users to communicate using a set of Rendezvous subject names.

- Subjects authorized for sending can flow from client transports out to local networks.

  A client transport that sends a message with an unauthorized subject does not receive any error indication; instead, the secure daemon silently discards the message.
Subject authorization applies equally to all users and all local networks.

All `_INBOX` subjects are implicitly authorized. It is not necessary to explicitly authorize `_INBOX` subjects.

To authorize secure daemon subjects, see Authorize Subjects on page 202.

If clients use fault tolerance, certified message delivery, or distributed queue features, you must authorize the appropriate administrative subjects; see these tables:

- Critical Subjects for Certified Delivery on page 404
- Critical Subjects for Fault Tolerance on page 409
- Critical Subjects for Distributed Queues on page 412
Security Factors

Store Files

The secure daemon store file contains very sensitive information. Store it on the local file system of the secure daemon’s host computer, with tight file access, in a physically secure environment. Ensure timely backup to secure media.

Core-Dump Files

Secure daemon process storage contains sensitive information in unencrypted form. Similarly, user program storage can contain passwords or private key data. It is essential to deny access to these processes and their core image files. We strongly recommend arranging operating system parameters to prevent creation of core files.

To guard against attacks, take these precautions:

- Configure the operating environment to avoid making core dumps.
- Configure the operating environment to prevent access to process memory (if possible).
- Ensure that file system storage is secure.

Daemon Certificates

Administrators must implement a secure mechanism to distribute the secure daemon’s public certificate to users (that is, either programmers of client programs or end users).

Ensure that users verify daemon certificates before using them with client programs. Ensure that users keep daemon certificates in files that are secure from unauthorized modification or tampering. Remember, a false certificate can give a rogue daemon access to user passwords.

CA-Signed Certificates

Rendezvous does not support separating a leaf certificate from its signing CA certificate. When arranging certificate data, you may supply either a self-signed certificate, or a complete certificate trust chain, including leaf, intermediate (which are optional), and root certificates—all in one certificate data file. In either
case, the entire certificate chain is contained in one package, and Rendezvous components verify trust by comparing the entire package.

To better understand the way in which Rendezvous uses certificates and certificate trust chains, compare it to the familiar model of web browser security. In the familiar model, web browsers generally store a set of certificates representing trusted certificate authorities (CAs), and use them to deduce the authenticity of many certificates—any certificate signed using one of those trusted CA certificates.

In contrast, to authenticate a user (or another daemon), Rendezvous secure daemons require that a client-supplied certificate must exactly match a trusted certificate previously stored with the daemon. Daemons use certificates to verify digital signatures and message integrity, but they do not use CA certificates to authenticate client certificates. Similarly, Rendezvous clients verify certificates from Rendezvous daemons by matching them against trusted certificates previously registered with the client program.

If you require CA-signed certificates, or if your organization already uses CA-signed certificates, you may use them by packaging each one together with its CA root certificate and intermediate certificates—a complete trust chain for each certificate. You can use standard certificate utilities to create certificate files in the appropriate encoding formats.

**Level of Trust—CA-Signed versus Self-Signed Certificates**

When client connects to a daemon, both forms of certificate (CA-signed and self-signed) represent equivalent levels of trust.

- The daemon accepts the client’s certificate only if the daemon is configured to accept that certificate as the identity of a valid user, or as the identity of another trusted daemon.
- The client accepts the daemon’s certificate only if the client has previously registered that certificate as the identity of a trusted daemon.

In these situations, self-signed certificates can be more convenient than CA-signed certificates, with no degradation in security.

However, when a browser connects to a daemon’s browser administration interface, CA-signed daemon certificate chains do represent a stronger level of trust than self-signed daemon certificates. Furthermore, using CA-signed daemon certificates can help avoid browser security warnings.
Passwords

Private key files use password-encryption for security. Nonetheless, these files are important points of vulnerability.

To guard against attacks, ensure that file system storage is secure, and keep all passwords secure.

- Do not store passwords in non-secure files or on non-secure file systems.
- Control access to sensitive files—even when those files are password-encrypted.
- Never hard-code passwords in application programs, nor accept them as command line parameters.
- Code programs to erase passwords from process storage before exiting.
- Never write passwords in convenient locations.
- Never send passwords in plaintext messages.
- Choose passwords carefully.
Behavioral Differences

Secure daemons exhibit slight differences in behavior from their non-secure counterparts. This section summarizes those differences.

Automatic Start and Stop

`rvd` can start either automatically or by explicit command. In contrast, administrators must start `rvsd` by explicit command.

`rvd` can stop automatically after an interval in which it has no clients (see `rvd` on page 42, and `-no-permanent` on page 44). In contrast, `rvsd` does not stop automatically.

Subject Gating

Secure daemons are silent when subject gating parameters preclude send or listen operations:

- Subjects authorized for *sending* can flow from client transports out to local networks.

  A client transport that sends a message with an unauthorized subject does not receive any error indication; instead, the secure daemon silently discards the message.

- Subjects authorized for *listening* can flow to client transports from local networks.

  A client transport that creates a listener with an unauthorized subject does not receive any error indication—but the resulting listener object never receives any messages.

Default Network and Service

Secure daemons and non-secure daemons behave differently when a client transport specifies a default value (that is, null) for its network or service parameter. Non-secure daemons use *external* defaults; see Specifying the UDP or PGM Service on page 21 and Constructing the Network Parameter on page 23. In contrast, secure daemons use *internal* defaults—which you can configure using the browser administration interface; see Default Network and Service on page 197.
Browser Connections

Secure daemons automatically open both HTTP and HTTPS ports for browser administration interface connections—unless you specify otherwise. When an HTTPS connection is available, the daemon uses it; that is, whenever possible, it transfers non-secure HTTP communication over to its secure HTTPS connection.

You can block the secure HTTPS connection by specifying `-http-only`, which leaves only the non-secure HTTP connection.

You can block all browser administration interface connections by specifying `-no-http`.

See Also

Network and Service Authorization on page 173
Default Network and Service on page 197
rvsd

Command

Syntax

```plaintext
rvsd -store filename
[-http [ip_address:]http_port]
[-https [ip_address:]https_port]
[-http-only]
[-https-only]
[-no-http]
[-listen [socket_protocol:]ip_address:]tcp_port]
[-no-multicast]
[-reliability time]
[-max-consumer-buffer size]
[-rxc-max-loss loss]
[-rxc-recv-threshold bps]
[-rxc-send-threshold bps]
[-rvdm control_channel]
[-rvdm-reverse-asym]
[-reuse-port inbox_port]
[-logfile log_filename]
[-log-max-size size]
[-log-max-rotations n]
[-log-config config_log_filename]
[-foreground]
```

Purpose

The command `rvsd` starts the Rendezvous secure communications daemon process—the secure counterpart to `rvd`.

Remarks

This section describes only those aspects where `rvsd` differs from `rvd`. For details that both daemons share, see `rvd` on page 42.

Although `rvd` usually starts automatically, administrators must start `rvsd` by explicit command.

Command Line Parameters
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-store filename</td>
<td>This file contains the security parameters that configure rvsd. rvsd reads this file when the process starts, and writes this file each time you change the configuration using the browser administration interface. The secure daemon store file contains very sensitive information. Store it on the local file system of the secure daemon’s host computer, with tight file access, in a physically secure environment. Ensure timely backup to secure media. See also Appendix B, Store Files, on page 419.</td>
</tr>
<tr>
<td>-http ip_address:http_port</td>
<td>The browser administration interface accepts connections on this HTTP or HTTPS port. Permit administration access only through the network interface specified by this IP address.</td>
</tr>
<tr>
<td>-http http_port</td>
<td>To limit access to a browser on the rvsd host computer, specify 127.0.0.1 (the local host address). When the IP address is absent, the daemon accepts connections through any network interface on the specified HTTP or HTTPS port. If the explicitly specified HTTP port is already occupied, the program exits. If the explicitly specified HTTPS port is already occupied, the program selects an ephemeral port. When the -http parameter is entirely absent, the default behavior is to accept connections from any computer on HTTP port 7580; If this default port is unavailable, the operating system assigns an ephemeral port number. When the -https parameter is entirely absent, the default behavior is to accept secure connections from any computer on an ephemeral HTTPS port. In all cases, the program prints (in its start banner and log file) the actual HTTP and HTTPS ports where it accepts browser administration interface connections.</td>
</tr>
<tr>
<td>-https ip_address:https_port</td>
<td>Disable HTTPS (secure) connections, leaving only an HTTP (non-secure) connection.</td>
</tr>
<tr>
<td>-https https_port</td>
<td>Disable HTTPS (secure) connections, leaving only an HTTP (non-secure) connection.</td>
</tr>
</tbody>
</table>
Disable HTTP (non-secure) connections, leaving only an HTTPS (secure) connection.

Disable all HTTP and HTTPS connections, overriding -http and -https.

rvsd (and by extension, rvsrd operating within the local network) opens an SSL client socket to establish communication between itself and its client programs. The -listen parameter specifies the SSL port where the Rendezvous daemon listens for connection requests from client programs. This -listen parameter of the secure daemon corresponds to the daemon parameter of the transport creation call (they must specify the same SSL port number).

The IP address specifies the network interface through which this daemon accepts SSL connections.

To bar connections from remote programs, specify IP address 127.0.0.1 (the loopback interface).

When the IP address is absent, the daemon accepts connections from any computer on the specified SSL port.

When this parameter is entirely absent, the default behavior is to accept connections from any computer on SSL port 7500.

For more detail about the choreography that establishes conduits, see Daemon Client Socket—Establishing Connections on page 28.

This parameter does not correspond to the service parameter of the transport creation call—but rather to the daemon parameter.

Send duplicate log output to this file for log items that record configuration changes. The daemon never rotates nor removes this special log file. Instead, this file remains as a record of all configuration changes.

When absent, the default is stderr.
These parameters are the same as for `rvd`. For details, see Command Line Parameters on page 43.
**rvsrd**

*Command*

**Syntax**

```
rvsrd -store filename
[-http [ip_address:]http_port]
[-https [ip_address:]https_port]
[-http-only]
[-https-only]
[-no-http]
[-idle]
[-listen [socket_protocol:]ip_address:]tcp_port]
[-no-multicast]
[-reliability time]
[-max-consumer-buffer size]
[-rvdm control_channel]
[-rvdm-reverse-asym]
[-reliability time]
[-reliability time]
[...]
```

**Purpose**
The command **rvsrd** starts the Rendezvous secure routing daemon process—the secure counterpart to **rvrd**.

**Remarks**
This section describes only those aspects where **rvsrd** differs from **rvrd**. For details that both daemons share, see **rvrd on page 124**.

Administrators must start **rvsrd** by explicit command.

**Command Line Parameters**
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-store filename</td>
<td>This file contains the security parameters that configure <code>rvsrd</code>, as well as the routing table entry and parameters that configure its routing daemon behavior. <code>rvsrd</code> reads this file when the process starts, and writes this file each time you change the configuration using the browser administration interface. The secure daemon store file contains very sensitive information. Store it on the local file system of the secure daemon’s host computer, with tight file access, in a physically secure environment. Ensure timely backup to secure media. See also Appendix B, Store Files, on page 419.</td>
</tr>
<tr>
<td>-http ip_address:http_port</td>
<td>The browser administration interface accepts connections on this HTTP or HTTPS port. Permit administration access only through the network interface specified by this IP address.</td>
</tr>
<tr>
<td>-http http_port</td>
<td>To limit access to a browser on the <code>rvsrd</code> host computer, specify <code>127.0.0.1</code> (the local host address).</td>
</tr>
<tr>
<td>-https ip_address:https_port</td>
<td>When the IP address is absent, the daemon accepts connections through any network interface on the specified HTTP or HTTPS port. If the explicitly specified HTTP port is already occupied, the program exits. If the explicitly specified HTTPS port is already occupied, the program selects an ephemeral port. When the <code>-http</code> parameter is entirely absent, the default behavior is to accept connections from any computer on HTTP port 7580; If this default port is unavailable, the operating system assigns an ephemeral port number. When the <code>-https</code> parameter is entirely absent, the default behavior is to accept secure connections from any computer on an ephemeral HTTPS port. In all cases, the program prints (in its start banner and log file) the actual HTTP and HTTPS ports where it accepts browser administration interface connections.</td>
</tr>
<tr>
<td>-https https_port</td>
<td></td>
</tr>
</tbody>
</table>
Disable HTTPS (secure) connections, leaving only an HTTP (non-secure) connection.

Disable HTTP (non-secure) connections, leaving only an HTTPS (secure) connection.

Disable all HTTP and HTTPS connections, overriding -http and -https.

rvsd (and by extension, rvsrd operating within the local network) opens an SSL client socket to establish communication between itself and its client programs. The -listen parameter specifies the SSL port where the Rendezvous daemon listens for connection requests from client programs. This -listen parameter of the secure daemon corresponds to the daemon parameter of the transport creation call (they must specify the same SSL port number).

The IP address specifies the network interface through which this daemon accepts SSL connections.

To bar connections from remote programs, specify IP address 127.0.0.1 (the loopback interface).

When the IP address is absent, the daemon accepts connections from any computer on the specified SSL port.

When this parameter is entirely absent, the default behavior is to accept connections from any computer on SSL port 7500.

For more detail about the choreography that establishes conduits, see Daemon Client Socket—Establishing Connections on page 28.

This parameter does not correspond to the service parameter of the transport creation call—but rather to the daemon parameter.
These parameters are the same as for `rvrd`. For details, see Command Line Parameters on page 124.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-idle</code></td>
<td></td>
</tr>
<tr>
<td><code>-reliability time</code></td>
<td></td>
</tr>
<tr>
<td><code>-max-consumer-buffer size</code></td>
<td></td>
</tr>
<tr>
<td><code>-rxc-max-loss loss</code></td>
<td></td>
</tr>
<tr>
<td><code>-rxc-recv-threshold bps</code></td>
<td></td>
</tr>
<tr>
<td><code>-rxc-send-threshold bps</code></td>
<td></td>
</tr>
<tr>
<td><code>-rvdm control_channel</code></td>
<td></td>
</tr>
<tr>
<td><code>-rvdm-reverse-asym</code></td>
<td></td>
</tr>
<tr>
<td><code>-reuse-port inbox_port</code></td>
<td></td>
</tr>
<tr>
<td><code>-logfile log_filename</code></td>
<td></td>
</tr>
<tr>
<td><code>-log-max-size size</code></td>
<td></td>
</tr>
<tr>
<td><code>-log-max-rotations n</code></td>
<td></td>
</tr>
<tr>
<td><code>-log-config config_log_filename</code></td>
<td></td>
</tr>
<tr>
<td><code>-foreground</code></td>
<td></td>
</tr>
</tbody>
</table>
Browser Administration Interface—rvsd and rvsrd

The browser administration interface lets you control rvsd and rvsrd from a web browser. You can configure their operating parameters and view internal data structures.

This section describes only those pages specific to the secure daemons. For information about pages they share with their non-secure counterparts, see Browser Administration Interface—rvd on page 56, and Browser Administration Interface—rvrd on page 132.

Topics

- Navigation, page 189
- General Information, page 193
- Daemon Parameters, page 195
- Users, page 199
- Authorize Network and Service Pairs, page 201
- Authorize Subjects, page 202
- Certificates, page 204
Navigation

All browser administration interface pages display a navigation panel at the left side of the page. Use these links to display other pages.

**Figure 49  rosd Navigation Panel**

*State:*
- General Information
- Clients
- Services

*Daemon Manager:*
- Subject Map
- Port Map

*Configuration:*
- Daemon Parameters
- XML Configuration
- Users
- Networks and Services
- Subjects
- Certificates

*Miscellaneous:*
- Current Log
- Copyright
- TIBCO Rendezvous Web Page
Figure 50  rvsrd Navigation Panel

State:
- General Information
- Clients
- Local Networks
- Connected Neighbors
- Services
- Daemon Manager:
  - Subject Map
  - Port Map
Configuration:
- Daemon Parameters
- Routers
- XML Configuration
- Users
- Networks and Services
- Subjects
- Certificates
Miscellaneous:
- Current Log
- Copyright
- TIBCO Rendezvous Web Page
<table>
<thead>
<tr>
<th>Category</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>General Information</td>
<td>This page displays information about an <code>rvsd</code> or <code>rvsrd</code> process; see General Information on page 193.</td>
</tr>
<tr>
<td>Clients</td>
<td></td>
<td>This page summarizes the client transports; see Clients on page 61.</td>
</tr>
<tr>
<td>Local Networks</td>
<td></td>
<td>This page summarizes the local networks of a router; see Local Networks on page 138.</td>
</tr>
<tr>
<td>Connected Neighbors</td>
<td></td>
<td>This page summarizes the actual neighbor connections of a router; see Connected Neighbors on page 140.</td>
</tr>
<tr>
<td>Services</td>
<td></td>
<td>This page summarizes network services activity; see Services on page 64.</td>
</tr>
<tr>
<td>Daemon Manager</td>
<td>Subject Map Summary</td>
<td>This page summarizes RVDM subject maps by service number; see Subject Map Summary on page 71.</td>
</tr>
<tr>
<td></td>
<td>Port Map</td>
<td>This page displays the RVDM port map; see Port Map on page 74.</td>
</tr>
</tbody>
</table>
### Configuration

<table>
<thead>
<tr>
<th>Category</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration</td>
<td>Daemon Parameters</td>
<td>This page lets you configure parameters that control configuration access and secure default values for service and network parameters; see Daemon Parameters on page 195. For rvsrd, this page also configures router logging; see Logging on page 147.</td>
</tr>
<tr>
<td>Routers</td>
<td></td>
<td>This page lets you configure routers. You can access additional configuration pages through links on this page. See Routers on page 148, and the sections that follow it.</td>
</tr>
<tr>
<td>XML Configuration</td>
<td></td>
<td>This page lets you view the current configuration as an XML document, and reconfigure the component by submitting an edited XML document.</td>
</tr>
<tr>
<td>Users</td>
<td></td>
<td>These pages let you register authorized users; see Users on page 199.</td>
</tr>
<tr>
<td>Networks and Services</td>
<td></td>
<td>This page lets you configure the network and service pairs that client transports can use for Rendezvous communication; see Authorize Network and Service Pairs on page 201.</td>
</tr>
<tr>
<td>Subjects</td>
<td></td>
<td>This page lets you configure the subjects that client transports of a secure daemon can use for sending or listening; see Authorize Subjects on page 202.</td>
</tr>
<tr>
<td>Certificates</td>
<td></td>
<td>This page lets you configure certificates that the daemon uses to identify itself in secure protocols. See Certificates on page 163.</td>
</tr>
<tr>
<td>Log Out</td>
<td></td>
<td>This item logs out the current user or administrator. See Log Out on page 147.</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Current Log</td>
<td>This page displays the most recent 4 kilobytes from the log file.</td>
</tr>
<tr>
<td></td>
<td>Copyright</td>
<td>The Rendezvous copyright page.</td>
</tr>
<tr>
<td></td>
<td>TIBCO Rendezvous Web Page</td>
<td>The product page from the TIBCO web site.</td>
</tr>
</tbody>
</table>
rvsd and rvsrd (like all Rendezvous components) display information about themselves on this page.

To display this page, click General Information in the left margin of any page of the secure daemon browser administration interface.

*Figure 51  rvsd General Information Page*

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component</td>
<td>The name of the program—rvsd or rvsrd.</td>
</tr>
<tr>
<td>version</td>
<td>Version number of the program.</td>
</tr>
<tr>
<td>license ticket</td>
<td>The license ticket that validates this process.</td>
</tr>
<tr>
<td>host name</td>
<td>The hostname of the computer where the daemon process runs. Notice that the daemon process can run on one computer, while you access its browser interface from another computer.</td>
</tr>
<tr>
<td>user name</td>
<td>The user who started the daemon process.</td>
</tr>
<tr>
<td>IP address</td>
<td>The IP address of the computer where the daemon process runs.</td>
</tr>
<tr>
<td>IP address</td>
<td>The IP address of the computer where the daemon process runs.</td>
</tr>
</tbody>
</table>

(Sheet 1 of 2)
<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>client port</td>
<td>The SSL port where the daemon listens for client connections.</td>
</tr>
<tr>
<td>network services</td>
<td>The number of network services on which this daemon’s clients communicate.</td>
</tr>
<tr>
<td>store file</td>
<td>File name of the daemon’s store file; see the command line parameter <code>-store</code> for <code>rvsd</code> on page 180, and for <code>rvsrd</code> on page 184.</td>
</tr>
<tr>
<td>process ID</td>
<td>The operating system’s process ID number for the component.</td>
</tr>
<tr>
<td>managed</td>
<td>A link indicates that this daemon is managed (see Chapter 7, Daemon Manager, on page 209). The link is the name of an RVDM instance; to view or interact with the instance, click this link. Otherwise, this daemon is a non-managed daemon.</td>
</tr>
<tr>
<td>control channel</td>
<td>The RVDM control channel specification of a managed daemon; see <code>-rvdm</code> on page 47.</td>
</tr>
<tr>
<td></td>
<td>For non-managed daemons, this field displays the string <code>disabled</code>.</td>
</tr>
</tbody>
</table>
Daemon Parameters

This page lets you configure parameters that affect overall daemon security.

To display this page, click **Daemon Parameters** in the left margin of any page of the secure daemon browser administration interface.

For **rvsd**, this page contains two areas—**Administrator and Password** panel, and a **Default Network and Service** panel. For **rvsrd**, this page adds a third panel for logging parameters; see **Logging on page 147**.

Administrator and Password

*Figure 52  Secure Daemon Administrator and Password Area*

Only authorized personnel have administrative access to secure daemons. (In contrast, to configure client program user names, see **Users on page 171**.)

When administrator identification information is *not* set, anyone who can connect to the browser administration interface can examine and reconfigure the daemon. This arrangement can be useful during initial configuration and testing phases. However, during regular operation we recommend limiting access.
Once administrator identification information is registered, the browser administration interface is locked against unauthorized access. The daemon prompts administrators to prove identity by typing a name and password. After providing proper identification, an authorized administrator is logged in, and has complete access to configure the daemon. If the administrator does not provide proper identification, the browser displays the General Information page and continues to prompt for a correct name and password.

Browsers remember administrator name and password information for the duration of the browser process. Merely closing the browser window does not erase this information. To guard against intruders you must terminate the browser process (all its windows).

### Primary Administrator

The first administrator to register is called the primary administrator. In addition to configuring the daemon, the primary administrator can also add, delete and modify identification information pertaining to the other administrators.

Each daemon configuration can store up to 16 additional administrator name and password pairs (after the primary administrator).

### One Administrator Session

Each daemon process permits only one administrator session at a time. When one administrator is logged in, other administrators are locked out; this prevents conflicts in which two administrators attempt to modify the configuration at the same time. To terminate a administrator session, see Log Out (below).

---

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type a name string.</td>
</tr>
<tr>
<td>Password</td>
<td>Type a password string.</td>
</tr>
<tr>
<td>Confirm Password</td>
<td>Type the password again.</td>
</tr>
<tr>
<td>Add/Update</td>
<td>Specify a name and password, then click this button to add a new user.</td>
</tr>
<tr>
<td></td>
<td>This action is available only to the primary administrator.</td>
</tr>
</tbody>
</table>
Log Out

To end an administrative session, click **Log Out** in the left margin of the browser administration interface. This item appears only when you are logged in as an Administrator.

Daemons automatically log out administrator sessions that have been idle for 10 minutes.

Default Network and Service

*Figure 53  Secure Daemon Default Network and Service*

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete</td>
<td>Click this button to delete administrator identification information.</td>
</tr>
<tr>
<td></td>
<td>This action is available only to the primary administrator.</td>
</tr>
<tr>
<td></td>
<td>Deleting the primary administrator also deletes all other administrators.</td>
</tr>
</tbody>
</table>
Secure daemons and non-secure daemons behave differently when a client transport specifies a default value for its network or service parameter. Non-secure daemons use external defaults; see Specifying the UDP or PGM Service on page 21 and Constructing the Network Parameter on page 23. In contrast, secure daemons use internal defaults—which you can configure using this panel.

Unless you explicitly set values for these default parameters, they remain null—indicating the absence of any default value.

When either default is absent, the secure daemon will refuse connections from programs that rely on the default. As a result the transport creation call in the program fails.

For background information, see Network and Service Authorization on page 173.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>Type the default network.</td>
</tr>
<tr>
<td>Service</td>
<td>Type the default UDP or PGM service.</td>
</tr>
</tbody>
</table>

See Also

Default Network and Service on page 178
Users

This page lets you configure the set of users that can connect to a secure daemon. (In contrast, to configure administrative users, see Administrator and Password on page 195.)

To display this page, click Users in the left margin of any page of the secure daemon browser administration interface.

For background information, see Users on page 171.

Figure 54  Secure Daemon Users Page
Add a New User

This panel lets you create new users.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Name</td>
<td>Required. Every user must have a unique name, distinct from every other user that this secure daemon administers. For syntax rules governing user names, see User Name and Password Identification on page 171. To add a user, type the user name, and click the Add User button.</td>
</tr>
</tbody>
</table>

Existing Users

This list displays the users currently authorized to connect to the secure daemon. Buttons operate on selected users from the list.

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password</td>
<td>Displays a page that lets you view and set the password for the selected user.</td>
</tr>
<tr>
<td>Certificates</td>
<td>Displays a series of pages that let you view and set public certificates for the selected user.</td>
</tr>
<tr>
<td>Remove Selected Users</td>
<td>Deletes one or more selected users from the list.</td>
</tr>
</tbody>
</table>
Authorize Network and Service Pairs

This page lets you configure the network and service pairs that users can access through the secure daemon.

To display this page, click **Networks and Services** in the left margin of any page of the secure daemon browser administration interface.

*Figure 55  Secure Daemon Authorize Network and Service Pairs Page*

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network &amp; Service Pairs</td>
<td>This table lists the pairs of network and service that all authenticated users may access through this secure daemon.</td>
</tr>
<tr>
<td></td>
<td>To remove a pair from this list, click to check its Select box, then click the <strong>Remove Selected Network &amp; Service Pairs</strong> button.</td>
</tr>
<tr>
<td>Add</td>
<td>To add access to a network and service pair, type the specifications and click the Add button.</td>
</tr>
</tbody>
</table>
Authorize Subjects

This page lets you configure the Rendezvous subjects that users can access through the secure daemon.

To display this page, click **Subjects** in the left margin of any page of the secure daemon browser administration interface.

For background information, see **Subject Authorization on page 173**.

**Figure 56  Secure Daemon Authorize Subjects Page**

<table>
<thead>
<tr>
<th>Select</th>
<th>Authorized to Listen</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>foo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Select</th>
<th>Authorized to Send</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>jax</td>
</tr>
</tbody>
</table>

Remove Selected Subjects  Reset

**_INBOX** subjects are implicitly authorized both for listening and sending. You do not need to authorize them explicitly on this page.

(Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorized to Listen</td>
<td>This table lists subjects to which authenticated users may subscribe.</td>
</tr>
<tr>
<td>Authorized to Send</td>
<td>This table lists subjects to which authenticated users may send messages.</td>
</tr>
</tbody>
</table>
(Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove Selected Subjects</td>
<td>To remove authorization for particular subjects, select the affected subjects and click this button.</td>
</tr>
<tr>
<td>Subject</td>
<td>To add access to a subject, type the subject here and click one of three buttons:</td>
</tr>
<tr>
<td></td>
<td>• Authorize to Listen</td>
</tr>
<tr>
<td></td>
<td>• Authorize to Send</td>
</tr>
<tr>
<td></td>
<td>• Authorize to Listen and Send</td>
</tr>
</tbody>
</table>
Certificates

This page lets you configure the X.509 certificates that a secure daemon uses to identify itself.

To display this page, click Certificates in the left margin of any page of the rvsd or rvsrd browser administration interface.

For background information, see Certificates and Security on page 52 in TIBCO Rendezvous Concepts.

Each daemon process keeps a list of certificates it can use to identify itself. These certificates are numbered for easy reference. The first panel on this page determines which of these certificates the daemon uses for particular tasks. The remainder of the page lets you enter the certificates.

Certificate Uses

Figure 57  rvsrd Certificate Uses Form

<table>
<thead>
<tr>
<th>For</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTPS (between daemon and Web browser)</td>
<td>Certificate #1</td>
</tr>
<tr>
<td>Routers to Routers (between the routers defined in this daemon and their neighbors)</td>
<td>Certificate #1</td>
</tr>
<tr>
<td>Daemon to Clients (between this daemon and applications)</td>
<td>Certificate #1</td>
</tr>
</tbody>
</table>

(Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTPS</td>
<td>Set the certificate for the secure browser administration interface.</td>
</tr>
<tr>
<td></td>
<td>To avoid security warnings from the web browser, distribute the public portion of this certificate to authorized administrators.</td>
</tr>
</tbody>
</table>
Routers to Routers
Set the certificate for secure SSL neighbor connections.
Distribute the public portion of this certificate to each applicable neighbor.
(This item is included in routing daemons only; it is absent from rvsd.)

Daemon to Clients
Set the certificate for secure SSL client transport connections.
Distribute the public portion of this certificate to each client program; see Secure Daemon on page 60 in TIBCO Rendezvous Concepts.
Certificate List

*Figure 58  rvsrd Certificate List*

**Certificate #1**

You may either specify the location of a certificate file **OR** copy and paste the text of a certificate.

**Add from File**

| Pathname: | 
| Password: | 

**Note:** The daemon reads this file only once, when adding the certificate. After that, the certificate is permanently kept in the store file.

**Add from Text**

<table>
<thead>
<tr>
<th>Text:</th>
</tr>
</thead>
<tbody>
<tr>
<td>------BEGIN CERTIFICATE-----</td>
</tr>
<tr>
<td>MlC6DCCA1GgAwIBAwIBATANBgkqhkiG9wOBAQQQFAzARAGbNVBAgTCkNhGlbmb3JuaWExEjAQBgNVBAcT</td>
</tr>
</tbody>
</table>
| ChMUVE1CQ08gU29mdHdhcmUsIElucXJjaGFtYmEgNTAwMDEwMjA2NzE=
| dXNgRz5naW51ZXJmbmcxHDAaBgNVAMTE2JpZ2RvdW0gBglghkkiG9wOBCQFEWDmluZm9HAPE1CQ08uY29tMB4X |

| Password: |

**Add from Text**  
**Reset**
<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>certificate number</td>
<td>Use this number to refer to the certificate in the Certificate Uses panel.</td>
</tr>
<tr>
<td>Add from File</td>
<td>Enter a file name and a private key password. When you click Add from File, the daemon reads the certificate with private key from the file. The file may be in either PEM encoding, or PKCS #12 format. See also Security Factors on page 175.</td>
</tr>
<tr>
<td>Add from Text</td>
<td>Paste the text of a certificate with private key. Enter a private key password. The certificate must be in PEM encoding. See also Security Factors on page 175.</td>
</tr>
</tbody>
</table>

**Self-Signed Certificate**

When the daemon creates its store file (the first time it starts), it also creates a self-signed certificate, and registers it in the list as certificate #1. You may use that certificate as is, add other certificates to the list, or delete it and enter other certificates.

The self-signed certificate expires one year after creation.

**CA-Signed Certificate**

You can also supply certificates signed by a certificate authority (CA). To use a CA-signed certificate, you must supply not only the certificate and private key, but also the CA’s public certificate (or a chain of such certificates). Concatenate these items in one file or string. For more details, see CA-Signed Certificates on page 175.

CA-signed certificates expire at dates recorded within the certificate data.
Chapter 7  **Daemon Manager**

Rendezvous daemon manager (RVDM) lets administrators coordinate and manage the behavior of Rendezvous daemons throughout an enterprise.

Although it has a similar name, this feature component is not related to the `DaemonManager` class of the configuration tools API.

**Topics**

- Overview of RVDM, page 210
- Managing Network Topology with RVDM, page 212
- Subject Map, page 214
- Port Map, page 223
- RVDM Server Operating Details, page 228
- Administrative Tasks, page 230
- `rvdm`, page 233
- Browser Administration Interface—RVDM Control Center, page 236
- User Login, page 237
- Navigation Tabs, page 238
- Home Tab, page 239
- Port Map Tab, page 240
- Subject Maps Tab, page 243
- Update, page 247
- Daemons Tab, page 250
- Incidents Tab, page 254
- Options Tab, page 255
- RVDM JMX MBeans, page 259
Overview of RVDM

Background

In its initial, distributed management paradigm, Rendezvous application programs determined network topology. Each program directed its data through the physical network by supplying parameters to Rendezvous transport objects. This distributed management paradigm is effective for many enterprises because it is simple, flexible and incurs only low overhead.

Motivation

However, in other enterprises, distributing control over the network can result in ineffective or inefficient use of resources. Centralizing network management can help resolve these issues.

Rendezvous daemon manager (RVDM) implements a centralized management paradigm, in which administrators manage network topology and resources, coordinating the behavior of Rendezvous daemons throughout an enterprise.

Component Architecture

RVDM consists of four parts that work in concert (see Figure 59 on page 211):

- **RVDM server** is a web application that stores daemon management information, and distributes it to managed daemons.

- **RVDM control center** is a browser administration interface for the server. Administrators use this interface to configure and manage daemon behavior.

- **JMX MBean** is a JMX interface in the server (new in release 8.2). Administrators can use this interface to configure daemon behavior.

  The JMX interface requires Java 1.6 (or later).

- **Managed daemons** are rvd instances that cooperate with the RVDM server.

  The RVDM server and managed daemons communicate over an RVDM control channel. (Administrators must configure this control channel in the RVDM server and in every managed daemon.)

Managed Daemons

In order to use RVDM, administrators must first configure Rendezvous communications daemons to interact with the RVDM server over a dedicated control channel. Rendezvous daemons configured in this way are called managed daemons; daemons not configured in this way are called non-managed daemons.

Instances of rvd, rvsd, rvrd and rvsrd are eligible to operate as managed daemons.

The RVDM server coordinates all managed daemons so that they behave according to the configuration that you specify.
Backward Compatibility

The Rendezvous daemon is a crucial component of RVDM. Daemons from release 8.0 (and later) can coexist in the same network with daemons from earlier releases, however, the following conditions restrict interoperability:

- Daemons earlier than release 8.0 cannot operate as managed daemons.
- Managed daemons and non-managed daemons cannot communicate on a service for which you have configured a subject map. This restriction prevents non-managed daemons from interfering with managed daemons (but it does not completely prevent the reverse). For details, see Mixed Environment—Subject Maps on page 220.
- Managed daemons and non-managed daemons can communicate on a service for which you have not configured any subject map. This feature lets you roll-out managed daemons gradually throughout a large enterprise, while maintaining interoperability—managing network topology only as the final step.
Managing Network Topology with RVDM

Network topology and data flow affect network behavior and performance. You can use RVDM to manage network topology.

Motivation

Separating Delivery from Transport

To understand the motivation for central management of network topology, consider this analogy. The customers of a railroad freight company are concerned that their shipments arrive on schedule and in good condition; they are usually not interested in choosing the exact tracks upon which their freight travels. In contrast, railroad track managers can improve speed, safety and resource efficiency by thoughtfully assigning trains to specific tracks.

A similar division of interests applies in some large information enterprises. Application developers and administrators are concerned that data arrives in a timely fashion, but the actual details of network transport are usually not of particular interest. Developers and administrators may assign transport services piecemeal or arbitrarily, and only later discover a more efficient way to isolate application data or to allocate network bandwidth. RVDM is a tool that lets administrators globally reassign application data to network services and multicast groups, in order to improve performance, scalability or flexibility.

Multicast Addressing

Multicast addressing lets you isolate data according to application destinations. Ordinary broadcast addressing requires all receiving Rendezvous daemons to examine each inbound data packet and filter by service and subject. With multicast addressing, sending daemons can pre-sort outbound data into multicast groups. Receiving host computers can efficiently ignore irrelevant packets at hardware and operating system levels—passing exactly the correct subset of packets to the receiving Rendezvous daemon. In a network that carries data from several unrelated applications, appropriate use of this technique can reduce daemon load, dropped packets and retransmissions, improving processor and I/O utilization, application performance, and overall network performance.

RVDM lets you administer multicast addressing from a centralized control point, instead of separately modifying many applications or application instances.

Network Topology Tools

RVDM includes two tools for managing network topology:

- A subject map instructs managed daemons to sort and separate message traffic on a specific Rendezvous service into multicast groups, based on message
subjects and listener subjects. You can use subject maps to improve network efficiency (as described above). For details, see Subject Map, page 214.

A managed service is a service for which RVDM manages its behavior (for example, with a subject map).

- The port map globally overrides the service (that is, port) specifications of application transports. You can use the port map to shift message traffic from an original service (which application developers specify in transport creation calls) to an effective service (which you configure with RVDM). For details, see Port Map on page 223.

**Transparency**

RVDM network topology tools are transparent to application programs in the sense that you do not need to recompile or relink application programs in order to use RVDM. Instead, managed daemons rearrange message flow according the configuration that you specify.

For example, consider existing application programs that specify transports to communicate over service 7500, and exchange messages with subjects Foo and Bar. An administrator can configure a subject map with service number 7500, such that messages with subject Foo actually traverse the network on multicast group 244.1.1.5, while messages with subject Bar traverse the network on multicast group 244.1.1.9.

For network communication, the daemons separate the two subjects into the specified multicast groups, while the transport objects within sending and listening applications remain unchanged. Mapping occurs entirely within the daemons that mediate between those applications and the network.
The primary tool for reorganizing Rendezvous network topology is the subject map.

The term subject map is an abbreviation for subject-to-multicast-group map. A subject map defines a mapping from Rendezvous subjects to multicast groups. When a managed daemon applies a subject map to a managed service, it sorts and separates messages into multicast groups, based on the message subjects. You can use subject maps to improve network efficiency.

You can configure subject maps in the RVDM control center; the RVDM server pushes the configuration to all its managed daemons, which do the actual sorting. You may configure any number of subject maps. Each subject map applies to a specific service (port number).

Scope

Subject maps apply to multicast messages (including broadcast messages), reliable messages, certified messages, and Rendezvous protocol messages. However, they do not apply to point-to-point (inbox) messages, nor direct communication.

Subject maps apply to both daemon variants—TRDP and PGM.

Interaction with Port Map

When the port map feature is enabled, the service number of a subject map refers to an effective port (not an original client application port). That is, managed daemons apply the port map first, then apply subject maps afterward. (For background information, see Port Map, page 223. For an example, see Order of Operations, page 224.)

Specification

The following values define a subject map:

- a managed service, specified by its service number (that is, a UDP or PGM port number)
- a mapping table, which maps Rendezvous subject names to multicast groups
  - Each row in a mapping table includes three items:
    - a Rendezvous subject name (wildcards are permitted)
    - multicast groups
    - a relative rank (which serves as a tie-breaker when a message subject matches several rows)
• default system groups, for two categories of system messages:
  — all Rendezvous system messages (that is, subject names that begin with \_RV)
  — other distinguished subjects (for example, \_RVCM and \_RVFT) that do not match any row of the mapping table
• default user groups, for non-system messages with any other subject name (that is, subjects that do not match any row of the mapping table, and are not Rendezvous system messages)

Note that the service number implies a specific and unique network interface as well (for more information, see Interaction between Service and Network Parameters, page 20).

Example

To understand the concepts of subject maps, consider the two example subject maps in Table 13.

Each subject map specifies a fine-grained division of the message flow within its service. Because the action of a subject map is limited to a specific managed service (specified by a service number), the two subject maps are independent, and do not interfere with one another, even though some of their multicast group numbers overlap.

Table 13  RVDM Subject Maps, Example 1

<table>
<thead>
<tr>
<th>Service Number</th>
<th>Default System Groups (DS)</th>
<th>Default User Groups (DU)</th>
<th>Mapping Table Subject</th>
<th>Multicast Groups</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>7500</td>
<td>225.9.9.1 (DS)</td>
<td>225.0.0.1 (DU)</td>
<td>Foo.Bar</td>
<td>225.1.1.1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Foo.&gt;</td>
<td>225.1.1.2</td>
<td>2</td>
</tr>
<tr>
<td>7501</td>
<td>225.9.9.2 (DS)</td>
<td>225.0.0.2 (DU)</td>
<td>*.Foo</td>
<td>225.1.1.1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Foo.*</td>
<td>225.1.1.2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Foo.&gt;</td>
<td>225.1.1.3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bar.*</td>
<td>225.1.1.4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Baz.&gt;</td>
<td>225.1.1.1</td>
<td>5</td>
</tr>
</tbody>
</table>

Wildcard Subject  Subject names in the mapping table may contain wildcard elements.
Matching

Managed daemons use the following matching procedure to sort a subject into a multicast group, based on the specification of a subject map:

1. Compare the subject against the subject column in each row of the mapping table. The subject can match zero, one or several rows.
2. User subjects that do not match any row of the mapping table automatically match the default user group.
3. Distinguished subjects that match the wildcard pattern _RV.> always match the default system group—even if they match an explicit row of the mapping table.
4. All other distinguished subjects (for example, _RVM.> and _RVFT.>) map to the default system group only if they do not match an explicit row of the mapping table.

(For basic information about Rendezvous subject matching, see Using Wildcards to Receive Related Subjects, page 66 in TIBCO Rendezvous Concepts.)

Semantics of Subject Maps

In effect, a subject map instructs managed daemons to override the transport parameters in application programs, and to use multicast groups instead. The application transport’s service parameter selects the subject map of a specific managed service, then (for each message and each subscription) the matching procedure determines the corresponding multicast group.

Mapping subjects to multicast groups determines two complementary actions in managed daemons:

- **Listening**
  
  When a managed daemon has a client transport, and the transport’s service number has a subject map, and that transport creates a listener, then the daemon maps the listener subject to the matching multicast groups, and receives network data on those multicast groups.

- **Sending**
  
  When a managed daemon has a client transport, and the transport’s service has a subject map, and that transport sends a message, then the daemon maps the message subject to a multicast group, and sends that message over that multicast group.
If the message subject matches more than one row of the mapping table, the daemon nonetheless sends the message on exactly one multicast group, using the row with the lowest rank to determine the corresponding multicast group.

**Example**

To understand the effects of matching on message flow, consider again the example of Table 13 on page 215.

**Single Match**

When an application transport using service 7500 creates a listener (L1) on subject Foo.Bar, the managed daemon maps the subject to group 225.1.1.1, and listens only on that group.

When an application transport using service 7500 sends a message with subject Foo.Bar, the managed daemon maps the message to group 225.1.1.1. Listener L1 receives the message.

**Several Matches**

When an application transport using service 7500 creates a listener (L2) on subject Foo.*, the managed daemon maps the subject to groups 225.1.1.1 and 225.1.1.2, and listens on both groups.

When an application transport using service 7500 sends a message with subject Foo.Bar, even though the subject matches two rows, the managed daemon uses rank to map the subject only to group 225.1.1.1, and sends the message only on that group. Listener L2 receives the message through group 225.1.1.1.

When an application transport using service 7500 sends a message with subject Foo.Foo, the managed daemon maps the message to group 225.1.1.2. Listener L2 receives the message through group 225.1.1.2.

**No Match**

**User Subject**

When an application transport with service 7500 sends a message with subject Bar.Foo, the subject does not match any row of the subject map, so the managed daemon maps it to the default user group, 225.0.0.1.

**System Subject**

When an application transport using service 7500 generates a HOST.STATUS advisory message (for example, subject _RV.INFO.SYSTEM.HOST.STATUS.hostid), the managed daemon maps the message to the default system group, 225.9.9.1.

**See Also**

For information about multicast groups, see Constructing the Network Parameter on page 23.

For information about multicast performance, see Multicast Addressing on page 25.
Asymmetric Multicast

Subject Map Syntax

- **Symmetric** multicast—the same group for listening and sending. For example, 225.1.1.1.
- **Asymmetric** multicast—two multicast groups, one for listening and the other for sending. For example, 225.2.2.2;225.1.1.1 (the general form is `listen;send`).

Motivation

Some enterprises use asymmetric multicast to direct message traffic along one dimension. **Figure 60** depicts an example in which two cache applications cooperate for fault tolerance (top) and many users (bottom, only two are shown) interact with them.

**Figure 60  Asymmetric Multicast (with RVDM)**

Green arrows in **Figure 60** show the required message flow, while red lines (marked with X) indicate undesirable flow:

- Each user must communicate with both caches in order to request and receive data.
- Users must *not* communicate with other users, because users do not need to see cache requests from other users, and forcing the daemon to discard those request messages (based on subject names) could decrease useful throughput.
- The two caches must *not* communicate with each another, so the backup cache’s daemon need not discard responses from the primary cache to the
users. Nonetheless, the two caches use a separate channel (not shown) to coordinate for fault tolerance.

Asymmetric multicast offers one way to satisfy these constraints. Users send requests to the cache on multicast group 225.1.1.1, which we call the upward group, and listen for responses on multicast group 225.2.2.2, which we call the downward group. The two caches do the inverse—they listen for user requests on the upward group, and send responses on the downward group. Messages flow up and down along separate channels, but not horizontally.

-configuring-the-subject-map

If the request and response messages use different subject names, then the administrator could configure two separate rows in the subject map—mapping request subjects to the upward group, and response subjects to the downward group. When designing subject name conventions for an enterprise or application, we recommend this approach.

-reversing-the-subject-map

In contrast, if an existing subject name convention requires that some subjects must flow both upward and downward, the administrator must use a different tactic when configuring the subject map.

One row of the subject map matches these bi-directional subjects, and maps them to the asymmetric multicast groups 225.1.1.1;225.2.2.2—that is, listen on the downward group, and send on the upward group. This configuration is correct for the users in the example of Figure 60 on page 218.

However, the caches in Figure 60 must reverse this configuration—they must listen on the upward group and send on the downward group. To arrange this inversion, the administrator must supply the -rvdm-reverse-asym option when starting the managed daemons that serve the caches. This option flips the direction of asymmetric multicast in all rows of the subject map (that is, in all rows that specify asymmetric multicast groups—it does not affect rows that specify symmetric multicast).

see-also

-rvdm-reverse-asym on page 47
Mixed Environment—Subject Maps

We strongly discourage using mixed environments in actual practice. Nonetheless, for the sake of completeness, this section presents the behavior of subject maps in a mixed environment, as well as ways to detect and resolve interference.

Consider a mixed environment, in which managed daemons with a subject map coexist with non-managed daemons (or managed daemons without subject maps).

Managed daemons with a subject map (for a particular service) cannot communicate with daemons that do not have a subject map (on that service). This restriction prevents non-managed daemons from interfering with managed daemons.

Figure 61 depicts two daemons and their applications. The managed daemon on the left (rvd A) has a subject map that uses multicast group 225.1.1.1 for some of its subjects—in particular, the subjects of application A. In contrast, the daemon on the right (rvd B) does not have a subject map, but application B has a transport that uses multicast group 225.1.1.1.

Figure 61 Mixed Environment—Subject Maps

The red arrow with an X indicates that managed daemon A discards all messages from daemon B. Logic within daemon A prevents interference from daemon B.

However, the situation in the opposite direction is less absolute. Ideally, messages would not flow from daemon A flow to daemon B either. When daemon B is earlier than release 8.0, this flow can occur (green arrow with dashed line), and can even flow to application B (because it happens to listen to multicast group 225.1.1.1). While we cannot prevent this flow, we do not support it, because the Rendezvous daemon cannot ensure reliable delivery in this situation. (The same logic in daemon A that prevents interference from daemon B also discards retransmission requests from daemon B.)
Detecting Interference

Daemon A detects potential interference, and reports it so network administrators can eliminate it at the source. When daemon A receives packets (including retransmission requests) from daemon B, it recognizes that they are not governed by a subject map. Daemon A responds with three actions:

- It discards the packet (preventing interference).
- It reports a DATALOSS advisory to its client transports, and to its log file. The error description string includes the words multicast destination.
- It reports the incident to the RVDM server.

Resolving Interference

When you notice these advisories or incidents, take action to eliminate the source of the interference. Techniques to resolve the situation might include any of the following (in consultation with the administrator for application B):

- Terminate application B.
- Incorporate daemon B into the subject mapping regimen.

See Also

In contrast, the behavior of daemons in a mixed environment with respect to the port map is quite different; see Mixed Environment—Port Map on page 226.

Non-Network Messages

It should be intuitively obvious that subject maps affect only messages that traverse the network. To clarify this distinction, Table 14 on page 221 presents examples of exempt and affected message subjects.

Table 14  RVDM: Examples of Exempt and Affected Subjects (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Exempt Subject</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exempt Subjects</td>
<td></td>
</tr>
<tr>
<td>_INBOX.&gt;</td>
<td>Although they traverse the network, point-to-point messages are exempt from subject mapping (because they cannot be multicast).</td>
</tr>
<tr>
<td>_LOCAL.&gt;</td>
<td>Local messages flow from the sending client to other clients of the same rvd. Since they do not traverse the network, they are exempt from subject mapping.</td>
</tr>
<tr>
<td>_RV.ERROR.SYSTEM.CLIENT.*</td>
<td>Some advisory messages (such as this example) flow from the daemon to a specific client transport. Since these messages do not traverse the network, they are exempt from subject mapping.</td>
</tr>
</tbody>
</table>
When you define a subject map, the parameters of the subject map replace the network parameters of application transports on the same service as the subject map. This action can enable multicast communication between applications that previously could not communicate.

Conversely, when you delete a subject map, communication channels return to those defined by the parameters specified by the applications. However, subject maps do not govern point-to-point (inbox) messages, so point-to-point channels established between applications (while the subject map existed) remain viable even after you delete the enabling subject map.

Such point-to-point back channels can arise as an unexpected side-effect of certified message delivery (RVCM) and other Rendezvous protocols that use point-to-point communications. For example, once two applications establish a certified delivery agreement, they could continue to communicate over a point-to-point back channel even after you delete the subject map that brought them together. This form of communication is extremely inefficient, and could cause erratic application behavior and poor network performance. (We strongly discourage deliberate use of such back channels.

To close such back channels, restart the applications involved.

Table 14  RVDM: Examples of Exempt and Affected Subjects (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Exempt Subject</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_RVFT. &gt;</td>
<td>Administrative messages (such as these examples) traverse the network, so they are affected by subject mapping.</td>
</tr>
<tr>
<td>_RVCM. &gt;</td>
<td>Some advisory messages (such as this example) flow from a daemon to other daemons on the network. Since these messages traverse the network, they are affected by subject mapping.</td>
</tr>
<tr>
<td>_RVCMQ. &gt;</td>
<td></td>
</tr>
</tbody>
</table>

**Deleting a Subject Map—the Point-to-Point Back Channel**

When you define a subject map, the parameters of the subject map replace the network parameters of application transports on the same service as the subject map. This action can enable multicast communication between applications that previously could not communicate.

Conversely, when you delete a subject map, communication channels return to those defined by the parameters specified by the applications. However, subject maps do not govern point-to-point (inbox) messages, so point-to-point channels established between applications (while the subject map existed) remain viable even after you delete the enabling subject map.

Such point-to-point back channels can arise as an unexpected side-effect of certified message delivery (RVCM) and other Rendezvous protocols that use point-to-point communications. For example, once two applications establish a certified delivery agreement, they could continue to communicate over a point-to-point back channel even after you delete the subject map that brought them together. This form of communication is extremely inefficient, and could cause erratic application behavior and poor network performance. (We strongly discourage deliberate use of such back channels.

To close such back channels, restart the applications involved.
Port Map

The port map is a secondary tool for reorganizing Rendezvous network usage. The port map instructs managed daemons to shunt message traffic among Rendezvous service ports. You can use the port map to rearrange and rationalize network port usage.

You can configure the port map in the RVDM control center; the RVDM server pushes the configuration to all its managed daemons, which do the actual shunting.

You may configure one port map.

Scope

The port map applies to reliable messages, certified messages, Rendezvous protocol messages, multicast messages, and point-to-point (inbox) messages. However, they do not apply to direct communication.

The port map applies to both daemon variants—TRDP and PGM.

Interaction with Subject Maps

When the port map feature is enabled, the service numbers of subject maps refer to effective ports (not original client application ports). That is, managed daemons apply the port map first, then apply subject maps afterward. (For background information, see Subject Map, page 214. For an example, see Order of Operations, page 224.)

Service Pairs

The port map is a set of service pairs, which are ordered pairs with this signature:

\[(\text{original port}, \text{effective port})\]

- The original port is the service port number that the client transport specifies.
- The effective port is the service port number that the managed daemon uses for network communication on behalf of client transports that specify the corresponding original port.

Shunting

In effect, the port map overrides the service parameter of application transport objects.

More specifically, each service pair specifies two complementary actions in managed daemons:

- Listening
When a daemon has a client transport with a matching original port, and the client uses that transport to create a listener, then the daemon listens for network packets on the effective port instead.

- **Sending**

  When a daemon has a client transport with a matching original port, and the client sends a message on that transport, the daemon sends the message over the network on the effective port instead.

  **Non-Recursive**

  The port map is a non-recursive mapping. That is, each daemon applies at most one service pair to each client transport.

  For example, consider this port map:

  **Table 15 Port Map, Example 2a**

<table>
<thead>
<tr>
<th>Original Port</th>
<th>Effective Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>7500</td>
<td>7501</td>
</tr>
<tr>
<td>7501</td>
<td>7502</td>
</tr>
</tbody>
</table>

  If an application transport specifies service 7500, then the daemon translates it to 7501—and not to 7502.

  **Order of Operations**

  Daemons apply the port map first, and then apply the subject map that pertains to the effective port (if one exists).

  For example, consider the effect of combining the example port map in Table 15 (above), with the subject maps in Table 16:

  **Table 16 RVDM Subject Maps, Example 2b**

<table>
<thead>
<tr>
<th>Service Number</th>
<th>Default System Groups (DS)</th>
<th>Default User Groups (DU)</th>
<th>Mapping Table Subject</th>
<th>Multicast Groups</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>7500</td>
<td>225.9.9.0 (DS)</td>
<td>225.0.0.0 (DU)</td>
<td>Foo</td>
<td>225.2.2.0</td>
<td>1</td>
</tr>
<tr>
<td>7501</td>
<td>225.9.9.1 (DS)</td>
<td>225.0.0.1 (DU)</td>
<td>Foo</td>
<td>225.2.2.1</td>
<td>1</td>
</tr>
<tr>
<td>7502</td>
<td>225.9.9.2 (DS)</td>
<td>225.0.0.2 (DU)</td>
<td>Foo</td>
<td>225.2.2.2</td>
<td>1</td>
</tr>
</tbody>
</table>

  At the listener:
1. Client application L creates a transport on original port 7500. The listening daemon maps original port 7500 to effective port 7501.

2. Client application L creates a listener for subject Foo on that transport. The listening daemon applies the subject map for the effective port, which is service 7501, and listens to multicast group 225.2.2.1 on service 7501.

At the sender:

3. Client application S sends a message with subject Foo on service 7500. The sending daemon translates original port 7500 to effective port 7501.

4. The sending daemon applies the subject map for the effective port, which is service 7501, so the daemon sends the message on multicast group 225.2.2.1 on service 7501.

At the listener:

5. The message arrives at the receiving daemon on effective port 7501. The daemon directs the message to the transport with original port 7500, which in turn directs it to the client listener for subject Foo.

Notice that throughout this example, the managed daemon applies only the subject map with service number 7501.

### Permitted & Forbidden Service Pairs

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Permitted</strong></td>
<td></td>
</tr>
<tr>
<td>Symmetric Swap</td>
<td>It is permitted to symmetrically map original port M to effective port N, while simultaneously mapping original port N to effective port M.</td>
</tr>
<tr>
<td>Merge Many-to-One</td>
<td>It is permitted to map several original ports to the same effective port.</td>
</tr>
<tr>
<td><strong>Forbidden</strong></td>
<td></td>
</tr>
<tr>
<td>Reflexive</td>
<td>It is forbidden to reflexively map original port R to effective port R.</td>
</tr>
<tr>
<td>One-to-Many</td>
<td>It is forbidden to map an original port to several effective ports.</td>
</tr>
</tbody>
</table>
Caution

The port map feature has far-reaching ramifications. Before attempting to use the port map feature, **read and understand this section thoroughly.**

While the effect of a port map might seem straightforward, the actual consequences can be very complicated. This section presents operational aspects of the port map feature.

**Switchover**

Applying the port map (or applying changes to the port map) implicitly involves all managed daemons. Depending on the size of your enterprise, the effect might involve hundreds of daemons and thousands of transport objects—temporarily interrupting Rendezvous communications throughout the network.

RVDM coordinates the switchover with all managed daemons. Managed daemons destroy old transport and listener objects, and create new objects to reflect the current port map. Meanwhile, daemons buffer outbound data from clients (that is, data from send operations in client programs), with possible side effects:

- **Time** The switchover could delay communications for several milliseconds, or even several seconds.

  If the delay exceeds a daemon’s reliability time, data loss can occur. (For background information, see [Reliability and Message Retention Time on page 34](#).)

- **Space** While daemons buffer outbound data from clients, you can expect daemon process storage to grow accordingly.

- **Interaction** One daemon that is slow to switch prohibits all the others from resuming communications.

**Mixed Environment—Port Map**

We strongly discourage using mixed environments in actual practice. Nonetheless, for the sake of completeness, this section presents the behavior of the port map in a mixed environment.

Consider a mixed environment, in which managed daemons with a port map coexist with non-managed daemons (or managed daemons without the port map). These two types of daemons can and do interoperate—though (once again) we strongly discourage reliance on this behavior.
Figure 62 depicts two daemons and their applications. The managed daemon on the left (rvd C) has a port map, which maps a transport in application C to the effective port 7979. In contrast, the daemon on the right (rvd D) does not have a port map, but application D has a transport that uses port 7979. Messages flow in both directions between applications C and D on service 7979.

![Mixed Environment—Port Map](image)

See Also In contrast, the behavior of daemons in a mixed environment with respect to subject maps is quite different; see Mixed Environment—Subject Maps on page 220.
RVDM Server Operating Details

Daemon List

Each managed daemon sends periodic administrative messages to the RVDM server indicating its status. Based on these status messages, the RVDM server maintains a list of managed daemons, annotated with status information. You can view this list in the RVDM control center; see Daemons Tab on page 250.

Inactive Daemon

When the stream of status messages from a managed daemon is interrupted, the RVDM server marks that daemon as *inactive*.

When the stream of status messages resumes, the server marks the daemon as *active* once again.

Purge

The server periodically purges inactive daemons from the daemon list. At each purge, the server removes from the list any daemons that have been inactive for longer than the purge threshold.

Configuration Updates

When administrators modify the configuration of RVDM, the server pushes the new configuration to all managed daemons. As each daemon receives and installs the updated configuration, it confirms receipt to the RVDM server.

The server collects and tallies these confirmation messages. If any active daemon fails to confirm receipt, the server reports an error. When an inactive daemon becomes active again, the server ensures that the daemon receives and confirms the configuration update.

Files

*Table 18  RVDM Files (Sheet 1 of 2)*

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVDM.options.xml</td>
<td>This file contains all the parameter settings that the RVDM server requires in order to start. Each server must have a private copy of this file; RVDM servers cooperating as a fault-tolerant pair must not share it.</td>
</tr>
<tr>
<td>RVDM.data.xml</td>
<td>This file contains configuration data. RVDM servers cooperating as a fault-tolerant pair must both use a single copy of this file in a shared location. See also Storage on page 258.</td>
</tr>
</tbody>
</table>
Table 18  RVDM Files (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVDM.log.txt</td>
<td>RVDM server writes log data to this file, including incidents from managed daemons, significant events pertaining to server operation, and auditing information.</td>
</tr>
</tbody>
</table>
Administrative Tasks

This section outlines administrative tasks associated with RVDM, with references to more detailed information.

Establish a Control Channel

Before using RVDM, you must establish a control channel, over which the RVDM server communicates with managed daemons.

To configure the control channel in the RVDM server, see Control Channel on page 257.

To configure the control channel in a managed daemon, see -rvdm on page 47.

We strongly discourage specifying asymmetric multicast groups for the control channel. The added complexity of asymmetry would not add any benefit, since managed daemons send only point-to-point messages to the RVDM server.

Start Server

To start the RVDM server, see rvdm on page 233.

Configure the Server

To configure the RVDM server (and as a result, the behavior of managed daemons) see the sections summarized in this table.

<table>
<thead>
<tr>
<th>About the Configuration Interface</th>
<th>About Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject Maps Tab on page 243</td>
<td>Subject Map on page 214</td>
</tr>
<tr>
<td>Port Map Tab on page 240</td>
<td>Port Map on page 223</td>
</tr>
<tr>
<td>Options Tab on page 255</td>
<td>—</td>
</tr>
</tbody>
</table>

Start Managed Daemons

To start a managed daemon, see -rvdm on page 47.
Forward RVDM Protocol Messages across Network Boundaries

Correct operation of RVDM software depends on protocol messages between RVDM and the managed daemons.

These messages travel freely within a single network segment. However, if your network consists of several segments connected by Rendezvous routing daemons, then you must instruct the routing daemons to forward the subjects in Table 19.

Routing daemons must forward all these subjects in both directions—import and export.

Similarly, if clients in your network use SSL to connect to rvsd or rvsrd, you must configure the secure daemon to authorize the subjects in Table 40.

Table 19  Critical Subjects for RVDM

<table>
<thead>
<tr>
<th>Subject</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_RVDM. &gt;</td>
<td>In release 8.1.0 (and later) RVDM software uses messages with these subjects to communicate with managed daemons. Routing daemons must forward these subjects in both directions.</td>
</tr>
<tr>
<td>SMM. &gt;</td>
<td>In release 8.0.0, RVDM software used messages with these subjects to communicate with managed daemons. When the set of managed daemons includes any daemons from release 8.0.0, routing daemons must forward these subjects in both directions (in addition to _RVDM. &gt;).</td>
</tr>
</tbody>
</table>
Modifying the Subject Maps or Port Map

Modifications to subject maps or the port map can dramatically affect daemon performance and data latency. We strongly recommend that you make modifications only during periods of low network usage.

This section explains the potential consequences of updating managed daemons.

While managed daemons update their local subject maps and port map, they must buffer all data. Then they must resolve all the buffered data using the new mappings.

When the overall network publishing rate is high, this buffering could result in very high growth of process memory usage in managed daemons. Potential problems could include exhausting available memory, high retransmission rates exhausting network bandwidth, data loss.

To avoid these severe consequences, we strongly recommend choosing to update managed daemons during times of low overall network activity. To estimate overall network activity, you must take into account all publishers on all multicast groups.
**rvdm**

*Command*

**Syntax**

```
rvdm [-http http_port]
[-idle]
[-rvdm-home dir]
[-logfile log_filename]
[-purge-threshold time]
[-rmi-port rmi_port]
[-mbean-port mbean_port]
```

**Purpose**

The command *rvdm* starts the RVDM server web application.

**Location**

<table>
<thead>
<tr>
<th>OS</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td><code>rv_home/RVDM/rvdm.bat</code></td>
</tr>
<tr>
<td>UNIX</td>
<td><code>rv_home/RVDM/rvdm.sh</code></td>
</tr>
</tbody>
</table>

**Remarks**

The RVDM server is a Java web application. You can configure and deploy the web application directly in a web container, or you can use this shell script to start it.

Configure RVDM while Idle

To configure RVDM while delaying any changes to daemon behavior, start RVDM with the command line parameter `-idle`.

RVDM Requires JRE 1.6

The web application requires JRE 1.6 (or later). This prerequisite is not available on the z/OS and IBM i operating system platforms.

Fault Tolerance

RVDM servers on the same control channel automatically attempt run as fault-tolerant partners. However, to correctly configure fault-tolerant operation, all partners must access the same data directory (in a shared location); to configure, see Data Directory on page 258.
## Command Line Parameters

(Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `-http http_port` | The browser administration interface accepts browser connections on this HTTP port.  
If the explicitly specified port is already occupied, the program exits.  
When this parameter is entirely absent, the default behavior is to accept connections from any computer on HTTP port 8080.  
If this default port is unavailable, the operating system assigns an ephemeral port number. |
| `-idle`           | When present, start `rvdm` in its *idle* state. You can configure RVDM parameters, managed services and the port map—but RVDM does not communicate with managed daemons.  
(When the configuration is complete and satisfactory, stop RVDM and restart it without this command line argument.)  
When absent, start `rvdm` in its *running* state—managing daemons. |
| `-rvdm-home dir`  | RVDM server uses this directory as the default location for persistent storage.  
- `RVDM.options.xml` is always in this directory.  
- `RVDM.data.xml` is in this directory by default, unless you override it with another location (see Data Directory on page 258).  
- `RVDM.log.txt` is in this directory by default, unless you override its file pathname (see below).  
When absent, the default is the value of the JVM property `user.home`. |
| `-logfile log_filename` | Send log output to this file.  
When absent, the default is `RVDM.log.txt`, in the directory denoted by `-rvdm-home`. |
RVDM regularly purges inactive daemons from its internal list of managed daemons. An inactive daemon is one for which RVDM has not received a heartbeat during an interval that exceeds this value (in seconds).

This value must be greater than or equal to 15 seconds.

When absent, the default is 1 hour (3600 seconds).

For background information, see Daemon List on page 228.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-purge-threshold</td>
<td>RVDM regularly purges inactive daemons from its internal list of managed daemons. An inactive daemon is one for which RVDM has not received a heartbeat during an interval that exceeds this value (in seconds). This value must be greater than or equal to 15 seconds. When absent, the default is 1 hour (3600 seconds). For background information, see Daemon List on page 228.</td>
</tr>
<tr>
<td>-rmi-port</td>
<td>The RMI registry within RVDM listens on this port for RMI registry requests. When absent, the default value is 1099. When present, the port you supply overrides the default. We recommend using the default value unless it conflicts with another RMI registry on the same host computer.</td>
</tr>
<tr>
<td>-mbean-port</td>
<td>The MBean server within RVDM listens on this port for connection requests from JMX consoles. When absent, the default value is 9589. When present, the port you supply overrides the default. For background information, see JMX Ports on page 259.</td>
</tr>
</tbody>
</table>
Browser Administration Interface—RVDM Control Center

RVDM control center is the browser administration interface to RVDM server.

RVDM control center requires a web browser with Javascript. Any of the following are acceptable:

- Microsoft Internet Explorer 6 and later (with latest patches)
- Firefox 1.5 and later
- Safari

Topics

- Navigation Tabs, page 238
- Home Tab, page 239
- Port Map Tab, page 240
- Subject Maps Tab, page 243
- Update, page 247
- Daemons Tab, page 250
- Incidents Tab, page 254
- Options Tab, page 255
**User Login**

The graphical user interface of RVDM consists of two kinds of pages:

- Display pages show non-sensitive information. Any user can view these pages.
- Edit pages allow authorized administrators to modify RVDM parameters, and configure the behavior of managed daemons. These pages require login credentials.

The first time you start an RVDM server instance, two default users are configured. The user names are user1 and user2, and the password for both of them is CHANGE_ME. This arrangement can be useful during initial configuration and testing phases. However, during regular operation we recommend limiting access.

To ensure security, you *must* change these default users and passwords; see Users on page 258.

**Lockout**

RVDM accepts only one user login at a time. Other users are locked out (including the same user login from another browser).

Users must log out before closing the RVDM browser administration interface, otherwise all users will be locked out.

If you are certain that another user is not actively using RVDM, you can logout that user (overriding the lock-out feature).

RVDM automatically cancels an inactive login after approximately 60 minutes. Alternatively, you can restart the RVDM web application, which cancels an open login and begins again with a clean slate.
Navigation Tabs

RVDM’s browser administration interface pages display a horizontal panel of navigation tabs. Use these tabs to display other pages.

**Figure 63  RVDM Navigation Tabs**

<table>
<thead>
<tr>
<th>Tab</th>
<th>Description</th>
<th>See Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home</td>
<td>General information.</td>
<td>239</td>
</tr>
<tr>
<td>Port Map</td>
<td>View and edit the port map.</td>
<td>240</td>
</tr>
<tr>
<td>Subject Maps</td>
<td>View and edit subject maps.</td>
<td>243</td>
</tr>
<tr>
<td>Daemons</td>
<td>View managed daemons.</td>
<td>250</td>
</tr>
<tr>
<td>Incidents</td>
<td>View error incidents.</td>
<td>254</td>
</tr>
<tr>
<td>Options</td>
<td>View and edit general configuration values.</td>
<td>255</td>
</tr>
</tbody>
</table>
Home Tab

The Home tab displays general information about the RVDM server.

Figure 64  RVDM Home Tab

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daemon Manager</td>
<td></td>
</tr>
<tr>
<td>Control Channel Service</td>
<td>7777</td>
</tr>
<tr>
<td>Control Channel Network</td>
<td>;239.77.77.77</td>
</tr>
<tr>
<td>Control Channel Daemon</td>
<td>7500</td>
</tr>
<tr>
<td>Fault Tolerance Status</td>
<td>This instance is active.</td>
</tr>
</tbody>
</table>
Port Map Tab

For information about the port map feature and its use, see Port Map on page 223. Table 20 on page 241 describes the graphical elements of the interfaces in Figure 65 and Figure 66.

Figure 65 shows an example port map. To modify the port map, click the Edit Port Map link, which displays an editable perspective on the port map, like the example in Figure 66 on page 241. (Editing the RVDM configuration requires login authentication.)

Figure 65  RVDM Port Map Tab

<table>
<thead>
<tr>
<th>Original Port</th>
<th>Effective Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>7500</td>
<td>7501</td>
</tr>
<tr>
<td>7502</td>
<td>7503</td>
</tr>
<tr>
<td>7504</td>
<td>7505</td>
</tr>
<tr>
<td>7506</td>
<td>7507</td>
</tr>
<tr>
<td>7508</td>
<td>7509</td>
</tr>
</tbody>
</table>

| Edit Port Map |
**Table 20  RVDM Port Map Interface (Sheet 1 of 2)**

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enable</strong></td>
<td></td>
</tr>
<tr>
<td>Enable Port Map</td>
<td>Enable or disable the port map. Because the port map feature entails the risk of unintended side effects, the default value of this parameter is disable (that is, this box is not checked, so the port map feature is disabled). To use the port map, you must explicitly enable this parameter. Changes take effect when you click <strong>Apply</strong>.</td>
</tr>
<tr>
<td><strong>Port Map Table</strong></td>
<td>The following parameters configure the port map.</td>
</tr>
<tr>
<td>Original Port</td>
<td>These two port values constitute a service pair.</td>
</tr>
<tr>
<td>Effective Port</td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>To delete a service pair from the port map, check its Delete box. Changes take effect when you click <strong>Apply</strong>.</td>
</tr>
</tbody>
</table>
Table 20  RVDM Port Map Interface (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Buttons</td>
<td></td>
</tr>
<tr>
<td>Add 1 Mapping Row</td>
<td>To add a new service pair to the port map, click this button, and supply</td>
</tr>
<tr>
<td></td>
<td>values for the client and effective ports. Changes take effect when you click</td>
</tr>
<tr>
<td></td>
<td><strong>Apply</strong>.</td>
</tr>
<tr>
<td>Apply</td>
<td>To apply new configuration values, click the <strong>Apply</strong> button.</td>
</tr>
<tr>
<td></td>
<td>For information about the update sequence, see <strong>Update on page 247</strong>.</td>
</tr>
</tbody>
</table>
Subject Maps Tab

For information about the subject mapping feature and its use, see Subject Map on page 214.

Table 21 on page 243 describes the graphical elements of the interfaces in Figure 67.

Figure 67 shows an example subject map. To modify the subject map, click the Edit this Subject Map link, which displays an editable perspective on the subject map, like the example in Figure 68 on page 244. (Editing the RVDM configuration requires login authentication.)

To add a new subject map, edit a different subject map, or delete a subject map, click the Add, Edit or Delete link, which displays a screen like the example in Figure 69 on page 246. (Editing the RVDM configuration requires login authentication.)

Table 21 RVDM Subject Maps Interface (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Selector</td>
<td></td>
</tr>
<tr>
<td>Service</td>
<td>Select a specific subject map by specifying its service (port number) from the drop-down menu. (If the port map is enabled, this value represents the effective service.)</td>
</tr>
</tbody>
</table>
Table 21  RVDM Subject Maps Interface (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Map Table</strong></td>
<td>Each row maps a subject to a multicast group.</td>
</tr>
<tr>
<td>Subject</td>
<td>Map this subject name.</td>
</tr>
<tr>
<td>Multicast Groups</td>
<td>Map the service and subject name to these multicast groups. For syntax and</td>
</tr>
<tr>
<td></td>
<td>semantics, see Asymmetric Multicast on page 218.</td>
</tr>
<tr>
<td>Rank</td>
<td>Rank is relative, showing the current position of a row in the matching</td>
</tr>
<tr>
<td></td>
<td>order. Managed daemons use rank to resolve conflicts when a subject</td>
</tr>
<tr>
<td></td>
<td>matches more than one row; see Several Matches on page 217.</td>
</tr>
<tr>
<td>Delete</td>
<td>To delete a row from this subject map, check its Delete box. Changes take</td>
</tr>
<tr>
<td></td>
<td>effect when you click Apply.</td>
</tr>
<tr>
<td>Add 1 Mapping Row</td>
<td>To add a new row to the subject map, click this button, and supply values</td>
</tr>
<tr>
<td></td>
<td>for the subject, multicast group and rank.</td>
</tr>
<tr>
<td>Apply</td>
<td>To apply new configuration values, click Apply.</td>
</tr>
</tbody>
</table>

Figure 68  RVDM Subject Map—Edit

Subject Map for Service 7500

Default System Groups:
239.88.88.89;239.88.88.88

Default User Groups:
239.99.99.100;239.99.99.9

☑ Allow publishing to wildcard subjects on this service

<table>
<thead>
<tr>
<th>Subject</th>
<th>Multicast Groups</th>
<th>Rank</th>
<th>Delete</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7500:A</td>
<td>224.3.2.1;224.1.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7500:B</td>
<td>224.3.2.1;224.1.2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Add 1 Mapping Row
Apply
### Table 22  RVDM Subject Map Edit Interface (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Default Groups</strong></td>
<td></td>
</tr>
<tr>
<td>Default System Groups</td>
<td>Distinguished subjects that do not match any row of the mapping table automatically match one of these multicast groups. For syntax and semantics, see Asymmetric Multicast on page 218.</td>
</tr>
<tr>
<td>Default User Groups</td>
<td>Subjects that do not match any row of the mapping table automatically match one of these multicast groups. For syntax and semantics, see Asymmetric Multicast on page 218.</td>
</tr>
<tr>
<td><strong>Wildcard Publish</strong></td>
<td></td>
</tr>
<tr>
<td>Allow publishing to wildcards subjects on this service.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• When enabled, managed daemons allow their client transports to publish messages with wildcard subjects on this service. Messages with wildcard subjects bypass the map table, traversing the network on the default user group (or, with asymmetric multicasting, the user send group).</td>
</tr>
<tr>
<td></td>
<td>By enabling this feature, you force all managed daemons to automatically join the default user group, which could expose them to additional network load, decreasing the benefit of subject mapping. We discourage enabling this feature except in situations where wildcard publishing is absolutely essential.</td>
</tr>
<tr>
<td></td>
<td>More generally, we discourage publishing to wildcard subjects because such subjects can often be broader than intended, so that unrelated applications might receive messages that they cannot parse.</td>
</tr>
<tr>
<td></td>
<td>• When disabled, managed daemons prohibit their client transports from publishing messages with wildcard subjects on this service. If a client sends to a wildcard subject, the daemon logs an error, reports an error advisory, and reports the incident to RVDM.</td>
</tr>
<tr>
<td></td>
<td>See also, CLIENT.ILLEGAL_PUBLISH on page 264 in TIBCO Rendezvous Concepts.</td>
</tr>
<tr>
<td><strong>Mapping Table</strong></td>
<td></td>
</tr>
<tr>
<td>Subject</td>
<td>Map this subject name.</td>
</tr>
</tbody>
</table>

TIBCO Rendezvous Administration
Table 22  RVDM Subject Map Edit Interface (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multicast Groups</td>
<td>Map the subject name to these multicast groups. For syntax and semantics, see Asymmetric Multicast on page 218.</td>
</tr>
<tr>
<td>Rank</td>
<td>Optional.</td>
</tr>
<tr>
<td></td>
<td>Rank is relative, showing the current position of a row in the matching order. Managed daemons use rank to resolve conflicts when a subject</td>
</tr>
<tr>
<td></td>
<td>matches more than one row; see Several Matches on page 217.</td>
</tr>
<tr>
<td></td>
<td>You may specify a non-negative integer rank to resolve conflicts when a subject matches more than one row; see Several Matches on page 217.</td>
</tr>
<tr>
<td></td>
<td>When absent, RVDM server computes a default value by finding the greatest rank specified so far, and adding one to that number.</td>
</tr>
<tr>
<td></td>
<td>After each update, RVDM server displays the rows in the new rank order, numbering them with consecutive non-negative integers.</td>
</tr>
<tr>
<td>Delete</td>
<td>To delete a row from this subject map, check its Delete box. Changes take effect when you click <strong>Apply</strong>.</td>
</tr>
<tr>
<td>Add 1 Mapping Row</td>
<td>To add a new row to the subject map, click this button, and supply values for the subject, multicast group and rank.</td>
</tr>
<tr>
<td>Apply</td>
<td>To apply new configuration values, click <strong>Apply</strong>.</td>
</tr>
<tr>
<td></td>
<td>For information about the update sequence, see Update on page 247.</td>
</tr>
</tbody>
</table>

Figure 69  RVDM Subject Maps—Add, Edit or Delete
When you edit the port map or a subject map, and click the **Apply** button, RVDM sends the updated instructions to its managed daemons. Since all managed daemons must implement exactly the same instructions, they behave in a way that is similar to transaction semantics:

1. RVDM sends updated instructions to managed daemons.
2. As each daemon receives the update, it stores the new instructions and replies to RVDM to indicate receipt.
3. Upon a second signal from RVDM, all managed daemons either implement the new instructions simultaneously, or roll-back to the previous set of instructions.

Progress During step 2, RVDM displays the progress of the managed daemons, as in Figure 70. Table 23 on page 248 describes the elements of the progress display.

**Figure 70  RVDM Update Progress**

**Deploying update...**

Waiting for replies...30 seconds remaining.  
1 out of 3 daemons replied.

The following daemons have not replied...

`jms02, colbert.na.tibco.com`

The following incidents were reported...
The progress display lets you track the managed daemons as they receive the updated instructions. You can also control the update by forcing daemons to skip immediately to step 3, or rolling back to cancel the update.

### Table 23  RVDM Update

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>progress bar</td>
<td>Step 2 can continue for at most 60 seconds. The progress bar indicates the time elapsed and remaining. Below the progress bar, a text field indicates the number managed daemons that have replied, and the total number of managed daemons.</td>
</tr>
<tr>
<td>The following daemons have not replied...</td>
<td>This field lists the names of the daemon hosts for which RVDM has not yet received a reply.</td>
</tr>
<tr>
<td>The following incidents were reported...</td>
<td>This field displays incidents—for example, a daemon replied that it could not implement the update.</td>
</tr>
<tr>
<td>Update Immediately</td>
<td>Clicking this button circumvents the 60-second waiting period, and forces all daemons that have received the new instructions to implement the update immediately.</td>
</tr>
<tr>
<td>Rollback</td>
<td>Clicking this button cancels the update. Daemons continue implementing the previous instructions.</td>
</tr>
</tbody>
</table>

When the update completes (step 3), RVDM displays a report, as in Figure 71 on page 249. The report displays the final result of the update, and lists all incidents. You can use the report to determine whether any daemons did not successfully update their instructions.

The report includes links to the daemon involved in each exception incident. Clicking the link displays that daemon’s browser administration interface (if the daemon is running).
These incidents were reported during this update:

- Hostname: jms02
  Process ID: 30134
  Exception: Daemon had not replied when update was finalized.
  [Browse to this daemon](#)

- Hostname: colbert.na.tibco.com
  Process ID: 10885
  Exception: Daemon had not replied when update was finalized.
  [Browse to this daemon](#)

Your update has been confirmed.
Daemons Tab

The Daemons tab displays information about the managed daemons that the RVDM server controls. For background information, see Daemon List on page 228.

Figure 72 shows an example with one managed daemon. Table 24 describes the elements of the table.

Figure 72   RVDM Daemons Tab

<table>
<thead>
<tr>
<th>Host Name</th>
<th>Active Transports</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Port Number</td>
</tr>
<tr>
<td>col</td>
<td>7501</td>
</tr>
<tr>
<td></td>
<td>7675</td>
</tr>
<tr>
<td>arr</td>
<td>7675</td>
</tr>
<tr>
<td>j62</td>
<td>7501</td>
</tr>
<tr>
<td></td>
<td>7675</td>
</tr>
</tbody>
</table>

Table 24   RVDM Daemons Interface (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hosts</td>
<td>This table displays a row for each host computer with a managed daemon. Yellow highlighting of a row indicates an inactive daemon (see Inactive Daemon on page 228).</td>
</tr>
<tr>
<td>Host Name</td>
<td>Host computer where the managed daemon is running. For more details about a daemon, view its browser administration interface by clicking its host name. (This feature is not available for daemons that disallow the browser administration interface, nor for inactive daemons.)</td>
</tr>
<tr>
<td>Active Transports</td>
<td>For each managed daemon, this table displays a sub-row for each service that has client transports (whether or not subject maps or the port map affect those transports). Each row displays the following fields.</td>
</tr>
<tr>
<td>Port Number</td>
<td>Service (port). To view network packet statistics for a host and port, click the port number; see Daemon Port Statistics on page 251. (If the port map is enabled, this value represents the effective service.)</td>
</tr>
</tbody>
</table>
Table 24  RVDM Daemons Interface (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transports</td>
<td>Number of transports on that service.</td>
</tr>
<tr>
<td>Listeners</td>
<td>Number of listeners on that service (summed over all transports on that service).</td>
</tr>
</tbody>
</table>

Daemon Port Statistics

Figure 73 presents a table of network packet statistics for one service of a managed daemon. Table 25 on page 252 describes the elements of the table.

You can also view statistics for one multicast group within a service (see Group Statistics, below).

Statistical data arrive at 90-second intervals.

Figure 73  RVDM Daemon Statistics
Table 25  RVDM Daemon Statistics (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>The name of the daemon host computer. To view statistics for a different</td>
</tr>
<tr>
<td></td>
<td>daemon host, select its name from this drop-down menu.</td>
</tr>
<tr>
<td>Service</td>
<td>The service (port) number. To view statistics for a different service,</td>
</tr>
<tr>
<td></td>
<td>select its number from this drop-down menu. (If the port map is enabled,</td>
</tr>
<tr>
<td></td>
<td>this value represents the effective service.)</td>
</tr>
<tr>
<td>Purge Statistics</td>
<td>To delete all statistical records for this Host and Service, click this</td>
</tr>
<tr>
<td></td>
<td>link.</td>
</tr>
<tr>
<td>Age</td>
<td>The time that this daemon has been active on the service. If the daemon is</td>
</tr>
<tr>
<td></td>
<td>not currently active on the service, this item displays the time it has</td>
</tr>
<tr>
<td></td>
<td>been inactive, as well as the time it had previously been active.</td>
</tr>
<tr>
<td>Group</td>
<td>The multicast group number for which this table presents statistics. The</td>
</tr>
<tr>
<td></td>
<td>string All Groups indicates that statistics are summed over all multicast</td>
</tr>
<tr>
<td></td>
<td>groups on the service.</td>
</tr>
<tr>
<td>Group Statistics</td>
<td>When a service has a subject map, a separate table lists the multicast</td>
</tr>
<tr>
<td></td>
<td>groups for the service. To view statistics only for a specific multicast</td>
</tr>
<tr>
<td></td>
<td>group, click its group number in this table; the result is a similar table,</td>
</tr>
<tr>
<td></td>
<td>limited to a single group.</td>
</tr>
<tr>
<td>Count</td>
<td>This column shows statistics as a count; for example, the number of packets.</td>
</tr>
<tr>
<td></td>
<td>All counts are cumulative since the daemon became active on the service.</td>
</tr>
<tr>
<td>Rate</td>
<td>This column shows statistics as a rate; for example, packets per second.</td>
</tr>
<tr>
<td></td>
<td>All rates are calculated as the total cumulative count divided by the age of</td>
</tr>
<tr>
<td></td>
<td>the service (in seconds).</td>
</tr>
<tr>
<td>Messages Sent</td>
<td>Outbound messages that this daemon sent on this service (and multicast</td>
</tr>
<tr>
<td></td>
<td>group).</td>
</tr>
<tr>
<td>Messages</td>
<td>Inbound messages that this daemon received on this service (and multicast</td>
</tr>
<tr>
<td>Received</td>
<td>group).</td>
</tr>
<tr>
<td>Bytes Sent</td>
<td>Outbound bytes (summed over all messages) that this daemon sent on this</td>
</tr>
<tr>
<td></td>
<td>service (and multicast group).</td>
</tr>
<tr>
<td>Bytes Received</td>
<td>Inbound bytes (summed over all messages) that this daemon received on this</td>
</tr>
<tr>
<td></td>
<td>service (and multicast group).</td>
</tr>
<tr>
<td>Packets Sent</td>
<td>Outbound packets that this daemon sent on this service (and multicast group).</td>
</tr>
</tbody>
</table>
Table 25  RVDM Daemon Statistics (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packets Received</td>
<td>Inbound packets that this daemon received on this service (and multicast group).</td>
</tr>
<tr>
<td>Packets Retransmitted</td>
<td>Outbound packets that this daemon retransmitted on this service (and multicast group).</td>
</tr>
<tr>
<td>Packets Missed</td>
<td>Inbound packets that this daemon missed on this service (and multicast group).</td>
</tr>
<tr>
<td>Inbound Dataloss (packets)</td>
<td>Inbound packets that this daemon missed on this service (and multicast group), and did not successfully receive retransmission.</td>
</tr>
<tr>
<td>Outbound Dataloss (packets)</td>
<td>Outbound packets for which this daemon received retransmission requests, but could not retransmit the data (because it was outside the reliability window).</td>
</tr>
</tbody>
</table>
Incidents Tab

The Incidents tab displays incidents reported by managed daemons. An *incident* is any asynchronous error that a managed daemon reports to RVDM—except an error related to RVDM pushing updated information to the managed daemon (this type of error causes a synchronous exception).

Figure 74 shows an example with one incident. Table 26 describes the elements of the table.

![RVDM Incidents Tab](image)

**Table 26  RVDM Incidents Interface**

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date &amp; Time</td>
<td>The timestamp indicates when the incident occurred.</td>
</tr>
<tr>
<td>Host Name</td>
<td>Name of the managed daemon’s host computer.</td>
</tr>
<tr>
<td>Description</td>
<td>This string describes the incident.</td>
</tr>
</tbody>
</table>

RVDM does not persistently record incidents, it merely displays them on this centralized tab for convenience. The permanent record of an incident is in the log file of the managed daemon that originally reports the incident.
Options Tab

The Options tab displays the current values for general parameters of the RVDM server, and lets you configure new values.

Figure 75 on page 256 shows an example configuration. Table 27 on page 256 describes the graphical elements of the interface.

To modify configuration, type new values in the appropriate fields, and click the Apply button.
### RVDM Options Tab

**Name**
- Name: **Daemon Manager 1**

**Control Channel**
- **Service**: 7777
- **Network**: 239.77.77.77
- **Daemon**: 7500

**Fault Tolerance**
- **Weight**: 20

**Storage**
- **Data Directory**: /rv/build/tmp/L

**Users**

<table>
<thead>
<tr>
<th>User</th>
<th>Description</th>
</tr>
</thead>
</table>
| User 1 | Username: user1  
Current Password:   
New Password (first entry):  
New Password (second entry): |
| User 2 | Username: user2  
Current Password:   
New Password (first entry):  
New Password (second entry): |

---

**Table 27  RVDM Options Interface (Sheet 1 of 3)**

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the RVDM server process (for identification only).</td>
</tr>
</tbody>
</table>

---

TIBCO Rendezvous Administration
Control Channel

RVDM server communicates with managed daemons on this control channel. This control channel must be distinct from any transport affected by the subject maps and port map. You must not use this channel to carry application data messages.

Furthermore, you must explicitly configure each managed daemon to use this control channel for RVDM control messages. (See also -rvdm on page 47.)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>RVDM server creates a control transport on this service, which it uses to communicate with managed daemons. Managed daemons must use the same service. Specify a port number for RVDM control messages. It is an error if the control channel service port appears in the port map.</td>
</tr>
<tr>
<td>Network</td>
<td>On computers with more than one network interface, this parameter instructs RVDM server to use a particular network interface for control communications. Specify a CIDR address that denotes the same physical network that applications use for data messages. When omitted, the default value is the host computer’s primary network interface. Multicast addressing is permitted; see Constructing the Network Parameter on page 23. However, we strongly discourage specifying asymmetric multicast groups for the control channel. The added complexity of asymmetry would not add any benefit, since managed daemons send only point-to-point messages to the RVDM server,</td>
</tr>
<tr>
<td>Daemon</td>
<td>RVDM listens for control channel connections from daemons on this port.</td>
</tr>
</tbody>
</table>

Fault Tolerance

Weight

Set the weight of this RVDM instance. Weight specifies relative precedence among fault-tolerant instances (with the same control channel). An instance with greater weight takes precedence over an instance with lesser weight. The default value is 10.
Storage

Data Directory  RVDM server stores its configuration data file (RVDM.data.xml) in this file-system directory.

For fault-tolerant operation, ensure that this directory is a shared location, accessible by all members of the fault tolerance group.

See also Appendix B, Store Files, on page 419.

Users

RVDM lets you define two user accounts (User 1 and User 2). A valid login with either user account permits editing RVDM configuration values.

Use these fields to change the name or password of these accounts. To leave user account information unchanged, omit these fields.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>Set the user’s name (string).</td>
</tr>
<tr>
<td>Current Password</td>
<td>You must supply the current password in order to change user information.</td>
</tr>
<tr>
<td>New Password</td>
<td>Enter the new password twice.</td>
</tr>
</tbody>
</table>

Command Buttons

Apply  To apply new configuration values, click the **Apply** button.
**RVDM JMX MBeans**

In addition to the browser administration interface, you can access RVDM by leveraging standard JMX tools.

**Overview**

The package `com.tibco.tibrv.rvdm.jmx` contains three MBean interfaces to RVDM:

- **DaemonManagerMBean** interfaces with an RVDM instance. You can use it to access the parameters in the Options tab (see Options Tab on page 255).

- **PortMapMBean** interfaces with the port map feature of an RVDM instance. You can use it to access the parameters in the Port Map tab (see Port Map Tab on page 240).

- **ServicesMBean** interfaces with the managed services feature of an RVDM instance. You can use it to access the parameters in the Subject Maps tab (see Subject Maps Tab on page 243).

You can write Java programs that use the methods of these interfaces to configure RVDM parameters. For complete documentation (HTML only) of these methods, see RVDM JMX MBean API Reference Pages.

You can use JConsole, or any other JMX-compliant console, to access the attributes and operations of these interfaces; see Using JConsole with RVDM on page 259.

**JMX Ports**

The RVDM JMX agent uses these ports:

- **RMI Port** The RMI registry within RVDM listens on this port for RMI registry requests. The default value is 1099; to override this default, see -rmi-port on page 235.

- **MBean Port** The MBean server in RVDM uses this port to receive connection requests from JMX consoles. The default value is 9589; to override this default, see -mbean-port on page 235.

**Using JConsole with RVDM**

To access RVDM using JConsole, do these steps.
1. When starting RVDM, it outputs the JMX service URL in its start banner and log file; as in this example:

   2009-02-05 10:56:30 rvdm: JMX Service URL:

   Record this URL.

2. Run JConsole (included with the JRE).

3. Paste the RVDM JMX service URL into JConsole.

4. Supply valid user credentials (see User Login on page 237).

5. Click the Connect button.
Relay agents support certified delivery in situations where persistent correspondents connect only intermittently to the network.

For example, many enterprises use laptop computers, which operate independently for extended periods and reconnect to the network when it is convenient. As another example, consider a persistent correspondent that runs as an ephemeral process; at regular intervals a UNIX `cron` job starts the process, which sends several certified messages and then exits. Relay agents support certified message delivery in situations like these.

### Topics

- `rvrad`, page 262

### See Also

Certified Message Delivery on page 139 in *TIBCO Rendezvous Concepts*

Relay Agent on page 169 in *TIBCO Rendezvous Concepts*
rvrad

Command

Syntax
rvrad  -name  name
       -store  filename
       [-service  service]
       [-network  network]
       [-daemon  daemon]

Purpose
The command rvrad starts a Rendezvous relay agent process. The relay agent is a process that stores certified messages (and associated protocol messages) for program processes that connect to the network intermittently. For an overview, see Relay Agent on page 169 in TIBCO Rendezvous Concepts.

Remarks
System administrators must start relay agent (rvrad) processes explicitly, and keep it running at all times. That is, the relay agent must be running whenever disconnected program processes reconnect to the network, and must remain running continuously to collect inbound messages on behalf of their client programs.

A relay agent process requires resources in proportion to the number of certified delivery client programs that it serves.

From the perspective of the Rendezvous daemon (rvd), the relay agent process (rvrad) behaves exactly like any other Rendezvous program. The relay agent’s -service, -network and -daemon parameters are completely analogous to the corresponding parameters of the transport creation call; rvrad uses these values to create a transport that it uses to communicate both with certified delivery client programs and with other relay agents.

When relay agents or their client programs operate across routing daemons, see Forward RVCM Administrative Messages across Network Boundaries on page 404.

On UNIX computers, the relay agent runs as a background process.

The relay agent does not use a browser administration interface.
## Command Line Parameters

(Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-name name</td>
<td>Bind this reusable name to the relay agent process. CM transports designate and locate their relay agents by name. Relay agent names must be unique. It is illegal to run two or more relay agents with the same name simultaneously. It is illegal for a relay agent to have the same name as a CM correspondent. We strongly discourage using the empty string as a relay agent name. The name must conform to the syntax rules for reusable names. For details, see Reusable Names on page 166 in TIBCO Rendezvous Concepts.</td>
</tr>
<tr>
<td>-store filename</td>
<td>Use this file to store relay agent state. The argument must represent a valid file name. Actual locations corresponding to relative file names conform to operating system conventions. The relay agent synchronizes the file to disk on a periodic schedule. See also Appendix B, Store Files, on page 419.</td>
</tr>
<tr>
<td>-service service</td>
<td>The relay agent creates a transport on this service, which it uses to communicate both with its clients and with other relay agents. Clients that communicate with this relay agent must use the same service and network. You can specify the service in several ways. For details, see Service Selection on page 20. NULL specifies the default rendezvous service.</td>
</tr>
<tr>
<td>-network network</td>
<td>On computers with more than one network interface, the -network parameter instructs the Rendezvous daemon to use a particular network for communications involving this transport. The relay agent creates a transport on this network, which it uses to communicate both with its certified delivery client programs and with other relay agents. Clients that communicate with this relay agent must use the same service and network. You can specify the network in several ways. For details, see Constructing the Network Parameter on page 23. NULL specifies the primary network interface for the host computer.</td>
</tr>
</tbody>
</table>
The `-daemon daemon` parameter instructs `rvrad` about how and where to find `rvd` and establish communication. The value of the `rvrad -daemon` parameter must match the `rvd -listen` parameter.

For details, see Daemon Client Socket—Establishing Connections on page 28.

You can specify a daemon on a remote computer. For details, see Remote Daemon on page 29. However, `rvrad` cannot start a remote `rvd` automatically; it must be already running on the remote computer.

If `-daemon` is not present, `rvrad` finds the local daemon on TCP socket 7500.

### See Also

- Forward RVCM Administrative Messages across Network Boundaries, page 404.
- Relay Agent on page 169 in TIBCO Rendezvous Concepts.
Chapter 9  **Rendezvous Agent (rva)**

The Rendezvous agent (rva) is a background process that supports Rendezvous communications for Java applets. For security reasons, remote untrusted applets cannot connect directly to the Rendezvous daemon (rvd); instead, they connect to rva, which in turn connects to rvd.

The Rendezvous agent protects the Rendezvous daemon from inappropriate requests. It acts as security barrier, protecting the Rendezvous network against interference.

Remote Java applets cannot start an rva process; instead, they must connect to an agent process that is already running. Network and system administrators must ensure that an appropriate agent process is running at the web server host computer, so applets can connect back to it.

**Topics**

- rva, page 266
- Web Site Considerations, page 274
rva

Command

Syntax

```
rva -store filename
[-http [ip_address:]http_port]
[-https [ip_address:]https_port]
[-http-only]
[-https-only]
[-no-http]
[-idle]
[-logfile log_filename]
[-foreground]
```

Purpose

The command `rva` starts the Rendezvous agent process. The Rendezvous agent is the gateway to the Rendezvous network for remote Java applets.

Remarks

Usually, administrators must start the Rendezvous agent (`rva`) process explicitly. A Java applet cannot start `rva`; an applet must connect to an agent that is already running.

From the perspective of the Rendezvous daemon (`rvd`), the agent process (`rva`) behaves exactly like any other Rendezvous program. The agent’s `Service`, `Network` and `Daemon` parameters are completely analogous to the corresponding parameters of the transport creation call (because `rva` uses these values to create a transport). In turn, the transport creation call uses these parameters to find or start an appropriate daemon and connect to it.

Licenses

Each `rva` process instance reads the license ticket file (`tibrv.tkt`) when it starts. To put new license tickets into effect, stop and restart `rva`.

If `rvd` is not running, then `rva` starts it automatically, and the new `rvd` process instance requires a valid `rvd` license (in addition to the `rva` license).

Subject Gating

For fine-grained control over all the information flowing in or out of your networks, limit communication by subject name.

The `Import` and `Export` parameters let system administrators restrict the flow of messages through `rva` based on subject names.

Point-to-point messages (`_INBOX.`) always pass through `rva`. They are not restricted by subject gating parameters.
For information about restricting with wildcard subjects, see Subject Filtering with Wildcards on page 85.

In release 6 (and later), the default behavior for rva subject gating is to restrict all subjects unless permitted by subject gating parameters.

In earlier releases, when gating parameters were not specified, the default behavior was to permit all subjects for import and export.

rva supports approximately 4000 clients through TCP connections, or approximately 1500 clients through HTTP connections. These limits are not additive.

Lower file descriptor limits can further reduce the maximum number of rva clients (see File Descriptor Limits on page 9).

Command Line Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-store filename</td>
<td>This file contains the parameters that configure rva.</td>
</tr>
<tr>
<td></td>
<td>rva reads this file when the process starts, and writes this file each time</td>
</tr>
<tr>
<td></td>
<td>you change the configuration using the browser administration interface.</td>
</tr>
<tr>
<td></td>
<td>See also Appendix B, Store Files, on page 419.</td>
</tr>
</tbody>
</table>
The browser administration interface accepts connections on this HTTP or HTTPS port. Permit administration access only through the network interface specified by this IP address.

To limit access to a browser on the rva host computer, specify 127.0.0.1 (the local host address).

When the IP address is absent, the daemon accepts connections through any network interface on the specified HTTP or HTTPS port.

If the explicitly specified HTTP or HTTPS port is already occupied, the program exits.

When the -http parameter is entirely absent, the default behavior is to accept connections from any computer on HTTP port 7680; If this default port is unavailable, the operating system assigns an ephemeral port number.

When the -https parameter is entirely absent, the default behavior is to accept secure connections from any computer on an ephemeral HTTPS port.

In all cases, the program prints (in its start banner and log file) the actual HTTP and HTTPS ports where it accepts browser administration interface connections.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-http ip_address:port</td>
<td>The browser administration interface accepts connections on this HTTP or HTTPS port. Permit administration access only through the network interface specified by this IP address.</td>
</tr>
<tr>
<td>-http port</td>
<td>To limit access to a browser on the rva host computer, specify 127.0.0.1 (the local host address).</td>
</tr>
<tr>
<td>-https ip_address:port</td>
<td>When the IP address is absent, the daemon accepts connections through any network interface on the specified HTTP or HTTPS port.</td>
</tr>
<tr>
<td>-https port</td>
<td>If the explicitly specified HTTP or HTTPS port is already occupied, the program exits.</td>
</tr>
<tr>
<td>-http-only</td>
<td>Disable HTTPS (secure) connections, leaving only an HTTP (non-secure) connection.</td>
</tr>
<tr>
<td>-https-only</td>
<td>Disable HTTP (non-secure) connections, leaving only an HTTPS (secure) connection.</td>
</tr>
<tr>
<td>-no-http</td>
<td>Disable all HTTP and HTTPS connections, overriding -http and -https.</td>
</tr>
<tr>
<td>-idle</td>
<td>When present, start rva in its idle state.</td>
</tr>
<tr>
<td></td>
<td>When absent, start rva in its running state—serving Java clients.</td>
</tr>
<tr>
<td></td>
<td>You can toggle the state at any time using the browser administration interface.</td>
</tr>
</tbody>
</table>
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-logfile log_filename</td>
<td>Send log output to this file. When absent, the default is stderr. Log file rotation is not available for rva.</td>
</tr>
<tr>
<td>-foreground</td>
<td>Available only on UNIX platforms. If present, rva runs as a foreground process. If not present, rva runs as a background process.</td>
</tr>
</tbody>
</table>

### Browser Administration Interface

(State 1 of 5)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>Toggle between idle and running. When running, rva accepts connections from Java client programs, and permits network communication access on behalf of those client programs. When idle, rva disconnects all Java client programs, and bars any new connections.</td>
</tr>
</tbody>
</table>

#### Network Connections

<table>
<thead>
<tr>
<th>Listen Port</th>
<th>The Rendezvous agent creates a TCP socket to establish communication between itself and its Java clients. This parameter specifies the TCP port where the agent listens for client connect requests. This port number corresponds to the port parameter of the Java <em>TibrvRvaTransport()</em> constructor (they must specify the same TCP socket number). The default TCP port is 7600—corresponding to the default port that <em>TibrvRvaTransport()</em> uses.</th>
</tr>
</thead>
</table>
When `rva` communicates on behalf of its Java clients, it communicates on this UDP or PGM service. As a result, the Java clients can communicate only with other programs that create transports in this service group.

The Rendezvous agent uses this parameter to connect to the Rendezvous daemon. This parameter is analogous to the `service` parameter of the transport creation call.

Each `rva` process can communicate on only one service. To communicate over more than one service, you must start additional `rva` processes.

You can specify the service in several ways. For details, see Service Selection on page 20.

On computers with more than one network interface, this parameter instructs the Rendezvous daemon to use a particular network for all communications involving this transport.

The Rendezvous agent uses this parameter to connect to the Rendezvous daemon. This parameter is analogous to the `network` parameter of the transport creation call.

Each `rva` process can send outbound broadcast messages to only one network. To communicate over more than one network, you must start additional `rva` processes.

You can specify the network in several ways. For details, see Constructing the Network Parameter on page 23.

This parameter instructs `rva` to find `rvd` and establish communication on a specific TCP port. The value of this `rva daemon` parameter must match the `rva listen` parameter.

For details, see Daemon Client Socket—Establishing Connections on page 28.

You can specify a daemon on a remote computer. For details, see Remote Daemon on page 29. However, `rva` cannot start a remote daemon automatically—you must start it manually on the remote computer.

With the default value, `rva` finds the local daemon on TCP port 7500.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>When <code>rva</code> communicates on behalf of its Java clients, it communicates on this UDP or PGM service. As a result, the Java clients can communicate only with other programs that create transports in this service group. The Rendezvous agent uses this parameter to connect to the Rendezvous daemon. This parameter is analogous to the <code>service</code> parameter of the transport creation call. Each <code>rva</code> process can communicate on only one service. To communicate over more than one service, you must start additional <code>rva</code> processes. You can specify the service in several ways. For details, see Service Selection on page 20.</td>
</tr>
<tr>
<td>Network</td>
<td>On computers with more than one network interface, this parameter instructs the Rendezvous daemon to use a particular network for all communications involving this transport. The Rendezvous agent uses this parameter to connect to the Rendezvous daemon. This parameter is analogous to the <code>network</code> parameter of the transport creation call. Each <code>rva</code> process can send outbound broadcast messages to only one network. To communicate over more than one network, you must start additional <code>rva</code> processes. You can specify the network in several ways. For details, see Constructing the Network Parameter on page 23.</td>
</tr>
<tr>
<td>Daemon</td>
<td>This parameter instructs <code>rva</code> to find <code>rvd</code> and establish communication on a specific TCP port. The value of this <code>rva daemon</code> parameter must match the <code>rva listen</code> parameter. For details, see Daemon Client Socket—Establishing Connections on page 28. You can specify a daemon on a remote computer. For details, see Remote Daemon on page 29. However, <code>rva</code> cannot start a remote daemon automatically—you must start it manually on the remote computer. With the default value, <code>rva</code> finds the local daemon on TCP port 7500.</td>
</tr>
</tbody>
</table>
Subject Gating

**Imported Subjects**

rva imports messages from clients (applets) to the network. Only messages with subject names that match these subject names are eligible for import. You can remove a subject at any time.

**Exported Subjects**

rva exports messages from the network to clients (applets). Only messages with subject names that match these subject names are eligible for export. You can remove a subject at any time.

HTTP Tunnel

See also HTTP Tunnel on page 277.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>When enabled, rva accepts HTTP connections from client transports. To enable HTTP tunneling, check this box.</td>
</tr>
<tr>
<td>Port</td>
<td>rva accepts HTTP connections from client transports on this port. Zero is a special value, which instructs rva to use its Listen Port for this purpose.</td>
</tr>
<tr>
<td>Client Timeout</td>
<td>In some situations, rva closes HTTP connections to flush data to clients through intervening proxy servers. Each client transport subsequently reestablishes its connection after a delay (specified in the client program). Meanwhile, rva maintains state information for the client. If this timeout expires before the client reconnects, rva can discard the state information. Ensure that this parameter (in seconds) is much greater than the reconnect delay parameter of every client transport.</td>
</tr>
<tr>
<td>Ping Interval</td>
<td>Intervening proxy servers might automatically close client connections to rva that appear inactive. To circumvent this feature, rva sends ping messages to clients when a connection has been idle for this interval (in seconds). To disable this feature, supply zero.</td>
</tr>
</tbody>
</table>
Max Client Queue | rva limits the length of the data queue for each client (data is outbound from rva, inbound to the client). When a queue exceeds this number of messages, rva discards subsequent messages as they arrive.
To disable this feature, supply zero.

Max Queue Size | rva limits the size of the data queue for each client (data is outbound from rva, inbound to the client). When a queue exceeds this size (in kilobytes), rva discards subsequent messages as they arrive.
To disable this feature, supply zero.

Active Flush | This parameter limits data latency (in seconds) caused by buffering in intervening proxy servers. If rva has directed data to a client, and this time limit has elapsed since rva last flushed the data, then rva closes the connection to force proxy servers to deliver the data to the client. This action occurs even when rva has more data queued for the client (that is, the queue is active).
If you are certain that no proxy servers intervene between rva and the client, set this parameter to zero.

Inactive Flush | This parameter limits data latency (in seconds) caused by buffering in intervening proxy servers. If rva has directed data to a client, and does not have more data queued for the client (that is, the queue is inactive), and this time limit has elapsed since rva last directed data to the client, then rva closes the connection to force proxy servers to deliver the data to the client.
If you are certain that no proxy servers intervene between rva and the client, set this parameter to zero.

Request Flush | This parameter limits data latency (in seconds) for reply messages. In some application domains, the client sends a request message, and receives only one small reply message. Setting this parameter smaller than inactiveFlush can expedite delivery of such reply messages to the client.
Max Proxy Buffer

This hint estimates the size (in kilobytes) of the largest buffer in intervening proxy servers. Before triggering an active flush, rva checks the amount of data it directed to the client since the last flush; if that amount is greater than this value, the proxy server has probably flushed its buffer automatically, so rva does not close the connection to flush the data.

If you are certain that no proxy servers intervene between rva and the client, set this parameter to zero.

### Certificates

These parameters control the use of certificates as administrator credentials.

### XML Configuration

View the current configuration as an XML document, and reconfigure the component by submitting an edited XML document.

### Security

These parameters control access to the configuration pages of the rva browser administration interface.
Web Site Considerations

The Rendezvous agent (rva) is a key component of any web site that distributes Rendezvous applets. A full treatment of web site administration is beyond the scope of this book; this section discusses issues specific to Rendezvous applets and rva.

Home Computer and Port

Java applets connect back to an rva process on the web server host computer by calling the constructor `TibrvRvaTransport()` with hostname and port arguments. An rva process must be running on the web server host computer, listening for client connections on a specific TCP port. Applet calls to `TibrvRvaTransport()` must use a TCP port that matches the `-listen` parameter of rva (both use 7600 as a default port number).

Administrators must inform applet developers of the TCP port where rva is listening for applet connections, so applet code can supply it as an argument to `TibrvRvaTransport()`. If you reconfigure rva to listen on a different port, applets cannot connect until updated to the new port number.

Open a Path through the Firewall

Administrators must configure the firewall so that TCP connection requests can propagate from the applet back to rva. Applets can only connect if this path is open.

For example, Figure 76 on page 275 illustrates a double firewall configuration. Applet connection requests travel on path A (TCP port 7600), and rva requests connection to the remote rvd using path B (TCP port 7577). Both these paths must be open.
Figure 76  Typical rva Web Site with Double Firewall
Isolate External from Internal

In the context of a web site with Rendezvous applets, it is especially important to isolate separate pathways for various programs, to protect internal Rendezvous application programs from external applets.

Techniques include using multicast addressing and restricting flow by subject. For more information, see these sections:

- Network Details, page 17
- Subject Gating, page 266

For example, near the bottom of Figure 76 on page 275, rvd links external applets with internal UDP or PGM service 7901. All internal Rendezvous programs that use service 7901 can exchange messages with applets across the Internet. In contrast, a private demonstration applet uses service 7854—it is effectively isolated from the external applets, as well as from internal programs using service 7901.
HTTP Tunnel

Java programmers can specify HTTP tunneling for communication between a client transport and rva. Administrators can configure rva parameters that affect this type of communication.

This section describes the general operation of this feature, and the rva parameters that affect it.

Overview
For an overview of HTTP tunneling, see these sections:
- HTTP Tunneling on page 15 in TIBCO Rendezvous Java Reference
- TibrvRvaTransport on page 223 in TIBCO Rendezvous Java Reference

Operation
When rva enables HTTP tunneling, it accepts client connection requests on an HTTP port. Client transports connect to rva at that port.

rva maintains state information for each client, as well as a queue of data for the client (data is outbound from rva, inbound to the client).

For efficiency, rva accumulates several messages in a queue before directing them to a client.

HTTP proxy servers might intervene between rva and the client. Proxy servers are usually transparent to both clients and rva. However, some intervening proxy servers might buffer data at a low level, causing delays in data delivery to clients. To limit data latency, rva can force proxy servers to flush their buffers by closing the client connection; several parameters affect this flushing action. Client transports automatically reestablish the connection, after a time delay specified in the client program. rva maintains client state (for a limited time) while waiting for the client to reconnect.

Parameters
For a list of rva parameters that affect this feature, see HTTP Tunnel on page 271.
Chapter 10  Current Value Cache

In many distributed applications new processes can join the system at any time. Often these new processes need access to the current information state of the system in order to function properly. In many cases a straightforward cache program can fill that need.

The Rendezvous distribution includes a utility program called rvcache, which caches the data from messages sent to each subject name. Whenever a Rendezvous program begins listening to a subject name, it can query rvcache to send it the current data for that subject.

The data cached for a subject can be either the most recent whole message on that subject, or a composite set containing the most recent value of each field sent on that subject.

Although rvcache resembles a simple database program in some respects, it differs in an important way. Namely, updates are implicit; that is, rvcache monitors message activity and automatically caches the data. Application components query for data by subject.

We recommend that administrators arrange for correct operation of rvcache. This chapter describes administrative considerations; for a command summary, see rvcache on page 291.

Topics

- Operation, page 280
- Resource Requirements, page 282
- Avoid Duplicates, page 283
- Ensure Continuous Service, page 284
- Crossing Network Boundaries, page 285
- Fault Tolerance, page 286
- rvcache, page 291
Operation

Although many distributed system components may depend upon rvcache, its caching operation remains transparent to them.

Other application programs do not send update messages specifically to rvcache. Instead, rvcache listens for a set of subjects, silently receiving messages and caching the most recent data on each subject as it arrives, as in Figure 77.

Figure 77  Transparent Caching by rvcache

However, other application programs do send query requests to rvcache. Figure 78 on page 281 illustrates this phase of rvcache operation:

1. rvcache listens to the subject _SNAP.> for query messages.

2. A program submits a query for the cached value of foo.bar by sending an empty message to the subject _SNAP. foo.bar (more generally, build the query subject on the template _SNAP. cached_subject).

3. rvcache receives the query, and extracts the cache subject from the query subject name. It sends the cached value for that subject to the reply subject of the query message.
Figure 78  Query and Response with rvcache
Resource Requirements

Load

For fastest response, run rvcache on a computer with a light processing load.

Storage

The exact amount of required storage space varies with three factors—the storage mode, the number of subjects cached, and the size of the stored message values.

- In standard operation, rvcache keeps a table of cached subjects in memory, while it keeps message data for those subjects in a store file on disk. The computer running rvcache must have sufficient storage of each type.

- In memory-only mode, rvcache keeps both in memory (both the table of cached subjects and the message data). The computer running rvcache must have sufficient memory. (For background information, see Memory-Only Mode on page 289.)

Distributed Caches

In some cases, you may find it expedient to distribute the resource requirements and the processing load among several computers. To achieve this goal, you can run several process instances of rvcache on separate computers. However, it is important that various process instances of rvcache cache disjoint sets of subjects (see Avoid Duplicates on page 283).
Avoid Duplicates

Listening programs rarely profit from receiving duplicate copies of the current data. To prevent duplicates, consider one of these strategies:

- Run exactly one `rvcache` service.
- Ensure that `rvcache` services store disjoint subject sets.
  
  When two or more `rvcache` services store the same subjects, then duplicate messages can result. If the subject sets do not overlap, then duplication cannot occur.

- Segregate `rvcache` services by listening on different UDP or PGM services.

You can use different UDP or PGM services to isolate groups of program processes so that members of each group receive Rendezvous messages exclusively from other members of the same group. In such cases, configure a separate instance of `rvcache` for each group by setting its service parameter to match the UDP or PGM service used by group members. For more detail about this parameter, see Service Selection on page 20.
Ensure Continuous Service

Interruptions in rvcache service can result in two undesirable consequences:

- Programs that query during the interruption do not receive the cached data.
- The cache does not record data from messages sent during the interruption. Gaps in the cache persist after the interruption.

To minimize these effects, we recommend that system administrators consider these strategies:

- Avoid interruptions whenever possible.
  
  For example, shift rvcache (along with its disk file) to an alternate host computer before scheduled downtime. Since rvcache is independent of the computer on which it runs, switching hosts can be an effective remedy.

- Reduce the length of unavoidable interruptions.
  
  Monitor the health of the rvcache process and its host computer. If any problem prevents smooth operation, promptly correct the situation or shift rvcache to an alternate host computer.

- Run rvcache as a fault-tolerant service.
  
  For details, see Fault Tolerance on page 286.
Crossing Network Boundaries

When a network boundary separates rvcache from its client programs, and a routing daemon (rvrd) connects them across that boundary, you must configure rvrd to ensure correct operation of rvcache.

Cached Subjects

The routing daemons (on both sides of the neighbor link) must permit all the cached subjects to flow from all senders to all rvcache processes.

Query Subjects

The rvrd configuration for exchanging query subjects depends on the distribution of rvcache and its query clients.

- If each network runs one local cache process, with all the caches synchronized (so they all contain the same data), then it is crucial that only one rvcache process receive each query. The routing daemons must not import or export _SNAP.> (the query subject).
- If only one network runs a cache process, and programs on other networks query it across the network boundary, then the routing daemons must forward _SNAP.> (the query subject) into the rvcache network. That is, rvrd must import these query names into the rvcache network; rvrd must export these query names from each query client network.
Fault Tolerance

Multiple process instances of rvcache can cooperate for fault-tolerant service. Fault tolerance protects rvcache service against hardware failures, process termination and segmentation of the local network.

In this configuration, two or more rvcache processes run on separate computers—usually on separate network segments. All cooperating processes listen for the same set of subjects, and store the current values of those subjects. Only one process (called the primary active process) actively sends the current values to new listeners. The remaining processes (called inactive backup processes) are inactive—unless they detect that the primary active process has failed. If the primary fails, one of the backup process activates in its place, restoring service automatically.

These chapters describe fault tolerance concepts and parameters in detail:

- Developing Fault-Tolerant Programs on page 227 in TIBCO Rendezvous Concepts.

For administrative details, see Fault Tolerance on page 407.

Usage

To run rvcache as a fault-tolerant service, start two or more rvcache processes. It is essential that all processes use identical parameters—with only one exception:

- The -store parameter specifies a file for persistent storage of the cache and configuration parameters. Member processes must not share this file. Each member must keep its own distinct cache file (we recommend storing it on a local disk).

Duplicating the Cache State

To duplicate the cache state, copy the cache file (so each process starts with an identical copy). Avoid file inconsistencies that can arise when copying the file while rvcache is running.
Replace and Merge

rvcache can store message data in two ways. For each subject, it can either replace all previously stored data with the contents of each new message, or it can merge information from the fields of the message into the stored data, overwriting only those fields specified in the new message.

Select one of these two storage methods each time you add a subject.

Furthermore, rvcache can merge data in either of two ways. The command line parameter -merge selects either shallow merge or deep merge as the behavior of the cache, consistent for all merged subjects. Consider a new message that contains nested messages as field values.

- Shallow merge replaces the old field value with the new nested message as an indivisible unit, without special treatment for the individual fields of the nested message. That is, merging occurs at only one level of nesting; level-one fields replace level-one fields.

- Deep merge inspects the fields of nested messages, and recursively merges the fields of a new nested message into fields of a stored nested message. Merging continues recursively to any depth of nesting.

Example

Table 28 on page 288 presents an example, contrasting the different behavior of replacement, shallow merge and deep merge.
Data stored in `rvcache` never expires. It remains in the cache until superseded or augmented by data from a new message on the same subject.
Memory-Only Mode

Store File  In standard operation, rvcache uses the store file to record parameter configuration and for persistent cache storage. Parameter configuration includes the list of cached subjects. Persistent cache storage includes the cached values of those subjects. The required command line parameter -store specifies the pathname of the store file.

Memory-Only  In some high-volume situations, writing cached values to the store file can result in an I/O bottleneck. If rvcache I/O presents a problem, you can disable persistent cache storage. The command line parameter -memory-only starts rvcache in memory-only mode (otherwise, rvcache runs in store mode).

Memory-only mode changes the operation of rvcache as follows:

- When rvcache starts, it does not read initial values from the store file. Consequently, rvcache starts with an empty cache for all subjects.
- rvcache keeps cached values only in process memory; it does not write values to the store file.
- However, rvcache still reads its configuration from the store file at start time, and writes configuration changes to the store file.

Memory-only mode increases the operating speed of rvcache at the cost of data persistence. If the process exits, all cached values are lost.

In memory-only mode, you can expect rvcache to consume more process memory than in store mode. Since it cannot retrieve infrequently accessed values from the store file, rvcache must keep all values in process memory. You must ensure that adequate memory is available. In addition, we recommend running the 64-bit version of rvcache where possible (since it can address twice as much process memory as the 32-bit version).

Switching from store mode to memory-only mode does not automatically erase all initial values from the store file. Even though rvcache does not read them, they remain in the store file (which as a result might be larger because of these unnecessary values).

Since a larger store file can slow rvcache initialization, we recommend starting with an empty store file when you begin using memory-only mode.

You can use the dumpXML command of tibrvcfg to dump the configuration of rvcache as an XML document. Then use the matchXML command to create an empty store file with the required configuration. For details, see Command Line Tool—tibrvcfg on page 347 in TIBCO Rendezvous Configuration Tools.
After switching from store mode to memory-only mode and back again, you cannot rely on cached values in the store file. Some subject configuration changes can erase the value of a subject as a side effect. Before switching modes, we recommend saving a backup copy of the store file.

Alternatively, you can use the `dumpXML` command of `tibrvcfg` to save a backup copy of configuration data before you switch modes. See Command Line Tool—`tibrvcfg` on page 347 in TIBCO Rendezvous Configuration Tools.

When configuring large or complex sets of subjects, the configuration API is more convenient than the browser administration interface; see Chapter 8 on page 271 in TIBCO Rendezvous Configuration Tools.
# rvcache

## Command

### Syntax

```plaintext
rvcache [-store filename]
[-http [ip_address:]http_port]
[-https [ip_address:]https_port]
[-http-only]
[-https-only]
[-no-http]
[-idle]
[-sync interval]
[-merge shallow | deep]
[-memory-only]
```

### Purpose

The program `rvcache` stores data from recent messages, indexed by subject name, and automatically sends the cached data to new listeners.

### Remarks

Given a set of one or more subject names, `rvcache` listens for messages addressed to those subjects. Each time it receives such a message, it stores the message’s data content.

When a client program queries for a cached subject, `rvcache` sends a reply message with the current cached value.

### Browser Administration Interface

To administer or configure `rvcache`, view `http://rvcache_host:http_port` with a web browser. When the program starts, it prints the actual HTTP administration port.

### State

`rvcache` can run in either of two states—running or idle.

When `running`, `rvcache` listens to subjects, caches message values, and responds to queries.

When `idle`, `rvcache` does not operate; however, the browser administration interface is available for configuring parameters.

### Initial Subject Configuration

The first time you run `rvcache`, you must configure its subjects and change its state to running. After that, `rvcache` reads the subject list from its file.

### Storage

`rvcache` stores the data in process memory and in a disk file. The command line parameter `-store` specifies the name of the disk file; if the file exists when `rvcache` starts, then `rvcache` reads the file to initialize its configuration parameters and to populate its cache in process memory.

The command line parameter `-sync` specifies the interval at which to synchronize the file-based store with process-based store.
# Command Line Parameters

(Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-store filename</code></td>
<td>Use <code>filename</code> to record parameter configuration and for persistent cache storage. For best performance, use a local file system (remote file servers can cause delays and synchronization difficulties). For more information, see Storage on page 291. See also Appendix B, Store Files, on page 419.</td>
</tr>
<tr>
<td><code>-http ip_address:http_port</code></td>
<td>The browser administration interface accepts connections on this HTTP or HTTPS port. Permit administration access only through the network interface specified by this IP address.</td>
</tr>
<tr>
<td><code>-https ip_address:https_port</code></td>
<td>To limit access to a browser on the rvcache host computer, specify 127.0.0.1 (the local host address). When the IP address is absent, the daemon accepts connections through any network interface on the specified HTTP or HTTPS port. If the explicitly specified HTTP port is already occupied, the program exits. If the explicitly specified HTTPS port is already occupied, the program selects an ephemeral port. When the <code>-http</code> parameter is entirely absent, the default behavior is to accept connections from any computer on HTTP port 7581; If this default port is unavailable, the operating system assigns an ephemeral port number. When the <code>-https</code> parameter is entirely absent, the default behavior is to accept secure connections from any computer on an ephemeral HTTPS port. In all cases, the program prints (in its start banner and log file) the actual HTTP and HTTPS ports where it accepts browser administration interface connections.</td>
</tr>
<tr>
<td><code>-http-only</code></td>
<td>Disable HTTPS (secure) connections, leaving only an HTTP (non-secure) connection.</td>
</tr>
<tr>
<td><code>-https-only</code></td>
<td>Disable HTTP (non-secure) connections, leaving only an HTTPS (secure) connection.</td>
</tr>
</tbody>
</table>
**Parameter** | **Description**
--- | ---
-no-http | Disable all HTTP and HTTPS connections, overriding -http and -https.
-idle | When present, start rvcache in its idle state.
When absent, start rvcache in its running state—caching values and responding to queries. However, if subjects are not configured, rvcache begins in its idle state.
You can toggle the state at any time using the browser administration interface.
-merge shallow | deep | For subjects that cache by merging the new value into the stored value, two types of merge behavior are available. When present, this parameter selects that behavior for all merged subjects. For complete details, see Replace and Merge on page 287.
When absent, the default behavior is shallow merging.
-memory-only | When present, rvcache keeps cached values in process memory only; it does not write values to the store file. Nor does it read initial values from the store file at start time. For details, see Memory-Only Mode on page 289.
When absent, rvcache writes cached values to the store file.
-sync interval | The synchronization behavior of operating systems varies. You can use this parameter to balance message processing speed against disk synchronization guarantees.
When absent, rvcache relies on the operating system for all synchronization. rvcache opens the store file in synchronous write mode—that is, the operating system writes every message to the file system before the write call returns.
When present, rvcache explicitly synchronizes data to the file system at this interval. rvcache opens the store file in asynchronous write mode—that is, write calls return independently of disk operations, and the operating system completes disk writes on its own schedule. Explicitly synchronizing at a fixed interval limits exposure to loss by enforcing an upper bound—though the operating system might also synchronize between intervals (reducing exposure even further).
Browser Administration Interface

(Sheet 1 of 3)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>information</td>
<td>This page displays general information about the rvcache process.</td>
</tr>
<tr>
<td>change state</td>
<td>Toggling the state of the rvcache process.</td>
</tr>
<tr>
<td></td>
<td>When running, rvcache listens to its cache subjects, caches values, and responds to queries.</td>
</tr>
<tr>
<td></td>
<td>When idle, rvcache does not operate; however, the browser administration interface is available for configuring parameters.</td>
</tr>
<tr>
<td></td>
<td>The program does not store this parameter.</td>
</tr>
<tr>
<td>certificates</td>
<td>This page lets you configure the certificates that rvcache uses to identify itself to web browsers. For more information, see the analogous section for secure daemons, Certificates on page 204.</td>
</tr>
<tr>
<td>security</td>
<td>These parameters control access to the configuration pages of the rvcache browser administration interface.</td>
</tr>
<tr>
<td>connection</td>
<td>rvcache uses these parameters to create its network transport object.</td>
</tr>
<tr>
<td></td>
<td>For general explanations, see Chapter 3, Network Details, on page 17.</td>
</tr>
<tr>
<td></td>
<td>See Service Selection on page 20.</td>
</tr>
<tr>
<td></td>
<td>See Network Selection on page 23.</td>
</tr>
<tr>
<td></td>
<td>See Daemon Client Socket—Establishing Connections on page 28.</td>
</tr>
<tr>
<td>fault tolerance</td>
<td>Enable or disable fault-tolerant operation.</td>
</tr>
<tr>
<td></td>
<td>The remaining parameters in this group apply only when fault tolerance is enabled.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Service</td>
<td>Use this UDP or PGM service for fault tolerance control messages between rvcache member processes. The default value is <code>rendezvous-ft</code>; if the operating system cannot interpret that service name, then the secondary default is UDP or PGM port 7504.</td>
</tr>
<tr>
<td>Network</td>
<td>Use this network for fault tolerance control messages between rvcache member processes. The default value is the computer’s primary network interface.</td>
</tr>
<tr>
<td>Group</td>
<td>Use this string as the name of the rvcache fault tolerance group. Processes with the same group name cooperate to provide fault-tolerant service. The default value is RVCACHE.</td>
</tr>
<tr>
<td>Weight</td>
<td>Set the weight of this rvcache process. Weight specifies relative precedence among fault-tolerant processes. A process with greater weight takes precedence over a process with lesser weight. The default value is 10.</td>
</tr>
<tr>
<td>Heartbeat</td>
<td>Use this floating point value (in seconds) as the fault tolerance heartbeat interval. Members of a fault tolerance group send status reports at this interval. We recommend that this value be slightly less than one third of the activation interval. The default value is 3 seconds.</td>
</tr>
<tr>
<td>Activation</td>
<td>Use this floating point value (in seconds) as the fault tolerance activation interval. This value represents the longest interruption in service before the partner process activates. It must be the same for all members of a fault tolerance group. The default value is 10 seconds.</td>
</tr>
</tbody>
</table>

**subjects**

| Subjects | To see information about a specific subject, click that subject in the current subject list. You can add new subjects or remove current subjects at any time. For more information, see Replace and Merge on page 287. |
### XML Configure

View the current configuration as an XML document, and reconfigure the component by submitting an edited XML document.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML Configure</td>
<td>View the current configuration as an XML document, and reconfigure the component by submitting an edited XML document.</td>
</tr>
</tbody>
</table>
Chapter 11  Performance Assessment (rvperf)

Performance assessment software can help you gauge and improve Rendezvous network performance, plan hardware purchases and software deployment, and test network configurations.

Topics

- Overview, page 298
- Principles of Operation, page 299
- Single Mode and Automatic Mode, page 300
- Automatic Mode—Binary Search, page 301
- Dataloss Advisory, page 302
- Multicast, Broadcast, Point-to-Point and Direct, page 303
- Before You Test, page 304
- rvperfm, page 306
- rvperms, page 311
- Interpreting the Report, page 314
- Usage and Examples, page 318
- Hardware Capabilities, page 319
- Wide Area Networks, page 321
- Certified Message Delivery, page 322
- Very Large Messages, page 323
- Sufficiency and Effects, page 324
- Locating Performance Obstacles, page 325
### Overview

Rendezvous performance assessment software is a tool for evaluators, reviewers, and network administrators. It measures the potential performance of Rendezvous software in an actual network situation, and outputs a detailed report.

This performance assessment software helps you compare various equipment options and network configurations, using the performance of Rendezvous software as a gauge.

Remember that speed benchmarks are relevant only in the context of a specific network with specific computers. Networks can differ widely in their performance. Adding or removing a problematic computer can dramatically alter performance.

Rendezvous benchmark data is not a guarantee of application performance. It demonstrates the potential maximum performance that your network can achieve. Although it can model common application behaviors regarding message sending and receiving, it cannot exactly mimic the actual performance of your Rendezvous applications. The performance assessment tool stresses message transport capabilities, but does not engage in other common application behaviors, such as calculations or managing a graphics display.

The performance assessment tool can establish an upper bound on application message transport performance, and help gauge some of the secondary effects of network usage patterns, but it cannot prove that an application as a whole will operate properly.

### Components

Performance assessment software consists of two executable programs:

- `rvperf m` (master) sends messages, gathers performance data, and outputs the report.
  
  See `rvperf m` on page 306.

- `rvperfs` (slave) subscribes to messages from `rvperf m`, and sends back data about its own speed and effectiveness.
  
  See `rvperfs` on page 311.

You can run `rvperf m` alone, or with any number of `rvperfs` processes in the network.
Principles of Operation

In our experience, Rendezvous distributed applications achieve optimal network performance when senders transmit messages in short batches, pausing briefly between batches. This observation is the foundation for the performance assessment tool.

Performance assessment software measures network performance by sending runs of messages, and compiling statistics. You can experiment by varying parameters such as message size, run length, batch size, pause interval—which affect the network data rate.

Each message contains $s$ bytes of payload data (plus message headers and packet overhead).

$rvperfm$ sends one or more sequences (or runs) of $m$ messages to the network.

Instead of sending a continuous stream of messages, $rvperfm$ groups them into batches of $b$ messages, pausing for an interval of $i$ seconds between the end of one batch and the start of the next batch.

While $rvperfm$ is sending messages, zero or more process instances of $rvperfs$ are listening for those messages and compiling statistics. At the end of each run, the $rvperfs$ processes report back to $rvperfm$, which outputs the statistics.

Listeners

$rvperfm$ can send messages to the network whether or not any $rvperfs$ processes are listening to receive those messages.

- When $rvperfs$ listeners are present, the performance assessment tool measures their capacity to receive messages as part of overall network performance.
- In the absence of $rvperfs$ listeners, the performance assessment tool measures the network performance of the sending computer only.
Two modes characterize the operation of `rvperf`:

- **In single mode**, `rvperf` sends a single run of messages, governed by its command parameters. At the end of the run, it outputs a report and exits.

  You can use single mode as a modeling tool to answer questions about network behavior under sustained load conditions. For example:
  - What happens to network performance when an application sends a batch of ten thousand messages without pausing?
  - Which computers in my network can send messages the fastest? Which can receive fastest?
  - How does introducing a router affect network throughput under normal network load conditions? How do peak loads affect network throughput?

- **In automatic mode**, `rvperf` sends several runs of messages, modifying the parameters for each run until it finds the batch size and interval parameters that yield maximum sustainable network throughput. At the end of each run, it outputs a report. Then it adjusts the parameters and starts the next run.

  The overall effect is that `rvperf` tunes its send rate to match the maximum receive rate of the slowest `rvperf` process. The last report before the process exits displays the parameters that yield the maximum throughput. (In the absence of `rvperf` processes, `rvperf` determines the maximum send rate that the network can support. Note that the parameter tuning algorithm is the same, only the significance of the result differs.)

  You can use the results of testing in automatic mode to tune applications for maximum performance in a specific network configuration.
Automatic Mode—Binary Search

In automatic mode, `rvperfm` uses a binary search algorithm to adjust its batch size and interval parameters between runs. These rules control `rvperfm` as it empirically determines the maximum throughput:

- For the first run, `rvperfm` determines the batch size and interval from command parameters or default values.

- If `rvperfm` loses data, it aborts the run immediately, and adjusts parameters to decrease the send rate for the next run.

- If even one of the `rvperf`s processes lost data during the run, then the send rate exceeds the maximum network throughput. In this case, `rvperfm` aborts the run, records the upper bound on maximum throughput, and adjusts the parameters to decrease the send rate for the next run.

- If all active `rvperf`s processes keep pace without lagging behind, and receive all the messages without losing data—then the send rate is lower than the maximum throughput. In this case, `rvperfm` records the lower bound on maximum throughput, and adjusts the parameters to increase the send rate for the next run.

- After a finite number of runs, the upper and lower bounds converge at the maximum throughput. When `rvperfm` exits, the report of the last run indicates the batch size and interval parameters that yield the maximum network throughput.

If no `rvperf`s processes are active, the parameters of the last run yield the maximum send rate for `rvperf` on its host computer (with the prevailing network conditions).
Dataloss Advisory

rvperf\textsubscript{m} and rvperf\textsubscript{s} both subscribe to DATALOSS advisories. At the end of each complete run, both programs report the number of advisories they received during the run.

If rvperf\textsubscript{m} receives a DATALOSS advisory, it aborts the run immediately. (This paragraph applies only to automatic mode; if rvperf\textsubscript{s} receives a DATALOSS advisory while rvperf\textsubscript{m} is in single mode, the run does not abort.)

If rvperf\textsubscript{s} receives a DATALOSS advisory, while receiving messages from rvperf\textsubscript{m} in automatic mode, then rvperf\textsubscript{s} informs rvperf\textsubscript{m} in its response to the next auto window poll, and rvperf\textsubscript{m} aborts the run. (This paragraph applies only to automatic mode; if rvperf\textsubscript{s} receives a DATALOSS advisory while rvperf\textsubscript{m} is in single mode, the run does not abort.)

See also, DATALOSS on page 269, TIBCO Rendezvous Concepts.
Multicast, Broadcast, Point-to-Point and Direct

Rendezvous can transport messages among application programs using several mechanisms. The performance assessment tool can model the performance of an application sending messages in any of these ways:

<table>
<thead>
<tr>
<th>Transport</th>
<th>Modelling</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multicast</td>
<td>Specify multicast addressing in the -network parameter.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Omit the -inbox parameter.</td>
<td></td>
</tr>
<tr>
<td>Broadcast</td>
<td>Do not specify multicast addressing in the -network parameter. TRDP only</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Omit the -inbox parameter.</td>
<td></td>
</tr>
<tr>
<td>Point-to-Point</td>
<td>Include the -inbox parameter.</td>
<td></td>
</tr>
<tr>
<td>Direct Point-to-Point</td>
<td>Specify a two part -service parameter to enable direct communication (bypassing rvd).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Include the -inbox parameter.</td>
<td></td>
</tr>
</tbody>
</table>

Performance Characteristics

The various transport mechanisms can display dramatically different performance profiles, which group into two broad classes.

<table>
<thead>
<tr>
<th>Transport</th>
<th>Description and Performance Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multicast;</td>
<td>Multicast and broadcast messages use an unmetered protocol with negative acknowledgment. A sender transmits packets as fast as possible. Receivers are responsible for requesting retransmission of missed packets. Many applications can gain efficiency by dividing very large multicast or broadcast messages into smaller pieces, sending them in batches, and pausing between batches to avoid overloading slow receivers (that is, receivers running on relatively slow computers).</td>
</tr>
<tr>
<td>Broadcast</td>
<td></td>
</tr>
<tr>
<td>Point-to-Point;</td>
<td>Point-to-point messages use a metered protocol with positive acknowledgment. A sender requires positive acknowledgment from the receiver before it transmits additional point-to-point packets. As a result, point-to-point packets (from a single sender) rarely arrive faster than a receiver can process them. Applications generally do not gain efficiency by dividing very large point-to-point messages into smaller pieces (since the protocol itself already meters delivery).</td>
</tr>
</tbody>
</table>
Before You Test

Before running Rendezvous performance assessment software, read this section carefully.

Test in an Insulated Environment

We strongly recommend that you run all performance tests in a network environment that is insulated from other Rendezvous applications and other network traffic.

Consider these two important benefits of an insulated environment:

- Insulation prevents performance assessment message traffic from disrupting deployed applications.
- Insulation ensures that traffic from other applications does not skew performance measurements.

Testing in a physically isolated network yields the most accurate measurements. When physical isolation is impractical, you can still obtain valid measurements by insulating tests within unused multicast addresses. However, in this arrangement, \texttt{rvperf} traffic can still affect the performance of other deployed network applications.

\textbf{rvd Reliability}

To ensure accurate and efficient testing, it is critical that you first disable the \texttt{rvd} reliable message storage feature of \texttt{rvd}.

- To disable reliability for non-managed daemons, manually start \texttt{rvd} (or \texttt{rvrd}) with the command line parameter \texttt{-reliability 0}.

This zero value instructs \texttt{rvd not} to retain outbound messages in case they are needed for retransmission. For more information, see Reliability and Message Retention Time on page 34.

\texttt{rvperf} attempts to determine the carrying capacity of the network. To do so, it tests whether the network and a set of receivers can absorb a run of messages without missing any packets. The reliable delivery feature of \texttt{rvd} defeats this purpose; it compensates for transient network problems by retaining and
retransmitting packets. This behavior is often beneficial in a production environment, but in a performance testing situation it is counterproductive—by compensating for network problems, it delays detection of those problems. This delay falsifies the results of performance testing, and unnecessarily prolongs the testing period.
rvperfm

Command

Syntax

rvperfm [-service service ]
[-network network ]
[-daemon daemon ]
[-subject subject ]
[-inbox]
[-auto]
[-non-vectored]
[-terse]
[-messages m ]
[-size size ]
[-interval interval ]
[-batch batch_size ]
[-cm]
[-cm-name name ]
[-cm-ledger filename ]
[-cm-sync]
[-h]

Purpose

rvperfm coordinates the tasks of measuring network performance. It sends messages to the network, and reports statistics to stdout.

Remarks

In single mode (without the flag -auto), rvperfm sends one run of messages, and then exits.

In automatic mode (with the flag -auto), rvperfm sends several runs of messages, adjusting the batch size and interval parameters to empirically determine the combination that yields maximum network throughput. After it finds the optimal settings, it exits; the parameters and report of the final run reflect optimal network performance. For details, see Automatic Mode—Binary Search on page 301.

Outline

Each run consists of these steps:

1. Dynamically discover the available rvperfs processes; output a list of participating instances. In the discovery step, rvperfm polls for listeners, and waits 3 seconds for ready signals from rvperfs processes; then it continues to the next step.

2. Send the run of messages.

3. Output statistics that measure the performance of the sender.

4. Output statistics that measure the performance of each receiver (if any).

5. Output a summary of error advisories pertaining to the sender.
Collision When two instances of rvperfm (simultaneously) attempt to use the same subject, service and network, at least one of them detects the collision and exits immediately.

Simultaneous instances that differ in subject or service (or both) do not constitute a collision. Such processes can coexist.

(Sheet 1 of 4)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| -service service | service is the service name or UDP or PGM port number that defines the service group.  
See Service Selection on page 20.  
If you do not specify the -service parameter, the default value is 7599. |
| -network network | network narrows the service group by selecting a local network by network name or IP network number (when the host computer has multiple network interfaces). It can also specify multicast addresses.  
See Network Selection on page 23.  
If you do not specify the -network parameter, the default value is the multicast address ";225.9.9.9". On operating systems that do not support multicast addressing, you must supply a valid broadcast network address. |
| -daemon daemon | The -daemon parameter instructs the program about how and where to find rvd and establish communication.  
See Daemon Client Socket—Establishing Connections on page 28.  
You can specify a daemon on a remote computer. For details, see Remote Daemon on page 29. However, the program cannot start a remote daemon automatically—you must start it manually on the remote computer.  
If you do not specify the -daemon parameter, the program finds the local daemon on TCP socket 7500. |
rvperf sends messages to this subject name. If you specify neither \(-\text{subject}\) nor \(-\text{inbox}\), then the program uses \_perf\ as a prefix to construct broadcast subjects.

rvperf probes the network to discover available instances of rvperfs. The first instance to respond becomes the sole receiver—rvperf sends point-to-point messages only to an inbox in that process instance.

(Since rvperf uses broadcast subjects for the initial discovery phase, it is not a contradiction to specify both \(-\text{inbox}\) and \(-\text{subject}\) parameters. When both parameters are present, \(-\text{inbox}\) determines the sending behavior.)

When present, rvperf operates in automatic mode, sending several runs of messages to automatically determine the optimal batch size and interval parameters for the network.

When absent, rvperf operates in single mode, sending only one run of messages.

See also, Automatic Mode—Binary Search on page 301.

When present, rvperf sends individual messages.

When absent, rvperf sends each batch of messages using a single vector call.

When present, suppress most reporting and simplify the final report.

The terse final report contains one line per receiver (rvperfs). Each line is a comma-separated list of the following values:

- send message rate, received message rate,
- send byte rate, received byte rate,
- elapsed send time, elapsed receiver time,
- receiver SlowConsumer, receiver DataLoss,
- messages sent, send size, batch interval, batch size, reply name

For descriptions of these values, see Interpreting the Report on page 314.

rvperf sends \(m\) messages per run.

If not present, the default is 10000 messages per run.
rvperfm sends messages with size bytes of payload data.

Use this size to model application data rates. This size does not include message header data nor packet overhead, so computing the network byte transfer rate from this size results in an slight underestimate of the actual throughput.

If not present, the default is 256 payload bytes in each message.

rvperfm sends messages in batches, waiting for pause seconds between the end of one batch and the start of the next batch.

When absent, the default pause is zero seconds.

In single mode, rvperfm sends the run with this interval.

In automatic mode, rvperfm sends the first run with this interval, adjusting the parameters in subsequent runs.

Change of Units: In earlier releases the value of this parameter was interpreted as milliseconds—now it is a floating point value interpreted as seconds.

rvperfm sends messages in batches, with batch_size messages in each batch.

When absent, the default is 128 messages per batch.

To send messages individually, specify 1 as the batch_size.

In single mode, rvperfm sends the run with this batch size.

In automatic mode, rvperfm sends the first run with this batch size, adjusting the parameters in subsequent runs.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-size size</td>
<td>rvperfm sends messages with size bytes of payload data.</td>
</tr>
<tr>
<td></td>
<td>Use this size to model application data rates. This size does not include</td>
</tr>
<tr>
<td></td>
<td>message header data nor packet overhead, so computing the network byte</td>
</tr>
<tr>
<td></td>
<td>transfer rate from this size results in an slight underestimate of the</td>
</tr>
<tr>
<td></td>
<td>actual throughput.</td>
</tr>
<tr>
<td></td>
<td>If not present, the default is 256 payload bytes in each message.</td>
</tr>
<tr>
<td>-interval pause</td>
<td>rvperfm sends messages in batches, waiting for pause seconds between the</td>
</tr>
<tr>
<td></td>
<td>end of one batch and the start of the next batch.</td>
</tr>
<tr>
<td></td>
<td>When absent, the default pause is zero seconds.</td>
</tr>
<tr>
<td></td>
<td>In single mode, rvperfm sends the run with this interval.</td>
</tr>
<tr>
<td></td>
<td>In automatic mode, rvperfm sends the first run with this interval, adjusting</td>
</tr>
<tr>
<td></td>
<td>the parameters in subsequent runs.</td>
</tr>
<tr>
<td>-batch batch_size</td>
<td>rvperfm sends messages in batches, with batch_size messages in each batch.</td>
</tr>
<tr>
<td></td>
<td>When absent, the default is 128 messages per batch.</td>
</tr>
<tr>
<td></td>
<td>To send messages individually, specify 1 as the batch_size.</td>
</tr>
<tr>
<td></td>
<td>In single mode, rvperfm sends the run with this batch size.</td>
</tr>
<tr>
<td></td>
<td>In automatic mode, rvperfm sends the first run with this batch size,</td>
</tr>
<tr>
<td></td>
<td>adjusting the parameters in subsequent runs.</td>
</tr>
</tbody>
</table>
When present, `rvperfm` sends messages with certified delivery features. If `rvperfm` also specifies `-cm`, then the programs establish a certified delivery agreement.

When `-cm` is present, but `-cm-name` is not, `rvperfm` operates with a non-reusable correspondent name.

When present, `rvperfm` specifies this reusable correspondent name when it enables certified delivery.

When absent, the operating system writes ledger file changes to the storage medium asynchronously.

When present, `rvperfm` uses this ledger file. You must also supply `-cm-name`.

When present, then operations that update the ledger file do not return until the changes are written to the storage medium. You must also supply `-cm-ledger` and `-cm-name`.

When present, output a parameter usage list to stdout, and exit immediately.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-cm</code></td>
<td>When present, <code>rvperfm</code> sends messages with certified delivery features. If <code>rvperfm</code> also specifies <code>-cm</code>, then the programs establish a certified delivery agreement.</td>
</tr>
<tr>
<td><code>-cm-name name</code></td>
<td>When present, <code>rvperfm</code> specifies this reusable correspondent name when it enables certified delivery. When <code>-cm</code> is present, but <code>-cm-name</code> is not, <code>rvperfm</code> operates with a non-reusable correspondent name.</td>
</tr>
<tr>
<td><code>-cm-ledger filename</code></td>
<td>When present, <code>rvperfm</code> uses this ledger file. You must also supply <code>-cm-name</code>.</td>
</tr>
<tr>
<td><code>-cm-sync</code></td>
<td>When present, then operations that update the ledger file do not return until the changes are written to the storage medium. You must also supply <code>-cm-ledger</code> and <code>-cm-name</code>. When absent, the operating system writes ledger file changes to the storage medium asynchronously.</td>
</tr>
<tr>
<td><code>-h</code></td>
<td>When present, output a parameter usage list to stdout, and exit immediately.</td>
</tr>
</tbody>
</table>
rvperfs

Command

Syntax

rvperfs [-service service]  
[-network network]  
[-daemon daemon]  
[-subject subject]  
[-non-vectored]  
[-cm]  
[-cm-name name]  
[-cm-ledger filename]  
[-cm-sync]  
[-h]

Purpose

rvperfs listens for messages from rvperfm, gathers and reports statistics to rvperfm at the end of each run.

Remarks

rvperfs operates passively; it sends messages only in response to requests from rvperfm.

You can leave process instances of rvperfs running idle. Each instance of rvperfs can report statistics from several consecutive process instances of rvperfm—as long as only one rvperfm executes at a time. You can relocate the rvperfm process from one host computer to another without restarting the rvperfs processes.

Unlike rvperfm, an rvperfs process never exits by itself. You must explicitly terminate each rvperfs process.

In addition to sending its statistics to rvperfm, rvperfs also prints its report to stdout.

Parameter | Description
---|---
-service service | service is the service name or UDP or PGM port number that defines the service group.

See Service Selection on page 20.

If you do not specify the -service parameter, the default value is 7599.
network narrows the service group by selecting a local network by network name or IP network number (when the host computer has multiple network interfaces). It can also specify multicast addresses.

See Network Selection on page 23.

If you do not specify the `network` parameter, the default value is the multicast address ";225.9.9.9".

The `daemon` parameter instructs the program about how and where to find rvd and establish communication.

See Daemon Client Socket—Establishing Connections on page 28.

You can specify a daemon on a remote computer. For details, see Remote Daemon on page 29. However, the program cannot start a remote daemon automatically—you must start it manually on the remote computer.

If you do not specify the `daemon` parameter, the program finds the local daemon on TCP socket 7500.

rvperfs listens for messages with this subject name.

If this parameter is absent, then rvperfs uses _perf as a prefix to construct broadcast subjects.

(When you specify the `-inbox` flag to rvperfm, you need not specify this rvperfs parameter.)

When present, rvperfs receives messages individually, using an ordinary listener.

When absent, rvperfs receives messages using a vector listener.

When present, rvperfs listens for messages using certified delivery features. If rvperfm also specifies `-cm`, then the programs establish a certified delivery agreement.

When present, rvperfs specifies this reusable correspondent name when it enables certified delivery.

When `-cm` is present, but `-cm-name` is not, rvperfs operates with a non-reusable correspondent name.

When present, rvperfs uses this ledger file. You must also supply `-cm-name`.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-network</code> network</td>
<td>network narrows the service group by selecting a local network by network name or IP network number (when the host computer has multiple network interfaces). It can also specify multicast addresses. See Network Selection on page 23. If you do not specify the <code>network</code> parameter, the default value is the multicast address &quot;;225.9.9.9&quot;.</td>
</tr>
<tr>
<td><code>-daemon</code> daemon</td>
<td>The <code>daemon</code> parameter instructs the program about how and where to find rvd and establish communication. See Daemon Client Socket—Establishing Connections on page 28. You can specify a daemon on a remote computer. For details, see Remote Daemon on page 29. However, the program cannot start a remote daemon automatically—you must start it manually on the remote computer. If you do not specify the <code>daemon</code> parameter, the program finds the local daemon on TCP socket 7500.</td>
</tr>
<tr>
<td><code>-subject</code> subject</td>
<td>rvperfs listens for messages with this subject name. If this parameter is absent, then rvperfs uses _perf as a prefix to construct broadcast subjects. (When you specify the <code>-inbox</code> flag to rvperfm, you need not specify this rvperfs parameter.)</td>
</tr>
<tr>
<td><code>-non-vectored</code></td>
<td>When present, rvperfs receives messages individually, using an ordinary listener. When absent, rvperfs receives messages using a vector listener.</td>
</tr>
<tr>
<td><code>-cm</code></td>
<td>When present, rvperfs listens for messages using certified delivery features. If rvperfm also specifies <code>-cm</code>, then the programs establish a certified delivery agreement.</td>
</tr>
<tr>
<td><code>-cm-name</code> name</td>
<td>When present, rvperfs specifies this reusable correspondent name when it enables certified delivery. When <code>-cm</code> is present, but <code>-cm-name</code> is not, rvperfs operates with a non-reusable correspondent name.</td>
</tr>
<tr>
<td><code>-cm-ledger</code> filename</td>
<td>When present, rvperfs uses this ledger file. You must also supply <code>-cm-name</code>.</td>
</tr>
</tbody>
</table>
When present, operations that update the ledger file do not return until the changes are written to the storage medium. You must also supply `-cm-ledger` and `-cm-name`.

When absent, the operating system writes ledger file changes to the storage medium asynchronously.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-cm-sync</code></td>
<td>When present, output a parameter usage list to <code>stdout</code>, and exit immediately.</td>
</tr>
<tr>
<td><code>-h</code></td>
<td>When present, output a parameter usage list to <code>stdout</code>, and exit immediately.</td>
</tr>
</tbody>
</table>
Interpreting the Report

This section describes the output from `rvperf`. First, `rvperf` outputs a header, version information, and a summary of its configuration parameters.

Next it polls the network to discover existing `rvperfs` processes. Each `rvperfs` process resets itself and signals its readiness to participate in a new run. When `rvperf` receives the ready signals, it prints an identifier for each participating `rvperfs`.

`rvperf` prints a brief string as it begins sending the run of messages, and another when it finishes sending the run. Then it outputs its run report:

1. Statistics that `rvperf` collects while sending the messages.
2. Statistics that each `rvperfs` process collects while receiving messages. Each group of statistics represents the performance of one `rvperfs` process.

`rvperf` Example Report

The annotated `rvperf` transcript in Figure 79 on page 315 shows the result of a run with one `rvperfs` process receiving the messages.
Figure 79  Report from rvperfm

TIB/Rendezvous performance analysis program
Copyright 1997-2001 by TIBCO Software Inc.
All rights reserved.

Version 6.7.5

Configuration parameters
Service: 8000
Network: ;225.9.9.9
Daemon: 8888
Subject: _perf

Number of messages: 102400
Payload bytes per message: 1024
Total payload bytes: 104857600 (100.0Mb)

Batch interval: 0.000000
Batch size: 1

Run #1 beginning...

Batch interval: 0.000000
Batch size: 1

Resetting receivers
Reset acknowledgment received from _INBOX.0A650224.1E0743AA4617B2004CDE8.1

Number of receivers 1

Sending data...

Sending complete
Elapsed time: 9.724587 seconds (0.378101 in flush)
Number of messages: 102400
Size of payload: 1024
Total payload bytes: 104857600
Batch Interval: 0.000000
Batch size: 1

Messages/second: 10530.010
Payload Bytes/second: 10782730.413 (10.3Mb)

Report from receiver _INBOX.0A650224.1E0743AA4617B2004CDE8.1
Elapsed time: 10.205277 seconds
Messages received: 102400 (100.0%)

Messages/second: 10034.025
Payload Bytes/second: 10274841.143 (9.8Mb)

Run complete

rvperfm probes for rvperfs processes, and outputs an identifier for each rvperfs

Send messages
Time to send the messages
Part of time to flush remaining packets to the net

Actual send rate
Receive statistics from rvperfs
All messages arrived

Actual receive rate
rvperfs Example Report

The excerpt in Figure 80 illustrates the output of rvperfs. First, the rvperfs process reports its configuration parameters. Next it outputs its globally unique inbox name, by which you can identify it in rvperfm reports.

Then rvperfs reports a reset message from rvperfm, signaling the start of a run. At the end of the run, rvperfs reports its statistics.

Figure 80 Report from rvperfs

Certified Delivery Agreements

When the performance programs use certified message delivery, rvperfm prints information about registration requests in its output; for example:

rvperfm

Resetting receivers
Reset acknowledgment received from _INBOX.0A65034A.DEA3B3232A8F085470.2
CM registration request received from rvperfs

Number of receivers 1

Similarly, rvperfs prints information about certified delivery registration in its output:
rvperfs  Ready...
Reset received from _INBOX.0A6503F6.11D3B3241239116F0.2
CM registration received from rvperfm
Test message received
Run beginning...

Elapsed Time

Both programs in the performance tool report the total time that elapsed in each complete run. The speed at which the Rendezvous daemon can deliver messages to the network depends on the network itself, the network interface card (and other hardware parameters), and the host operating system. If rvperfm sends at a faster rate than the network can accept, rvperfm retains messages in its outbound queue until the network can accept them.

Sending complete
Elapsed time: 9.737 seconds

In this example, 9.737 seconds elapsed from the time that rvperfm sent the first message of the run, until the time that the daemon transmitted the last message of the run to the network.
Usage and Examples

The Rendezvous performance assessment tool is extremely flexible; it can measure performance in many different configurations of hardware, operating systems, network topologies, and operating conditions. The results can guide hardware purchases, tune for optimal network performance, or validate the suitability of Rendezvous software in specific situations. The remaining sections of this chapter present selected examples.

Remember, it is crucial to gauge performance in actual deployment networks, rather than in a laboratory. Benchmarks produced in one network rarely apply to other networks. Changing the network (by adding or removing computers, routers, or other elements) can change performance dramatically.

Network Stress

On a fast computer Rendezvous software can overwhelm the capacity of the network. Other programs operating during this kind of performance test can display symptoms of network stress.
Hardware Capabilities

In this group of examples, the performance assessment tool measures the speed of specific computers—individually or in a group.

Optimal Sustained Receive Rate

What is the maximum rate at which a specific computer can receive a stream of messages?

To answer this question, run `rvperfs` on the receiving computer. Then run `rvperfm` in automatic mode on another computer; select a message size that reflects the messages that your application will send when deployed.

```bash
receiver> rvperfs

sender> rvperfm -auto -size 1000
```

After `rvperfm` experiments with its parameters, the final run indicates the values that yield the optimal receive rate for the receiving computer under prevailing network conditions. (However, you must validate this measurement; see below.)

A similar test with several receivers determines the optimal rate of the slowest receiver.

Validate against Max Transfer Rate

To validate an optimal receive rate, check that it is strictly less than the maximum transfer rate from `rvperfm` to `rvd` (see below).

- If `rvperfm` on the sending computer has successfully transferred a run of messages at a rate strictly greater than the optimal receive rate, then that receive rate is valid.

- If the measured receive rate is approximately equal to the maximum transfer rate, it might be because some limitation on the sending host is causing an artificially low result for the receive test.

Finding the Max Transfer Rate

To obtain the sender’s maximum transfer rate, run `rvperfm` in automatic mode on the sending computer, without any `rvperfs` processes to receive the messages; use the same message size as in the receive test.

```bash
sender> rvperfm -auto -size 1000
```

After `rvperfm` experiments with its parameters, the final run indicates the values that yield the maximum transfer rate to the daemon. This result is not a useful measure of network performance; its only legitimate use is to validate measurements of receive rates.
Fixed Receive Rate

Can all computers on this network receive 2000-byte messages at a sustained rate of approximately 5 batches per second, with 10 messages per batch?

To answer this question, run `rvperfs` on each of the receiving computers. Then run `rvperfm` in single mode on another computer.

```
receiver1> rvperfs
receiver2> rvperfs
...
receiver42> rvperfs
sender> rvperfm -size 2000 -batch 10 -interval .2
```

The run report indicates whether the receivers keep pace with the sender under prevailing network conditions.
Wide Area Networks

In a wide area network (WAN) the transit time between sites can limit throughput. To keep information flowing smoothly, it is essential to measure the optimal throughput rates for the entire WAN, and limit sending rates to avoid exceeding overall network capacity.

Consider a global network connected using the Rendezvous routing daemon, rvrd. You can use the performance assessment tool for these tasks:

- Measure optimal sustainable throughput rates for the entire WAN.
- Compare actual speed and throughput of available WAN carrier links.
- Compare different neighbor configurations between rvrd components.
- Select rvrd host hardware.
- Demonstrate the effects of exceeding network capacity.
- Discover optimal locations from which to send messages to the rest of the network.
Certified Message Delivery

Certified message delivery introduces additional complexity. The performance assessment tool can help you measure its effects on application performance.

When the \(-cm\) parameter is present, \(rvperf\) sends messages using Rendezvous certified delivery features. Before each run, it clears its ledger (whether file-based or process-based), and sends a test message to provoke \(-cm\) receivers to register for certified delivery. After the registration period, it sends the run of messages.

Number of Certified Receivers

When a certified sender process is operating near maximum capacity (either the capacity of its host computer, or the network capacity), then the number of certified receivers can dramatically affect the timing results.

Throughput of certified messages decreases as the number of registered receivers increases. This decrease is a direct result of confirmation messages flowing back from certified receivers to the sender. You can use the performance tool to measure the network capacity for certified messages with varying numbers of registered receivers.

Ledger

Certified delivery depends on a ledger to track messages and confirmations. Two types of ledger are available; each has a different effect on performance:

- A file-based ledger with asynchronous I/O offers persistence at the cost of disk operations. With asynchronous file I/O, some information could be lost in the event of sudden termination.

- A file-based ledger with synchronous I/O offers greater certainty at the cost of additional speed because the disk write operations block. Synchronous file I/O dramatically reduces the probability of lost information in the event of sudden termination.

You can use the performance tool to compare the effect of these options on certified message throughput.
Very Large Messages

Rendezvous software can transport very large messages; it divides them into small packets, and places them on the network as quickly as the network can accept them. In some situations, this behavior can overwhelm network capacity; applications can achieve higher throughput by dividing large messages into smaller chunks and regulating the rate at which it sends those chunks. You can use the performance tool to evaluate chunk sizes and send rates for optimal throughput.

This example, sends one message consisting of ten million bytes. Rendezvous software automatically divides the message into packets and sends them. However, this burst of packets might exceed network capacity, resulting in poor throughput:

```
sender> rvperfm -size 10000000 -messages 1
```

In this second example, the application divides the ten million bytes into one thousand smaller messages of ten thousand bytes each, and automatically determines the batch size and interval to regulate the flow for optimal throughput:

```
sender> rvperfm -size 10000 -messages 1000 -auto
```

By varying the `-messages` and `-size` parameters, you can determine the optimal message size for your applications in a specific network. Application developers can use this information to regulate sending rates for improved performance.
Sufficiency and Effects

Designers of distributed applications need to assess the effect of a proposed application on the network—long before deployment, and often before any code exists. The performance assessment tool can help answer questions such as these:

- Can Rendezvous software in this network transfer data at the rate projected for this application?

- Increased message traffic affects the operation of network infrastructure and elements such as routers, WAN links, individual computers, and previously deployed network applications (including Rendezvous applications, as well as mounted remote file systems, telnet, and others). What are the secondary effects of deploying an application that sends messages at the projected data rate?

Limits of Performance Assessment

Although the performance assessment tool can measure sufficiency of network transport, and the secondary effects of projected message traffic, its measurements are an idealized abstract. It cannot measure the total effect of a proposed, and as yet unimplemented, application.

Generating data to send in messages, processing inbound messages, displaying data from inbound messages to the user—all of these activities and their affect on the application’s host computer are beyond the scope of the performance assessment tool. For example, this tool can determine that the network can absorb 300 query messages per second, but this figure does not indicate whether a database application can actually process queries and return results at that rate.

The performance assessment tool can establish an upper bound on application message transport performance, and help gauge some of the secondary effects, but it cannot prove an application as a whole will operate properly.
Locating Performance Obstacles

An application that performs more poorly than expected could be sending messages faster than the network can accept them. The performance assessment tool can help in two ways:

- **Use** `rvperfm` in automatic mode to determine the optimal send rate for the network. Then adjust the application to send messages at that rate.
  
  For a specific example of this method, see Very Large Messages on page 323.

- **Set** `rvperfm` parameters to mimic the sending behavior of the application. Then adjust `rvperfm` parameters to improve performance. Finally, adjust the application’s behavior.

  A network protocol monitor (such as `rvtrace`) can help you diagnose performance obstacles using this method. For more information, see Protocol Monitor (`rvtrace`) on page 345.
Chapter 12  Latency Assessment (rvlat)

Latency assessment software can help you gauge message latency in your network.

Topics

- Overview, page 328
- Using rvlat, page 331
- rvlat, page 333
- Output, page 338
Overview

In some application domains, response time is critically important. Several factors affect network latency, including network bandwidth conditions, hardware capabilities, multitasking, and messaging throughput patterns.

The latency assessment tool, rvlat, can help you understand the latency characteristics of your network. rvlat measures latency statistics and produces reports.

Message latency (as measured by rvlat) is the round-trip time between the client call that sends a request message to a server, and the message callback when the client receives a response from the server.

rvlat is an executable program that runs in two modes—as a requesting client or as a responding server. To use rvlat, you must run one instance of each mode.

Principles of Operation

The basic operation of rvlat is similar to Rendezvous performance assessment software, even though it measures a different property of the network. An rvlat client process sends a run of messages to a server; the server replies to those messages; the client receives the replies and measures latency statistics. You can vary parameters such as message size, run length, batch size, pause interval—which affect the network latency. (For descriptions of these quantities, see Chapter 11, Performance Assessment (rvperf), on page 297.)

rvlat can measure multicast or broadcast latency. It does not measure point-to-point latency.

You can use rvlat to measure latency while communicating through Rendezvous local daemons, remote daemons, and TIBCO Messaging Appliance™ P-7500.

Measuring Technique

rvlat measures the round-trip time for a request-reply message pair:

1. The client timestamps its outbound request message.
2. The server responds to a request by immediately returning the same message to the client.
3. The client timestamps the inbound reply, and measures the difference between the two timestamps to obtain the round-trip time.
Many applications that require low latency send messages in only one direction. However, clock synchronization between two computers is not precise enough to accurately measure one-way travel time. To avoid this difficulty, rvlat measures round-trip time using a single clock.

Nonetheless, measuring round-trip time can also distort the results in several ways. For example, doubling the number of messages doubles the network bandwidth usage, and the effect on Rendezvous can be different for one-way versus two-way communication. The two computers might have different throughput capabilities. Timestamps, data computations and data output at the client add overhead. Turn-around time at the server adds a small overhead.

Serial & Batch Modes

rvlat can measure round-trip time in two modes. To get a full understanding of your network’s latency characteristics, we recommend measuring latency in both modes.

- In serial mode (the default behavior), the client sends one request at a time. When the reply arrives, the client records the round-trip time. Only after processing the reply does the client send the next request message.
  
  Serial mode can help you understand patterns of latency variation over time.

- In batch mode, the client sends a batch of messages as rapidly as possible, then pauses for a specified interval while gathering replies and measuring their round-trip times. When the interval elapses, the client sends another batch. The -batch parameter specifies batch mode, and requires a size argument (that is, the number of requests per batch).

  Batch mode simulates high-throughput network conditions, which can produce different latency characteristics than low-throughput conditions.

  When you specify batch mode, you may optionally specify vectored mode as well—which sends each batch as a vector of messages, using a single send call.

Random Sampling

In serial mode and with small batches, the distorting factors are minimal. However, when the batch size is large, the distortion can be more noticeable.

You can reduce these distorting factors large batch sizes by reducing the number of round-trip messages. The -sample parameter instructs the server to respond to a subset of the request messages that it receives, using a probability-based sampling method.

You can use sampling to create high-throughput network conditions, while dramatically reducing the volume of data collected.
For example, consider a run of 1,000,000 requests with a message payload of 100 bytes each. Sending 1,000,000 requests but only 5000 replies (a 0.5% sample) represents a network bandwidth load of approximately 100,500,000 bytes. The 0.5% sample distorts the results less than a 100% sample, and collects far less data, yet the client still has enough data points to measure latency under high-throughput conditions.

**Caveat** However, this sampling technique can also miss important patterns in the data. For example, if latency spikes occur with regular periodicity, random sampling might miss some or all of those spikes.
Using rvlat

rvlat runs as a client-server pair. You cannot run several clients with one server, nor vice versa.

We recommend that you write a script to run rvlat, redirecting its output streams to capture files (see Output Streams on page 338).

Start the server before starting the client.

Subjects

rvlat uses subjects that match RVLAT.> to carry its requests, replies, control signals and data.

- If you use subject mapping (with RVDM), then you must ensure that these subjects are appropriately mapped.
- If the network interposes a routing daemon between the client and server, you must configure it to forward these subjects in both directions.

Test Conditions

Measure latency in a controlled environment. That is, ensure that no other applications are consuming CPU or I/O resources on the two test computers, and that no other applications are consuming network bandwidth. For example:

- Terminate all other user applications.
- Terminate scheduled jobs (UNIX cron), and anything else that uses CPU cycles.
- Use a network analyzer to detect other bandwidth usage, and eliminate it.
- To reduce variability that can distort latency measurements, you can set the reliability interval to zero.

Test Strategies

Run a series of trials lasting between 30 seconds and a few minutes.

Set message size to approximate expected data payloads for your applications.

Use serial mode to establish baseline round-trip statistics under uniform low-throughput conditions, and to understand network behavior. Then use batch mode to understand how latency degrades under high-throughput conditions. Vary batch size to simulate the actual data rates of your applications.
Data Strategies

Capture summaries from several trials in a spreadsheet, so you can easily analyze and manipulate the results.

Large datasets are unwieldy. When it is crucial to capture raw data (with datapoints), use the -sample parameter to reduce dataset size.

Use the -spikes parameter to capture only the outliers.

Interpreting Data

Use plotting and statistical tools to graph latency data, visualize patterns, and correlate those patterns with network behavior.
**rvlat**

**Command**

**Syntax**

```bash
rvlat { cli | srv }
[-service service ]
[-network network ]
[-daemon daemon ]
[-sample response_rate]
[-messages m ]
[-time t ]
[-size size ]
[-batch batch_size]
[-interval interval ]
[-inbox]
[-vectored]
[-terse]
[-datapoints]
[-spikes threshold ]
[-w filename]
[-h]
```

**Purpose**

`rvlat` measures network latency. The client sends request messages to the server, and reports statistics to `stdout` and `stderr`. The server responds to client requests immediately.

**Outline**

Each run consists of these steps:

1. Send a run of messages.
2. Output statistics that measure the performance of the *sender*.
3. Output statistics that measure the performance of each *receiver* (if any).
4. Output a summary of error advisories pertaining to the *sender*.

---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mode Parameters</strong></td>
<td></td>
</tr>
<tr>
<td>cli</td>
<td>Run a client instance.</td>
</tr>
<tr>
<td>srv</td>
<td>Run a server instance.</td>
</tr>
</tbody>
</table>
Parameters that Apply to Both Client and Server

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-service service</code></td>
<td><code>service</code> is the service name or UDP or PGM port number that defines the service group.</td>
</tr>
<tr>
<td></td>
<td>See Service Selection on page 20.</td>
</tr>
<tr>
<td></td>
<td>When absent, the default value is 12486.</td>
</tr>
<tr>
<td><code>-network network</code></td>
<td><code>network</code> narrows the service group by selecting a local network by network name or IP network number (when the host computer has multiple network interfaces). It can also specify multicast addresses.</td>
</tr>
<tr>
<td></td>
<td>See Network Selection on page 23.</td>
</tr>
<tr>
<td></td>
<td>When absent, the default value is the multicast address &quot;;224.1.1.5&quot;. On operating systems that do not support multicast addressing, you must supply a valid broadcast network address.</td>
</tr>
<tr>
<td><code>-daemon daemon</code></td>
<td>The <code>-daemon</code> parameter instructs the program about how and where to find <code>rvd</code> and establish communication.</td>
</tr>
<tr>
<td></td>
<td>See Daemon Client Socket—Establishing Connections on page 28.</td>
</tr>
<tr>
<td></td>
<td>You can specify a daemon on a remote computer. For details, see Remote Daemon on page 29.</td>
</tr>
<tr>
<td></td>
<td>However, the program cannot start a remote daemon automatically—you must start it manually on the remote computer. Note that using a remote daemon could increase latency.</td>
</tr>
<tr>
<td></td>
<td>When testing latency through TIBCO Messaging Appliance P-7500, supply the <code>hostname:port</code> of the P-7500.</td>
</tr>
<tr>
<td></td>
<td>When absent, the program finds the local daemon on TCP socket 50000.</td>
</tr>
</tbody>
</table>
When `rvlat` sends messages in batches, it can send either individual messages (with `Send`) or message vectors (with `Sendv`). The `Send` and `Sendv` calls have different latency characteristics. You can use this parameter to test either call.

When present, `rvlat` sends each batch of messages as a single vector (using the `Sendv` API call).

When absent, `rvlat` sends each message using a separate `Send` call.

### Receiving

When `rvlat` receives messages, it can dispatch them as individual messages, or as message vectors. You can use this parameter to test either paradigm.

When present, `rvlat` receives messages with a vector listener.

When absent, `rvlat` receives messages individually with an ordinary listener.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `-vectored` | **Sending**
When `rvlat` sends messages in batches, it can send either individual messages (with `Send`) or message vectors (with `Sendv`). The `Send` and `Sendv` calls have different latency characteristics. You can use this parameter to test either call.

When present, `rvlat` sends each batch of messages as a single vector (using the `Sendv` API call).

When absent, `rvlat` sends each message using a separate `Send` call. **Receiving**

When `rvlat` receives messages, it can dispatch them as individual messages, or as message vectors. You can use this parameter to test either paradigm.

When present, `rvlat` receives messages with a vector listener.

When absent, `rvlat` receives messages individually with an ordinary listener. |
| `-h` | When present, output a parameter usage list to `stdout`, and exit immediately. |
| `-help` | |

### Server-Only Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `-sample response_rate` | For background information, see [Measuring Technique on page 328](#). When present, the server uses a random number generator to select a subset of requests to which it responds, while ignoring all the rest. The value of `response_rate` specifies the probability of a response (as a percentage) for each message.

When absent, the server responds to 100% of the request messages it receives.

We do not recommend using `-sample` when measuring in serial mode. |

### Client-Only Parameters that Control Measuring

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `-messages m` | When present, `rvlat` sends a run of `m` messages.

When absent, the default is a run of 10,000 messages. |
When present, `rvlat` sends a run of messages that continues for $t$ seconds. When `-messages` is also present, `-time` overrides `-messages`. When absent, the default behavior sends a specific number of messages (rather than running for a specific time).

- **-time $t$**
  - `rvlat` sends request messages with $size$ bytes of payload data.
  - Use this size to model application data rates. This size does not include message header data nor packet overhead.
  - When absent, the default is 0 payload bytes in each message.

- **-batch batch_size**
  - When present, `rvlat` sends messages in batches, with `batch_size` messages in each batch.
  - When absent, `rvlat` sends messages serially, sending each request immediately after receiving a response to the previous request.

- **-interval pause**
  - When `rvlat` sends messages in batches, it waits for `pause` seconds between the end of one batch and the start of the next batch.
  - When absent, the default pause is 1 second.
  - In serial mode (that is, when `-batch` is absent), `rvlat` ignores this parameter.

- **-inbox**
  - When present, the client sends request messages to an inbox on the server (using point-to-point protocols). The server responds to an inbox on the client.
  - When absent, the client sends request messages to a multicast subject (using either multicast or broadcast protocols, as specified in the `-network` parameter). The server responds to a multicast subject.
### Client-Only Parameters that Control Output

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| **-terse** | The client can output two types of reports:  
- A terse report with limited information, for import to spreadsheets  
- A verbose, human-readable report  
The two types of report include the same information, and both are in CSV format.  
When this option is present, `rvlat` outputs only a terse report to `stdout`.  
When absent, `rvlat` outputs two reports—a terse report to `stdout`, and a human-readable verbose report to `stderr`. |
| **-datapoints** | When present, `rvlat` outputs each round-trip data point (in milliseconds) to `stdout`.  
Caution: This option can generate an unwieldy volume of data. |
| **-spikes threshold** | When present, `rvlat` outputs data points greater than `threshold` to `stderr`.  
Each data point includes the round-trip time and the sequence number of the request message (within the run). |
| **-w filename** | When present, `rvlat` takes output data that would otherwise go to `stdout`, and instead writes it to the specified file for later analysis.  
If the file is not empty, `rvlat` appends the data at the end of the file.  
Output to `stderr` is not affected; see [Output Streams on page 338](#). |
Output

rvlat produces several kinds of output, including summary, raw data points, and outliers; human-readable and spreadsheet-ready; as well as error messages (which indicate problems with the command line parameters).

Output Streams

rvlat directs its output to two streams:

- stderr for human-readable output
  This category includes the human-readable summary, the spike data points (when requested), and error messages (if any).
- stdout for spreadsheet output
  This category includes the terse spreadsheet summary, and the raw data points (when requested).

Summaries

After a run, rvlat produces a summary of its data. The summary includes 13 comma-separated values (see Table 29). The summary is available in two forms:

- Human-readable summary, including units and abbreviated labels for each value
- Terse spreadsheet summary, including only the numeric values (with neither units nor labels)

Table 29  rvlat Summary Output

<table>
<thead>
<tr>
<th>#</th>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>max</td>
<td>Maximum latency (in milliseconds)</td>
</tr>
<tr>
<td>2</td>
<td>min</td>
<td>Minimum latency (in milliseconds)</td>
</tr>
<tr>
<td>3</td>
<td>mean</td>
<td>Average latency (in milliseconds)</td>
</tr>
<tr>
<td>4</td>
<td>stddev</td>
<td>Standard deviation (in milliseconds)</td>
</tr>
<tr>
<td>5</td>
<td>&gt;1ms</td>
<td>Spikes—number of messages with latency greater than 1 millisecond.</td>
</tr>
</tbody>
</table>

(Notice that this 1 millisecond summary threshold is independent of any threshold you might specify using the -spikes parameter.)
With the `-datapoints` option, *rvlat* outputs a column of raw data points, one value per line to stdout. Each value is the time (in milliseconds) for one round-trip message exchange. The values are purely numeric, suitable for spreadsheets.

After all the data points, *rvlat* outputs the terse summary line to stdout.

### Spikes

With the `-spikes` option, *rvlat* outputs high-latency data points (those with round-trip time greater than a threshold). This output to stderr is in two columns. Each row is one data point, which consists of two comma-separated values—the sequence number of the request message and its round-trip time (in milliseconds).

After all the spike data, *rvlat* outputs the human-readable summary line to stderr (unless `-terse` suppresses it).

### Discarded Data Points

If the round-trip time is faster than the client computer’s clock resolution (measuring time of day), the client can record the round-trip time as zero. Zero data points would invalidate the statistical measurements, so the client discards them.
The occurrence of any discards indicates that measurements are less precise than they could be. A computer with finer-grained time of day clock could produce more precise measurements.

If any discards occur, the client outputs a warning message. If this warning appears, we recommend selecting a different computer for the client.
Chapter 13  Measuring Tools for IPM

This chapter describes executable tools that measure performance and latency while communicating using IPM.

Topics

- IPM Tools, page 342
- Command Line Parameters, page 343
IPM Tools

The product distribution includes executable tools that measure performance and latency while communicating using IPM; see Table 30.

Each IPM tool corresponds to a regular tool that communicates through a Rendezvous daemon. For a description of the regular tool, see TIBCO Rendezvous Administration.

Table 30  Measuring Tools

<table>
<thead>
<tr>
<th>IPM Tool</th>
<th>Corresponding Regular Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>rvperfm_ipm</td>
<td>rvperfm</td>
</tr>
<tr>
<td>rvperfs_ipm</td>
<td>rvperfs</td>
</tr>
<tr>
<td>rvlat_ipm</td>
<td>rvlat</td>
</tr>
</tbody>
</table>
Table 31 presents the differences in command line parameters between IPM tools and their corresponding regular tools.

Table 31  IPM Tools—Command Line Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IPM Parameters</strong></td>
<td></td>
</tr>
<tr>
<td>-reliability time</td>
<td>You can control the reliability window of IPM measuring tools using this parameter; see Reliable Delivery &amp; Latency on page 254 in TIBCO Rendezvous Concepts. When present, IPM retains messages for time (in seconds). The value must be a non-negative integer. For least distortion of latency measurements, we recommend zero. When absent, IPM uses its own default (5 seconds).</td>
</tr>
<tr>
<td><strong>Invalid Parameters</strong></td>
<td></td>
</tr>
<tr>
<td>-daemon daemon</td>
<td>IPM tools ignore this parameter (which is available with the corresponding regular tools).</td>
</tr>
</tbody>
</table>
This chapter describes rvtrace, the Rendezvous protocol monitoring tool, which is distributed with Rendezvous Software Release 8.3.0.

Topics

- Overview, page 346
- Limitations, page 347
- Passive Monitor, page 349
- The pcap Facility, page 350
- Data Capture Files, page 352
- rvtrace, page 354
- Filtering, page 360
- Interpreting the Report, page 362
- Multicast Data Statistics, page 365
- Multicast Retransmit Statistics, page 369
- Point-to-Point Statistics, page 376
- Subject Statistics, page 383
- SNMP, page 386
Overview

Rendezvous protocol monitor, rvtrace, is a tool for network administrators. It monitors network packets, categorizes them, and reports statistics at regular intervals.

Network administrators can use rvtrace to diagnose network difficulties in real time, answering questions such as these:

- Which computers are inundating the network?
- Which computers are sending or receiving an inordinate number of retransmission requests?

Snapshots

rvtrace operates by capturing network packets, extracting information from packet headers, and gathering statistics about those packets. At the end of each interval, it compiles a statistical snapshot, and resets its counters for the next interval.

rvtrace can output those statistics in table format, or you can use SNMP to query the most recent snapshot.

Prerequisites

rvtrace is a tool for experienced network administrators.

- You must already understand IP protocols and addressing conventions.
- You must already understand Rendezvous software from an administrator’s perspective.
- To use rvtrace effectively, you must understand the topology of your network.

Licensing

rvtrace is sold and licensed separately from other Rendezvous components.

After purchasing rvtrace, you must include the rvtrace license ticket in the file tibrv.tkt. If rvtrace does not find a valid license ticket, it runs for a 10-minute evaluation period.
Limitations

Range Limitations

rvtrace cannot examine packets unless they traverse the immediate network segment in which rvtrace is running. For example, point-to-point conversations within or between other network segments are invisible to rvtrace. Most saliently, retransmission requests and retransmission rejections are point-to-point packets, so they are visible to rvtrace only when they either originate or terminate in the local network segment. Consequently, in some situations rvtrace can detect retransmission broadcasts, even though it cannot detect the point-to-point packets that request retransmissions.

Switched network environments (such as switched Ethernet, or ATM) further limit the usefulness of rvtrace as a diagnostic tool. Since switching hardware forwards every point-to-point packet directly to its destination host, rvtrace detects point-to-point packets only when they either originate or terminate in the computer running rvtrace. In some switched networks, you can ameliorate this situation by disabling switching behavior—for example, by setting one port to diagnostic mode, or by using a diagnostic utility. This tactic can yield the full stream of point-to-point packets in a limited portion of the network; run rvtrace in that portion.

In addition, some network switching hardware can route multicast packets to a network segment only when a host in the segment is actually listening to the corresponding multicast group. Such high specificity further limits the range of rvtrace.

Protocol Limitation

rvtrace supports only UDP multicast and UDP point-to-point protocols. It does not support PGM or RPTP protocols.

Interface Limitation

rvtrace supports only Ethernet interfaces.

rvtrace does not support these (or any other) non-Ethernet interfaces: Token Ring, FDDI, ATM.
Platform Support and Limitations

rvtrace operates on all platforms that Rendezvous supports—except VMS and z/OS.

TIBCO supports rvtrace on Microsoft Windows platforms, but requires that you first obtain and install WinPcap (see Obtaining pcap on page 350).

However, we do not support rvtrace on Windows SMP (Symmetric Multi-Processor) platforms at this time.

WinPcap does not support SMP platforms, and might not operate correctly in multiprocessor environments. For more information, see www.winpcap.org/misc/faq.htm.

On Windows platforms, we strongly recommend that you upgrade to the most recent version of WinPcap.

UnixWare places security restrictions on programs that open interfaces in promiscuous mode (such as pcap). To run rvtrace on UnixWare platforms, you must dedicate a separate physical interface for that purpose.

Mac OS X does not support rvtrace.
rvtrace is a passive monitor. It neither uses nor interferes with Rendezvous daemon processes. rvtrace does not add any packet traffic to the network.

rvtrace does not require a Rendezvous daemon for its operation. Instead, it collects network packets using the pcap facility.

Performance Effects

Although rvtrace is a passive monitor, it opens the network device in promiscuous mode, which consumes CPU and network resources on its host computer (in proportion to total network traffic). Running rvtrace on the same computer as any Rendezvous daemon (rvd or routing daemon) indirectly affects the operation of the local daemon by consuming these resources. Running rvd and rvtrace together on the same computer changes the timing and loading profiles of the host computer.

Avoid this situation whenever possible. Instead, run rvtrace on a computer that is otherwise free of Rendezvous activity.

We further recommend that rvtrace run on a computer that is fast enough to process every Ethernet packet that appears at its network interface.
The pcap Facility

rvtrace uses the pcap facility to capture network packets.

Obtaining pcap

Before using rvtrace, you must first ensure that the pcap facility is properly installed.

On most UNIX platforms, pcap is ready to use.

For Windows, you can download the WinPcap NDIS packet capture driver from this URL:

- http://www.winpcap.org/install/default.htm

For Windows platforms with multiple network interfaces, see also Selecting the Network Interface on page 351.

Packet Filtering

pcap has a flexible filtering language for selecting the set of packets to capture. rvtrace inherits this language through its -filter parameter.

You can select packets based on source, destination, host, network interface, port, packet length, and protocol. Packets that match the filter appear in rvtrace output; packets that do not match are ignored.

See Also

- -filter expr on page 356
- Filtering, page 360
Selecting the Network Interface

UNIX  On UNIX platforms with more than one network interface, use `iniftst` to determine the correct interface name.

Windows  On Windows platforms with more than one network interface, it is sometimes difficult to determine the correct interface name. The remainder of this section presents a method to determine it:

1. If data capture appears correct, then the remaining steps are not needed. However, if the captured data is all zeroes, then specify a different network interface (using `rvtrace -i`). Only one of the interfaces carries the data that `rvtrace` requires.

2. Install the WinDump utility from this URL:
   
   http://www.winpcap.org/windump/install/default.htm

3. Use this command to obtain a list of interface names:
   
   `windump -D`

4. Try each interface until the data appears reasonable.

Consider this example:

C:\>windump -D
1.\Device\Packet_{D7308399-80B2-4CA1-A9E6-C90DD74511A8} (EL574ND4 Ethernet Adapter)
2.\Device\Packet_NdisWanIp (NdisWan Adapter)

C:\>rvtrace -i \Device\Packet_{D7308399-80B2-4CA1-A9E6-C90DD74511A8}
Data Capture Files

rvtrace can write packets into a capture file, and read a stream of packets from a file (as if from the network).

Motivation

Packet capture files are an important tool for problem diagnosis. Several techniques are useful:

- Capture packet data for later analysis.
- Capture packet data for further analysis at a remote location.
- Capture packets at high speed, then replay later when I/O delays are acceptable.

In general, rvtrace can capture packets to a file faster than it can display statistics. Large amounts of display data can create I/O delays, which could cause rvtrace to miss packets. For example, in a heavily loaded network, displaying subject statistics for many subjects could have this undesirable result.

You can use data capture files to sidestep this difficulty. For example, capture a five-minute snapshot of packets (capturing suppresses display); then replay packets from the file, displaying statistics when the consequences of I/O delays are no longer problematic.

Output File Rotation

The rotation regimen for data capture output is almost identical to the rotation regimen for log files; see Log Rotation on page 54.

The only difference between them, is that rvtrace always deletes an older existing file before opening a file for writing packet data. (That is, it never appends to the end of an existing data capture file.)
Output Rotation Backward Compatibility

The command line option `-w-rotate total_size` is deprecated in release 7.5, and will become obsolete in a subsequent release. We recommend migrating to the new rotation parameters at your earliest convenience.

In the meantime, we preserve backward compatibility by converting the value of this deprecated parameter to corresponding values of the new parameters:

- `-w-rotate total_size` retains its old meaning—specifying the total size for all data capture files. The maximum size for each individual file in the rotation is `total_size/10`.

- If both old and new parameters are present, the new parameters take precedence (overriding the old parameter).
rvtrace

Command

Syntax rvtrace [ -i interface ]
[-r input_file ]
[-addr expr ]
[-src expr ]
[-dst expr ]
[-port expr ]
[-filter expr ]
[-w output_file ]
[-w-max-size size ]
[-w-max-rotations n ]
[-no-display]
[-addrinfo]
[-u update_interval ]
[-no-mcast]
[-ptp]
[-no-subjects]
[-hostmsgs]
[-rate]
[-logfile log_file ]
[-log-max-size size ]
[-log-max-rotations n ]
[-snmp]
[-foreground]
[-h]

Purpose rvtrace is a network protocol monitor that specializes in Rendezvous protocols. It collects and prints statistics about network packets.

Remarks rvtrace runs in a loop—capturing packets, analyzing them, categorizing them, and periodically printing a summary to standard output.

An rvtrace process never exits by itself (except as a consequence of a command syntax error); you must explicitly terminate each process.

Delimit all parameters and arguments with a space character.

(Sheet 1 of 6)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
<td></td>
</tr>
<tr>
<td>-i interface</td>
<td>The program monitors packets on the network interface with this name. If absent, the default value varies, depending on operating system and network hardware. For Windows platforms, see also Selecting the Network Interface on page 351.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>-r input_file</td>
<td>When present, read recorded packets from input_file instead of a network interface. This option overrides the -i parameter. For more information, see Data Capture Files on page 352.</td>
</tr>
</tbody>
</table>

### Data Filtering

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| -addr expr    | Filter the set of packets to process only those with source or destination in the set of hosts or networks specified in expr. For filter expression syntax and semantics, see Filtering on page 360. Enclose filter expressions in quotation marks (").

The parameter -addr filter is equivalent to:
-filter udp and (src filter or dst filter)

When any of the parameters -src, -dst, -addr, or -port are present, rvtrace concatenates them into a single effective filter. However, when the -filter parameter is present, rvtrace ignores all four of these parameters, and -filter overrides them.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| -src expr     | Filter the set of packets to process only those that originate from the set of hosts or networks specified in expr. For filter expression syntax and semantics, see Filtering on page 360. Enclose filter expressions in quotation marks (").

The parameter -src expr is equivalent to:
-filter udp and src expr

When any of the parameters -src, -dst, -addr, or -port are present, rvtrace concatenates them into a single effective filter. However, when the -filter parameter is present, rvtrace ignores all four of these parameters, and -filter overrides them.
Filter the set of packets to process only those with destination in the set of hosts or networks specified in `expr`. For filter expression syntax and semantics, see Filtering on page 360.

Enclose filter expressions in quotation marks (".").

The parameter `dst filter` is equivalent to:
- `filter udp and dst filter`

When any of the parameters `src`, `dst`, `addr`, or `port` are present, `rvtrace` concatenates them into a single effective filter. However, when the `-filter` parameter is present, `rvtrace` ignores all four of these parameters, and `-filter` overrides them.

Filter the set of packets to process only those with destination port in the set of ports specified in `expr`. For filter expression syntax and semantics, see Filtering on page 360.

Enclose filter expressions in quotation marks (".").

The parameter `port port` is equivalent to:
- `filter udp and dst port port`

When any of the parameters `src`, `dst`, `addr`, or `port` are present, `rvtrace` concatenates them into a single effective filter. However, when the `-filter` parameter is present, `rvtrace` ignores all four of these parameters, and `-filter` overrides them.

Filter the set of packets to process only those that match `expr`. For filter expression syntax and semantics, see Filtering on page 360.

Enclose filter expressions in quotation marks (".").

When present, this parameter overrides the `src`, `dst`, `addr`, and `port` parameters.
When present, write packet information to `output_file` for later replay or analysis.

When absent, do not record packet information to a file.

For more information, see Data Capture Files on page 352.

When `-w` is present, `rvtrace` does not display statistics. To see statistics, use `-r` to read the packet capture file.

When both `-r` and `-w` are present, `rvtrace` reads packets from `input_file`, filters them, and then recaptures the filtered packets to `output_file`. You can use this technique to prune an existing capture file by reducing information or filtering extraneous traffic.

- `-w-max-size size`  
  When present, activate the capture-file rotation regimen (see Data Capture Files on page 352 and Log Rotation on page 54).

- `-w-max-rotations n`  
  When you specify these options, you must also specify `-w`.

  `size` is in megabytes. If `size` is non-zero, it must be in the range [100, 2097152]. Values outside this range are automatically adjusted to the nearest acceptable value. Zero is a special value, which disables rotation. When `-w-max-size` is zero or absent, a single capture file may grow without limit (other than the limit of available storage).

  `n` indicates the maximum number of files in the rotation. When `-w-max-rotations` is absent, the default value is 10.

When present, do not output statistics. Nonetheless, `rvtrace` continues to compile statistics, which are available through SNMP queries.

When absent, `rvtrace` outputs statistics (either to stdout, or to a log file).

Summarize network packet at this time interval (in seconds). If absent, the default value is 10 seconds.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-addrinfo</td>
<td>When present, display network totals, subtotals, and detail rows. When absent, display only network totals and subtotal rows. For example output, see Figure 81, rvtrace Output with -addrinfo, on page 363, and Figure 82, rvtrace Output without -addrinfo, on page 363.</td>
</tr>
<tr>
<td>-no-mcast</td>
<td>When present, omit the multicast table. When absent, display the multicast table; see Multicast Data Statistics on page 365.</td>
</tr>
<tr>
<td>-ptp</td>
<td>When present, display the point-to-point table; see Point-to-Point Statistics on page 376; see also Range Limitations on page 347. When absent, omit the point-to-point table.</td>
</tr>
<tr>
<td>-no-subjects</td>
<td>When present, omit the subject table. When absent, display the subject table; see Subject Statistics on page 383.</td>
</tr>
<tr>
<td>-hostmsgs</td>
<td>When present, display Rendezvous HOST messages at the conclusion of each interval. TIBCO personnel might request that you supply rvtrace output transcript that includes these messages. These messages useful only to TIBCO personnel.</td>
</tr>
<tr>
<td>-rate</td>
<td>When present, display packet counts as per-second rates. When absent, display the actual number of packets in the update interval.</td>
</tr>
</tbody>
</table>

**Log Output**

| -logfile log_file | Send log output to this file. When absent, the default is stdout.                                                                                       |
rvtrace uses the pcap facility, which requires root privileges (because it must open the raw ethernet device in promiscuous mode). Without appropriate privileges, pcap denies permission to initialize, and rvtrace exits immediately.

- The pcap library calls reject improperly formed filter expressions. It reports them with messages such as this:
  
  pcap_compile: syntax error

  This error causes rvtrace to exit.
Filtering

You can modify the set of packets that `rvtrace` processes by supplying either the `-filter` parameter, or a combination of the `-src`, `-dst`, `-addr`, and `-port` parameters. Filter expressions specify the set of packets to process.

`rvtrace` uses the `pcap` facility to capture and filter packets. The `tcpdump` utility also uses `pcap`, so the syntax and semantics for `rvtrace` and `tcpdump` filter expressions are identical. Table 32 summarizes the subset of filter expressions that are relevant to `rvtrace`; for additional options, see documentation for `tcpdump`. (Disclaimer: `pcap` and `tcpdump` are not TIBCO products; we do not sell, support or document them.)

Each row of Table 32 constitutes an `expr`, and can be used in place of the syntax marker `expr` elsewhere in Table 32, and in the parameter table for `rvtrace`.

When specifying a filter expression to an `rvtrace` parameter, enclose the expression in quotation marks (").

Table 32  Filter Expressions (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host Expressions</strong></td>
<td></td>
</tr>
<tr>
<td><code>host host</code></td>
<td>Process a packet if either the IP source or destination of the packet is <code>host</code>. Specify <code>host</code> as a name or an IP address.</td>
</tr>
<tr>
<td><code>dst host host</code></td>
<td>Process a packet if its IP destination is <code>host</code>.</td>
</tr>
<tr>
<td><code>src host host</code></td>
<td>Process a packet if its IP source is <code>host</code>.</td>
</tr>
<tr>
<td><strong>Network Expressions</strong></td>
<td></td>
</tr>
<tr>
<td><code>net net</code></td>
<td>Process a packet if either the IP source or destination of the packet is <code>net</code>. Specify <code>net</code> as a name or an IP network number.</td>
</tr>
<tr>
<td><code>dst net net</code></td>
<td>Process a packet if its IP destination is <code>net</code>.</td>
</tr>
<tr>
<td><code>src net net</code></td>
<td>Process a packet if its IP source is <code>net</code>.</td>
</tr>
</tbody>
</table>
### Table 32  Filter Expressions (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port or Service Expressions</strong></td>
<td></td>
</tr>
<tr>
<td><code>port port</code></td>
<td>Process a packet if either the IP source or destination port of the packet is <code>port</code>. Specify <code>port</code> as a service name or a UDP port number.</td>
</tr>
<tr>
<td><code>dst port port</code></td>
<td>Process a packet if its IP destination port is <code>port</code>.</td>
</tr>
<tr>
<td><code>src port port</code></td>
<td>Process a packet if its IP source port is <code>port</code>.</td>
</tr>
<tr>
<td><strong>Broadcast or Multicast Expressions</strong></td>
<td></td>
</tr>
<tr>
<td><code>ip broadcast</code></td>
<td>Process a packet if it is an IP broadcast packet.</td>
</tr>
<tr>
<td><code>ip broadcast expr</code></td>
<td>If <code>expr</code> is present, then process the packet only if it also meets the criteria of <code>expr</code>.</td>
</tr>
<tr>
<td><code>ip multicast</code></td>
<td>Process a packet if it is an IP multicast packet.</td>
</tr>
<tr>
<td><code>ip multicast expr</code></td>
<td>If <code>expr</code> is present, then process the packet only if it also meets the criteria of <code>expr</code>.</td>
</tr>
<tr>
<td><strong>Protocol Expressions</strong></td>
<td></td>
</tr>
<tr>
<td><code>udp</code></td>
<td>Process a packet if it is an IP packet with protocol type <code>udp</code>. (All Rendezvous packets are UDP packets.)</td>
</tr>
<tr>
<td><code>udp expr</code></td>
<td>If <code>expr</code> is present, then process the packet only if it also meets the criteria of <code>expr</code>.</td>
</tr>
<tr>
<td><code>ip</code></td>
<td>Process a packet if it is an IP packet.</td>
</tr>
<tr>
<td><code>ip expr</code></td>
<td>If <code>expr</code> is present, then process the packet only if it also meets the criteria of <code>expr</code>.</td>
</tr>
<tr>
<td><strong>Boolean Operators</strong></td>
<td>Use parentheses to group boolean expressions; use appropriate escape characters to override shell-specific semantics of parentheses.</td>
</tr>
<tr>
<td><code>expr1 and expr2</code></td>
<td>Process a packet if it meets both criteria.</td>
</tr>
<tr>
<td><code>expr1 expr2</code></td>
<td></td>
</tr>
<tr>
<td><code>expr1 or expr2</code></td>
<td>Process a packet if it meets either criterion.</td>
</tr>
<tr>
<td><code>not expr</code></td>
<td>Process a packet if it does not meet the criterion.</td>
</tr>
</tbody>
</table>
Interpreting the Report

The remaining sections of this chapter describe the output from rvtrace.

Figure 81 on page 363 shows a sample of the output that rvtrace prints at the conclusion of each interval, when the -addrinfo flag is present:

- Time stamp—identifies the interval
- Multicast Data Statistics—summarizes Rendezvous multicast and broadcast packets during the interval, organized by UDP port (service) and destination address (see Multicast Data Statistics on page 365)
- Multicast Retrans Statistics—summarizes requests to retransmit packets of multicast and broadcast data (this table does not appear in Figure 81; see Multicast Retransmit Statistics on page 369)
- PTP Statistics—summarizes Rendezvous point-to-point packets during the interval, organized by UDP port (service) and destination address (see Point-to-Point Statistics on page 376)
- Subject Statistics—recapitulates Rendezvous multicast and broadcast message activity during the interval, featuring information about subject names (see Subject Statistics on page 383)

Notice that each table begins with a network total, and then breaks down the total into subtotals and fine-grained categories.

Figure 82 on page 363 shows a sample of the less verbose output that rvtrace prints when the -addrinfo flag is absent. Notice that tables omit the fine-grained categories—displaying only the network total and subtotals

General Network Load

To assess network load, inspect the Data and Bytes columns of the Multicast Data Statistics table, and the Data and Bytes columns of the Point-to-Point Statistics table.

Number of Senders

To determine the number of Rendezvous daemons that sent data messages during an interval, count the number of distinct source addresses in all source rows of the Multicast Data Statistics table and the Point-to-Point Statistics table.
Scanning for Problems

To quickly review `rvtrace` output for problems, scan down the right side of the page, looking for non-zero values in the **Bad**, **Gaps**, and **Rbytes** columns of the multicast data tables. Non-zero values in these columns indicate a problem; look more closely at statistics in other columns in that interval and subsequent intervals.

**Figure 81  `rvtrace` Output with `-addrinfo`**

<table>
<thead>
<tr>
<th>Port</th>
<th>Address</th>
<th>SCid</th>
<th>Data</th>
<th>Bytes</th>
<th>Null</th>
<th>Rdata</th>
<th>Rbytes</th>
<th>Gaps</th>
<th>Bad</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Totals</td>
<td>22</td>
<td>3978</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>5039</td>
<td>10.97.128.255</td>
<td>11</td>
<td>1989</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>10.97.128.30</td>
<td>1</td>
<td>529</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>60</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>10.97.128.31</td>
<td>10</td>
<td>1460</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>60</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>5662</td>
<td>10.97.128.255</td>
<td>11</td>
<td>1989</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>10.97.128.30</td>
<td>1</td>
<td>529</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>60</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>10.97.128.31</td>
<td>10</td>
<td>1460</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>60</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

**Figure 82  `rvtrace` Output without `-addrinfo`**

<table>
<thead>
<tr>
<th>Port</th>
<th>Address</th>
<th>SCid</th>
<th>Data</th>
<th>Bytes</th>
<th>Null</th>
<th>Rdata</th>
<th>Rbytes</th>
<th>Gaps</th>
<th>Bad</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Totals</td>
<td>22</td>
<td>3978</td>
<td>23</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5039</td>
<td>10.97.128.255</td>
<td>10</td>
<td>1460</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5662</td>
<td>10.97.128.255</td>
<td>11</td>
<td>1989</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Multicast Subject Statistics**

<table>
<thead>
<tr>
<th>Port</th>
<th>Address</th>
<th>SCid</th>
<th>Mids</th>
<th>Bytes</th>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Totals</td>
<td>21</td>
<td>3449</td>
<td>23</td>
<td>_RV.INFO.SYSTEM.HOST.STATUS.OA61801E</td>
</tr>
<tr>
<td>5039</td>
<td>10.97.128.255</td>
<td>10</td>
<td>1460</td>
<td>11</td>
<td>subject.1</td>
</tr>
<tr>
<td>5662</td>
<td>10.97.128.255</td>
<td>11</td>
<td>1989</td>
<td>12</td>
<td>_RV.INFO.SYSTEM.HOST.STATUS.OA61801E</td>
</tr>
<tr>
<td>5662</td>
<td>10.97.128.255</td>
<td>10</td>
<td>1460</td>
<td>10</td>
<td>subject.1</td>
</tr>
</tbody>
</table>

TIBCO Rendezvous Administration
Bad Packets

Bad packets lack UDP checksums, or are corrupt in some other way.

Bad packets usually indicate a severe misconfiguration or network problem. Remedy the situation immediately.

Checksums are crucial to correct operation of Rendezvous software; see Enable Packet Checksums on page 6.

False Bad Packets

In some situations, `rvtrace` can incorrectly report bad packets.

When a sending host computer enables checksum off-loading features, the network interface card (rather than the CPU) adds checksums to outbound packets. If `rvtrace` is running on the same host as the sender, it captures outbound packets before the checksums have been added. `rvtrace` detects the missing checksums, and reports bad packets. However, by the time these packets actually reach the network, they might not be bad packets.
**Multicast Data Statistics**

Figure 83 shows a multicast data table (from `rvtrace -addrinfo`). The text below introduces important concepts. Table 33 on page 366 describes the columns in detail.

**Figure 83  Multicast Packet Statistics**

<table>
<thead>
<tr>
<th>Port</th>
<th>Address</th>
<th>SCid</th>
<th>Data</th>
<th>Bytes</th>
<th>Null</th>
<th>Rdata</th>
<th>Rbytes</th>
<th>Gaps</th>
<th>Bad</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totals</td>
<td></td>
<td></td>
<td>79</td>
<td>23758</td>
<td>47</td>
<td>4</td>
<td>1032</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5863 *</td>
<td>10.101.2.255</td>
<td>15</td>
<td>4298</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5864 *</td>
<td>10.101.2.255</td>
<td>0</td>
<td>15</td>
<td>4298</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>5865 *</td>
<td>10.101.2.255</td>
<td>0</td>
<td>8</td>
<td>2225</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5866 *</td>
<td>10.101.2.255</td>
<td>8</td>
<td>2225</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>5867 *</td>
<td>10.101.2.255</td>
<td>12</td>
<td>4086</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5868</td>
<td>10.101.2.36</td>
<td>0</td>
<td>12</td>
<td>4086</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20000 *</td>
<td>224.1.1.12</td>
<td>10</td>
<td>3790</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>10.101.2.102</td>
<td>39365</td>
<td>10</td>
<td>2580</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>10.101.2.102</td>
<td>39364</td>
<td>10</td>
<td>2580</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>516</td>
</tr>
</tbody>
</table>

Notice that the rows divide visually into six groups, as indicated by a number in the **Port** column and an asterisk (*).

**Network Total Row**

The first row (immediately after the table and column headings, and before the four groups) is a **network total row**; the word *Totals* in the **Address** column is a visual cue. This row shows the grand total of multicast and broadcast packets on the network during the interval. For example, the **Data** column shows the total number of data packets that `rvtrace` detected on the network during the interval.

The remaining rows display more fine-grained information about those packets—grouping them by UDP service, destination address, and source address.

**Subtotal Groups**

A number in the **Port** column indicates the UDP service for its row, and the group of rows that follow it. A blank in this column means that the row has the same port as the row above, and is part of the same subtotal group. Notice how the pattern of numbers and blanks in the **Port** column visually indicates the subtotal groups.

**Destination Row**

* flags a row as a **destination subtotal row**. A blank (space character) in this column flags a row as a **source row**. Each group begins with a destination subtotal row, followed by one or more source rows.
Each destination subtotal row is the heading and subtotal for the source rows that follow it. For example, consider the destination row with 20000 in the Port column, and 224.1.1.12 in the Address column. The Data column indicates 20 packets on UDP service 20000 sent to the multicast group 224.1.1.12. The two subsequent source rows indicate that those 20 packets came from two sources—the daemon or IPM with SCid 39365 at 10.101.2.102 sent 10 packets, while SCid 39364 at the same host sent another 10 packets. The subtotal 20 in turn contributes to the grand total 51 in the first row.

A destination subtotal row governs the source rows below it (until the next destination subtotal row). That is, the UDP service (port) and address in the governing row apply to those source rows. Similarly, the governing row address implies either multicast or broadcast protocol, and this protocol also applies to the statistics in the source rows that it governs. (Naturally, all of this information also applies to the governing row itself.)

**Source Row**

Each source row shows a very narrow subset of packet activity during the interval—packets on a specific UDP service (port), with a specific destination address, and originating at a specific source (IP address). The Address column shows the source; the UDP service and destination address are specified in the governing row (that is, the nearest preceding destination subtotal row).

**Statistics**

In destination rows numbers in statistics columns count packets with the destination specified in the Address column.

In source rows numbers in statistics columns count packets originating from the IP address in the Address column.

In network total rows, numbers in statistics columns represent the packet totals for the network during the interval.

*Table 33  Multicast Packet Statistics—Column Headings (Sheet 1 of 3)*

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>In destination subtotal rows, this column contains a UDP port number indicating the Rendezvous service for the group of rows that it begins. In source rows this column is blank; the service in the nearest preceding destination row also applies to the source row.</td>
</tr>
<tr>
<td>*</td>
<td>Asterisk (*) in this unlabeled column indicates a destination subtotal row. Blank in this column indicates a source row.</td>
</tr>
</tbody>
</table>
Table 33  Multicast Packet Statistics—Column Headings (Sheet 2 of 3)

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
</table>
| Address | In *destination rows* this column shows the destination address shared among a group of packets. It can be an IP address or a multicast group.  
In *source rows* this column shows the IP address from which group of packets originate.  
In *network total rows*, this column contains the word *Totals*. |
| SCid | Service communication ID.  
In *source rows*, this value differentiates the source of packets when several daemons or IPM instances on the same host computer reuse the same service port.  
Zero in this column indicates that the source is the only sender on that service and host.  
In destination rows this column is blank. |
| Data | Data packets.  
This column shows the number of multicast or broadcast data packets. |
| Bytes | Data bytes.  
This column sums the number of payload bytes over the data packets (as counted in the *Data* column). |
| Null | Null packets.  
When a Rendezvous daemon has no data packets to transmit, it periodically sends *null packets* to maintain continuity. This column displays the number of null packets that *rvtrace* detected. |
| Rdata | Retransmitted data packets.  
*rvttrace* counts retransmitted packets separately from first-time data packets. This column displays the number *retransmitted* data packets during the interval. Semantics of this column are otherwise analogous to the *Data* column.  
For statistics concerning retransmission requests and rejections, see *Multicast Retransmit Statistics* on page 369. |
| Rbytes | Retransmitted bytes.  
This column sums the number of payload bytes over the retransmitted data packets (as counted in the *Rdata* column). |
A sequence gap can occur in two situations:

- `rvtrace` misses one or more packets; that is, the hardware or operating system on which `rvtrace` is running drops one or more packets.
- The network infrastructure drops one or more packets between their source and `rvtrace`.

To determine which of these two situations has actually occurred, check the `Rdata` values within the interval and in subsequent intervals. If `Rdata` remains at zero, then it is likely that `rvtrace` alone is missing packets. If `Rdata` is non-zero, then it is likely that the network infrastructure is dropping packets (`Rdata` is non-zero because other daemons on the network are requesting retransmission of the missing packets).

---

**Table 33  Multicast Packet Statistics—Column Headings (Sheet 3 of 3)**

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaps</td>
<td>Sequence gaps. <code>rvtrace</code> tracks the serial numbers of Rendezvous packets. The <strong>Gaps</strong> column counts the missing packets in each sequence of multicast or broadcast data packets. For more information, see <strong>Gaps Diagnoses on page 368</strong>.</td>
</tr>
</tbody>
</table>
| Bad    | Bad packets. This column shows the number of packets that lack UDP checksums, or are corrupt in some other way. ![Warning]
> See **Bad Packets on page 364**. |
| R      | Reliability. A numeric value indicates the reliability of a specific service on a specific host. Hyphen (–) is a place holder in rows that don’t represent a host (that is, in port rows and total rows). |
Multicast Retransmit Statistics

Sending Rendezvous daemon process instances retransmit missed packets upon request from receiving Rendezvous daemons. This table displays statistics related to those retransmission requests. For statistics concerning the actual retransmitted packets, see Multicast Data Statistics on page 365—in particular, the \textit{Rdata} and \textit{Rbytes} columns.

The \textit{Rreq} column of this table counts point-to-point packets. In contrast, the actual retransmitted data packets use the same protocol (multicast or broadcast) as the original data packets that they recapitulate (as do the rejection notices in the \textit{Rrej} column).

In switched ethernet environments point-to-point packets remain invisible to \texttt{rvtrace}—except for packets addressed specifically to the \texttt{rvtrace} host computer. This fact severely limits the usefulness of retransmit statistics in switched networks.

In some switched networks, you can disable switching behavior—for example, by setting one port to diagnostic mode, or by using a diagnostic utility. This tactic can yield the full stream of point-to-point packets in a limited portion of the network; run \texttt{rvtrace} in that portion.

Figure 84 shows a multicast retransmit table (from \texttt{rvtrace -addrinfo}). The text below introduces important concepts. Table 34 on page 370 describes the columns in detail.

\begin{table}[h]
\centering
\begin{tabular}{llrrrrr}
\hline
Port & Address & Scid & Rreq & Rreq & Rrej & Bad \\
\hline
Totals & & 12 & 54 & 0 & 12 \\
5662 & & 4 & 20 & 0 & 4 \\
\* & 10.97.128.30 & - & 0 & 0 & 0 & 0 \\
\* & 10.97.128.31 & 50416 & 4 & 20 & 0 & 4 \\
5039 & & 8 & 34 & 0 & 8 \\
\* & 10.97.128.30 & - & 0 & 0 & 0 & 0 \\
\* & 10.97.128.31 & 58026 & 8 & 34 & 0 & 8 \\
\end{tabular}
\caption{Multicast Retransmit Packet Statistics}
\end{table}
The first row (immediately after the table and column headings) is a network total row; the word Totals in the Address column is a visual cue. This row shows the grand total of packets related to retransmission detected on the network during the interval.

The remaining rows display more fine-grained information about those packets—grouping them by UDP service, and destination or source IP address.

The second row in Figure 84 is a port subtotal row—its columns subtotal the statistics over the subsequent destination and source rows which it governs (until the next port subtotal row). A number in the Port column indicates the UDP service for its row, and the group of rows that follow it. A blank in this column means that the row has the same port as the row above, and is part of the same subtotal group. Notice how the pattern of numbers and blanks in the Port column visually indicates the subtotal groups.

For each IP address with retransmission request activity, this table contains a destination row and a source row—always paired in that order. An * and an IP address (in the Address column) flags a row as a destination row. A blank (space characters) flags a row as a source row. The address in the destination row also applies to the source row that immediately follows it.

This table displays each packet twice—once in a destination row, and once in a source row.

In each statistical column, the number in the port subtotal row is equal to the sum of the values in the destination rows, which is also equal to the sum of the values in the source rows.

In many networks it is possible to match the numbers in the source row for one IP address against the numbers in the destination row for another IP address. From this information you can deduce which Rendezvous daemons are missing packets and requesting retransmissions.

Table 34  Multicast Retransmit Packet Statistics—Column Headings (Sheet 1 of 3)

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>In port subtotal rows, this column contains a UDP port number indicating the Rendezvous service for the group of rows that it begins. In destination and source rows this column is blank; the service in the nearest preceding port subtotal row governs the destination and source rows below it.</td>
</tr>
<tr>
<td>*</td>
<td>Asterisk (*) in this unlabeled column indicates a destination row. Blank in this column indicates a source row.</td>
</tr>
</tbody>
</table>
### Table 34  Multicast Retransmit Packet Statistics—Column Headings (Sheet 2 of 3)

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>In <em>destination rows</em> this column shows the destination IP address of retransmission request packets (that is, the Rendezvous daemon that originally sent data packets). In <em>source rows</em> this column shows the IP address from which retransmission request or rejection packets originate (that is, the Rendezvous daemon that missed receiving data packets). In <em>network total rows</em>, this column contains the word <strong>Totals</strong>.</td>
</tr>
<tr>
<td>SCid</td>
<td>Service communication ID. In <em>destination rows</em> this column differentiates the destination ID of retransmission request packets (that is, the Rendezvous daemon that originally sent data packets). In <em>source rows</em> this column is blank.</td>
</tr>
<tr>
<td>Rreq</td>
<td>Retransmission requests. This column displays the number of packets that contain retransmission requests. Each request packet counts separately, even if several request packets specify the same data packet numbers for retransmission. For example, if two daemons each request retransmission of the data packets numbered 121–125, and a third daemon requests retransmission of the data packets numbered 100–144, then $Rreq$ is 3.</td>
</tr>
<tr>
<td>Rseq</td>
<td>Retransmission sequence numbers. Each retransmission request packet can solicit one or more data packets for retransmission. This column sums the number of data packets for which retransmission is requested over the request packets (as counted in the $Rreq$ column). If some data packets are requested several times, each data packet counts separately each time it is requested. For example, if three daemons request retransmission of data packets numbered 121–125, then the $Rseq$ sum is 15. For more information, see <strong>Diagnoses on page 372</strong>.</td>
</tr>
<tr>
<td>Rrej</td>
<td>Retransmission rejection notices. Although Rendezvous daemons comply with retransmission requests whenever possible, sometimes the requested packets are no longer available. This column displays the number of packets that contain retransmission rejections. (Daemons send these rejection notices in multicast packets.)</td>
</tr>
</tbody>
</table>
Table 34  Multicast Retransmit Packet Statistics—Column Headings (Sheet 3 of 3)

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bad</td>
<td>Bad packets. This column shows the number of packets that lack UDP checksums, or are corrupt in some other way.</td>
</tr>
</tbody>
</table>

⚠️ See Bad Packets on page 364.

Diagnoses

Scanning for Problems on page 363 described a quick scanning technique for locating problems in rvtrace output, looking for non-zero values in the Bad, Gaps, and Bytes columns of the multicast data tables. When such a scan indicates problems, look more closely at the retransmit statistics in nearby intervals.

Rseq measures retransmission requests for missed multicast or broadcast packets. Non-zero Rseq values generally indicate a problem. The ratio $\frac{Rdata}{Data}$ measures the severity of the problem. Small ratios indicate low-level problems, which you can investigate as time permits. Ratios of 2% or greater indicate potentially serious network problems; investigate further. High ratios that last for only one interval, could indicate an intermittent problem, which could become more serious in other situations.

Notice that Rseq tabulates packets that serve a feedback mechanism within the protocol. A data receiver becomes a feedback sender when it detects that it has missed data packets. So the Rseq value in source rows indicates a data receiver sending retransmission requests. Conversely, the Rseq value in destination rows indicates a data sender receiving retransmission requests.

Consider the following two examples.

Difficulty at One Specific Receiver

Figure 85 on page 373 shows rvtrace output for three intervals, which indicate a difficulty at one specific receiver. The administrator must investigate that receiver, its network hardware, and its load.

Several situations could cause this pattern in rvtrace display output. For example:
- One slow computer is flooded by too much data from a network of faster senders; the receiver cannot process inbound data as fast as the rest of the network.
- One receiver with intermittent network interface failures or a loose network cable.

**Figure 85  Rseq Reveals Difficulty at a Receiver**

### Multicast Packet Statistics

<table>
<thead>
<tr>
<th>Port</th>
<th>Address</th>
<th>SCid</th>
<th>Data</th>
<th>Bytes</th>
<th>Null</th>
<th>Rdata</th>
<th>Rbytes</th>
<th>Gaps</th>
<th>Bad</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totals</td>
<td>989</td>
<td>311535</td>
<td>1</td>
<td>641</td>
<td>201959</td>
<td>9</td>
<td>0</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 7599 | 225.9.9.10 | 989 | 311535 | 1 | 641 | 201959 | 9 | 0 | - |
| 10.101.3.237 | 49039 | 989 | 311535 | 0 | 640 | 201600 | 9 | 0 | 60 |
| 10.101.3.251 | 34523 | 0 | 0 | 1 | 1 | 359 | 0 | 0 | 60 |

### Multicast Retrans Packet Statistics

<table>
<thead>
<tr>
<th>Port</th>
<th>Address</th>
<th>SCid</th>
<th>Rseq</th>
<th>Rreq</th>
<th>Rseq</th>
<th>Rreq</th>
<th>Bad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totals</td>
<td></td>
<td>20</td>
<td>2222</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 7599 | | 20 | 7 | 0 | 0 |
| * 10.101.3.74 | | 0 | 0 | 0 | 0 |
| * 10.101.3.237 | 49039 | 20 | 2222 | 0 | 0 |
| * 10.101.3.246 | | 0 | 0 | 0 | 0 |
| * 10.101.3.251 | 34523 | 0 | 0 | 0 | 0 |

### Snapshot 2010-06-02 10:14:38 (10.0 elapsed seconds)

<table>
<thead>
<tr>
<th>Port</th>
<th>Address</th>
<th>SCid</th>
<th>Data</th>
<th>Bytes</th>
<th>Null</th>
<th>Rdata</th>
<th>Rbytes</th>
<th>Gaps</th>
<th>Bad</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totals</td>
<td>1000</td>
<td>315044</td>
<td>2</td>
<td>62</td>
<td>19530</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 7599 | 225.9.9.10 | 1000 | 315044 | 2 | 62 | 19530 | 0 | 0 | - |
| 10.101.3.237 | 49039 | 999 | 314685 | 0 | 62 | 19530 | 0 | 0 | 60 |
| 10.101.3.246 | 34523 | 1 | 359 | 2 | 0 | 0 | 0 | 0 | 60 |

### Snapshot 2010-06-02 10:14:48 (10.0 elapsed seconds)

<table>
<thead>
<tr>
<th>Port</th>
<th>Address</th>
<th>SCid</th>
<th>Data</th>
<th>Bytes</th>
<th>Null</th>
<th>Rdata</th>
<th>Rbytes</th>
<th>Gaps</th>
<th>Bad</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totals</td>
<td></td>
<td>999</td>
<td>314685</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

| 7599 | 225.9.9.10 | 999 | 314685 | 2 | 0 | 0 | 0 | 0 | 0 | - |
| 10.101.3.237 | 49039 | 999 | 314685 | 0 | 0 | 0 | 0 | 0 | 60 |
| 10.101.3.246 | 33201 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 60 |
| 10.101.3.251 | 34523 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 60 |
In Figure 85 on page 373, the first interval shows 9 sequence gaps in the multicast statistics table—that is, 9 gaps in the stream of multicast packets. The Rseq column of the multicast retransmit table contains further details; the host at address 10.101.3.246 requested 2211 packets for retransmission, while the other hosts requested a total of 11 packets. Conclude that the locus of the problem is at 10.101.3.246, and that retransmit requests from the other receivers are side effects.

The second interval of Figure 85 shows zero sequence gaps—the problem has abated. Nonetheless, the Rdata and Rbytes columns indicate that retransmissions continue as Rendezvous daemons recover from the problem by resending stored data.

By the third interval of Figure 85, everything has returned to normal.

**Difficulty at One Specific Sender**

Figure 86 on page 375 shows output indicating a difficulty at one specific sender. The administrator must investigate that sender, its sending applications, and its network hardware.

Several situations could cause this pattern in rvtrace display output. For example:

- The sender is flooding the network—that is, it is sending packets faster than most other daemons on the network can receive them.
- The sender has intermittent network interface failures or a loose network cable.

In Figure 86, the multicast statistics table shows 411 sequence gaps—that is, 411 gaps in the stream of multicast packets. Moreover, all the missing packets originate at one sender, 10.101.3.246. The Rseq column of the multicast retransmit table contains further details; both of the receivers in the network requested those packets for retransmission—that is 10.101.3.74 and 10.101.3.237 both sent retransmit requests to 10.101.3.246. Conclude that the locus of the problem is at 10.101.3.246.

The Rdata column of the multicast statistics table shows that before the end of the interval, the sender had retransmitted all 411 missing packets. The problem was a brief glitch—the Rendezvous reliable transport mechanisms easily smoothed over this temporary rough spot. Nonetheless, if such behavior recurs, the administrator must investigate the problem.
### Rseq Reveals Difficulty at a Sender

**Figure 86**

**Snapshot 2010-05-12 08:49:37 (10.0 elapsed seconds)**

<table>
<thead>
<tr>
<th>Port</th>
<th>Address</th>
<th>SCid</th>
<th>Data</th>
<th>Bytes</th>
<th>Null</th>
<th>Rdata</th>
<th>Rbytes</th>
<th>Gaps</th>
<th>Bad</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totals</td>
<td></td>
<td>586</td>
<td>185220</td>
<td>0</td>
<td>411</td>
<td></td>
<td>129465</td>
<td>411</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>7599</td>
<td>225.9.9.10</td>
<td>586</td>
<td>185220</td>
<td>0</td>
<td>411</td>
<td></td>
<td>129465</td>
<td>411</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>10.101.3.246</td>
<td>49039</td>
<td>586</td>
<td>185220</td>
<td>0</td>
<td>411</td>
<td>129465</td>
<td>411</td>
<td>0</td>
<td>60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Port</th>
<th>Address</th>
<th>SCid</th>
<th>Freq</th>
<th>Rseq</th>
<th>Rreq</th>
<th>Bad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totals</td>
<td></td>
<td>6</td>
<td>822</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7599</td>
<td></td>
<td>6</td>
<td>822</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>* 10.101.3.74</td>
<td></td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>* 10.101.3.237</td>
<td></td>
<td>3</td>
<td>411</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>* 10.101.3.246</td>
<td></td>
<td>3</td>
<td>411</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>* 10.101.3.246</td>
<td></td>
<td>6</td>
<td>822</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

* Notes: * indicates a multicast sender.
Point-to-Point Statistics

Figure 87 shows a point-to-point (PTP) table (from rvtrace -addrinfo -ptp). The text below introduces important concepts. Table 35 on page 377 describes the columns in detail.

In switched ethernet environments point-to-point packets remain invisible to rvtrace—except for packets addressed specifically to the rvtrace host computer. Since this fact severely limits the usefulness of reporting point-to-point statistics, rvtrace omits them from its output unless you specify the -ptp command line option.

In some switched networks, you can disable switching behavior—for example, by setting one port to diagnostic mode, or by using a diagnostic utility. This tactic can yield the full stream of point-to-point packets in a limited portion of the network; run rvtrace in that portion.

<table>
<thead>
<tr>
<th>Port</th>
<th>Address</th>
<th>SCid</th>
<th>Data</th>
<th>Bytes</th>
<th>AckR</th>
<th>Ack</th>
<th>Nak</th>
<th>Bad</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Totals</td>
<td></td>
<td>18</td>
<td>4158</td>
<td>18</td>
<td>18</td>
<td>0</td>
<td>36</td>
</tr>
<tr>
<td>20000</td>
<td>10.101.2.102</td>
<td>23001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>18</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>* 10.101.2.102</td>
<td>23002</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>* 10.101.2.249</td>
<td>20000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>23001</td>
<td>10.101.2.102</td>
<td>23001</td>
<td>10</td>
<td>2310</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>* 10.101.2.102</td>
<td>23002</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>* 10.101.2.249</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>23002</td>
<td>8</td>
<td>1848</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>* 10.101.2.102</td>
<td>23002</td>
<td>8</td>
<td>1848</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>* 10.101.2.249</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Network Total Row The first row (immediately after the table and column headings) is a network total row; the word Totals in the Address column is a visual cue. This row shows the grand total of packets related to retransmission detected on the network during the interval.
The remaining rows display more fine-grained information about those packets—grouping them by UDP service, and destination or source IP address.

**Port Subtotal Row**

The second row in Figure 87 is a port subtotal row—its columns subtotal the statistics over the subsequent destination and source rows which it governs (until the next port subtotal row).

A number in the Port column indicates the UDP service for its row, and the group of rows that follow it. A blank in this column means that the row has the same port as the row above, and is part of the same subtotal group. Notice how the pattern of numbers and blanks in the Port column visually indicates the subtotal groups.

**Destination and Source Rows**

For each IP address with point-to-point data packet activity, this table contains a destination row and a source row—always paired in that order. An * and an IP address (in the Address column) flags a row as a destination row. A blank (space characters) flags a row as a source row. The address in the destination row also applies to the source row that immediately follows it.

**Counting Packets**

This table displays each packet twice—once in a destination row, and once in a source row.

In each statistical column, the number in the port subtotal row is equal to the sum of the values in the destination rows, which is also equal to the sum of the values in the source rows.

In many networks it is possible to match the numbers in the source row for one IP address against the numbers in the destination row for another IP address. From this information you can deduce which Rendezvous daemons are exchanging point-to-point data packets and requesting retransmissions.

**Table 35  Point-to-Point Statistics—Column Headings (Sheet 1 of 3)**

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>In port subtotal rows, this column contains a UDP port number indicating the Rendezvous service for the group of rows that it begins. In destination and source rows this column is blank; the service in the nearest preceding port subtotal row governs the destination and source rows below it.</td>
</tr>
<tr>
<td>*</td>
<td>Asterisk (*) in this unlabeled column indicates a destination row. Blank in this column indicates a source row.</td>
</tr>
</tbody>
</table>
Table 35  Point-to-Point Statistics—Column Headings (Sheet 2 of 3)

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>In <em>destination rows</em> this column shows the destination IP address of retransmission request packets (that is, the Rendezvous daemon that originally sent data packets). In <em>source rows</em> this column shows the IP address from which retransmission request or rejection packets originate (that is, the Rendezvous daemon that missed receiving data packets). In <em>network total rows</em>, this column contains the word Totals.</td>
</tr>
<tr>
<td>SCid</td>
<td>Service communication ID. In <em>destination rows</em> this column differentiates the destination ID of retransmission request packets (that is, the Rendezvous daemon that originally sent data packets). In <em>source rows</em> this column is blank.</td>
</tr>
<tr>
<td>Data</td>
<td>Point-to-point data packets. This column shows the number of point-to-point data packets.</td>
</tr>
<tr>
<td>Bytes</td>
<td>Point-to-point data bytes. This column sums the number of payload bytes over the point-to-point data packets (as counted in the <em>Data</em> column).</td>
</tr>
<tr>
<td>AckR</td>
<td>Acknowledgement request packets. Sending Rendezvous daemons explicitly request positive acknowledgment for groups of point-to-point data packets. This column shows the number of packets containing acknowledgment requests for point-to-point data packets.</td>
</tr>
<tr>
<td>Ack</td>
<td>Acknowledgement packets. Receiving Rendezvous daemons explicitly acknowledge groups of point-to-point data packets upon request from sending daemons. This column shows the number of packets containing acknowledgments for point-to-point data packets.</td>
</tr>
<tr>
<td>Nak</td>
<td>Negative acknowledgement packets. Receiving Rendezvous daemons use negative acknowledgments to request retransmission of missing data point-to-point packets. This column displays the number of packets containing retransmission requests for point-to-point data packets. For more information, see Nak Diagnoses on page 381.</td>
</tr>
</tbody>
</table>
Nak Diagnoses

Nak measures the number of point-to-point packets that request retransmission of point-to-point data.

Non-zero Nak values to or from a specific address usually indicates one of these problems:

- A faulty network interface card at a specific computer.
- A faulty or overloaded network infrastructure component (for example, switching or router hardware).
- A fast sender is overwhelming a slower receiver with point-to-point packets.
- A sender on a fast network is overwhelming a network infrastructure component by sending point-to-point packets to a receiver on a slower network.

Begin by checking the specific interface card, and widen the investigation to other components until you can resolve the difficulty.

Figure 88 displays example output with this pattern.

- SCid 35 at address 10.101.2.102 is sending point-to-point data to SCid 16 at 10.101.3.249.
- The AckR column shows that 10.101.2.249 received 68 requests for acknowledgement to 10.101.2.102.
- The Nak column shows that SCid 16 at 10.101.2.249 did not receive all the packets correctly, and sent 23 NAKs back to SCid 35 at 10.101.2.102. These NAKs constitute retransmission requests for the missed point-to-point packets.
- The Ack column shows that eventually, 10.101.2.249 did receive all 68 retransmitted packets correctly, recovering from the problem.
- This particular example report does not contain sufficient information to determine the locus of the problem—it could be either at the sender or the receiver.

*Figure 88  Nak Indicates Faulty Network Card or Infrastructure Component*

<table>
<thead>
<tr>
<th>Port</th>
<th>Address</th>
<th>SCid</th>
<th>Data</th>
<th>Bytes</th>
<th>AckR</th>
<th>Ack</th>
<th>Nak</th>
<th>Bad</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1716</td>
<td>597168</td>
<td>68</td>
<td>68</td>
<td>23</td>
</tr>
<tr>
<td>Totals</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20000</td>
<td>10.101.2.102</td>
<td>22</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>10.101.2.102</td>
<td>35</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>68</td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1716</td>
<td>597168</td>
<td>68</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>*</td>
<td>10.101.2.249</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>68</td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1716</td>
<td>597168</td>
<td>68</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Subject Statistics

The subject table counts multicast and broadcast messages (not packets) and organizes statistics by Rendezvous subject name (in addition to UDP service and destination address).

Figure 89 shows a subject table (from rvtrace -addrinfo). The text below introduces important concepts. Table 36 on page 384 describes the columns in detail.

Figure 89  Multicast Subject Statistics

<table>
<thead>
<tr>
<th>Port</th>
<th>Address</th>
<th>Multicast Subject Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SCid</td>
</tr>
<tr>
<td>-----</td>
<td>----------</td>
<td>------</td>
</tr>
<tr>
<td>Totals</td>
<td></td>
<td>21</td>
</tr>
<tr>
<td>5863</td>
<td>10.101.2.255</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>10.101.2.41</td>
<td>0</td>
</tr>
<tr>
<td>20000</td>
<td>224.1.1.12</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>10.101.2.102</td>
<td>39365</td>
</tr>
<tr>
<td></td>
<td>10.101.2.102</td>
<td>39364</td>
</tr>
</tbody>
</table>

Network Total Row

The first row (immediately after the table and column headings) is a network total row; the word Totals in the Address column is a visual cue. This row shows the grand total of messages that rvtrace detected on the network during the interval. The remaining rows display more fine-grained information about those messages—grouping them by UDP service, destination address, subject name, and source address.

Subject Subtotal Groups

A character string in the Subject column indicates the Rendezvous subject name for its row, and the group of rows that follow it. A blank in this column means that the row has the same subject as the row above, and is part of the same subtotal group. Notice how the pattern of subject names and blanks in the Subject column visually indicates the subtotal groups. The visual pattern of numbers in the Port column echoes this division.

Each subject subtotal group begins with a subject row (which is also a destination row) followed by one or more source rows.

Destination and Source Rows

* flags a row as a destination row. A blank (space character) in this column flags a row as a source row.
Each destination row is the heading and subtotal for the source rows that follow it. For example, consider the destination row with foo.1 in the Subject column. The Msgs column indicates 20 multicast messages. The two subsequent source rows indicate that those 20 messages came from two sources on the host 10.101.2.102—that is, SCid 39365 sent 10 messages, while 39364 sent another 10 messages. The subtotal 20 in turn contributes to the grand total 21 in the network total row.

A subject row governs the source rows below it (until the next subject row). That is, the subject, UDP service (port), and address in the governing row apply to those source rows. Similarly, the governing row address implies either multicast or broadcast protocol, and this protocol also applies to the statistics in the source rows that it governs. (Naturally, all of this information also applies to the governing row itself.)

Statistics

In destination rows numbers in statistics columns count messages with the destination specified in the Address column.

In source rows numbers in statistics columns count messages originating from the IP address in the Address column.

In network total rows, numbers in statistics columns represent the message totals for the network during the interval.

<p>| Table 36 Subjects Statistics—Column Headings (Sheet 1 of 2) |</p>
<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>In destination subtotal rows, this column contains a UDP port number indicating the Rendezvous service for the group of rows that it begins. In source rows this column is blank; the service in the nearest preceding destination row also applies to the source row.</td>
</tr>
<tr>
<td>*</td>
<td>Asterisk (*) in this unlabeled column indicates a destination subtotal row. Blank in this column indicates a source row.</td>
</tr>
<tr>
<td>Address</td>
<td>In destination rows this column shows the destination address shared among a group of messages. It can be an IP address or a multicast group. In source rows this column shows the IP address from which group of messages originate. In network total rows, this column contains the word Totals.</td>
</tr>
</tbody>
</table>
Subject Table Diagnoses

The subject table reveals interesting information about the subject name space, and its use within the network:

- Programs that send messages in violation of the subject usage guidelines for your enterprise
- Duplicate process instances of a sending program
- Subjects that consume large portions of network capacity
rvtrace embeds an SNMP agent. You can use SNMP applications to query rvtrace statistics and trap SNMP events (as listed in Table 37).

Table 37  SNMP Objects in rvTrace (Sheet 1 of 8)

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Network Totals</strong></td>
<td></td>
</tr>
<tr>
<td>rvNetTotals</td>
<td>Grouping for all network totals.</td>
</tr>
<tr>
<td>rvTrdpMCPktTotals</td>
<td>Grouping for network totals related to multicast data packets.</td>
</tr>
<tr>
<td>rvTrdpMCDataPktTotal</td>
<td>Total multicast or broadcast data packets.</td>
</tr>
<tr>
<td>rvTrdpMCDataByteTotal</td>
<td>Total payload bytes in all multicast or broadcast data packets.</td>
</tr>
<tr>
<td>rvTrdpMCRtPktTotal</td>
<td>Total multicast or broadcast retransmission packets.</td>
</tr>
<tr>
<td>rvTrdpMCRtByteTotal</td>
<td>Total payload bytes in all multicast or broadcast retransmit packets.</td>
</tr>
<tr>
<td>rvTrdpMCNullPktTotal</td>
<td>Total null packets.</td>
</tr>
<tr>
<td></td>
<td>When a Rendezvous daemon has no data packets to transmit, it periodically</td>
</tr>
<tr>
<td></td>
<td>sends <strong>null packets</strong> to maintain continuity.</td>
</tr>
<tr>
<td>rvTrdpMCSeqGapTotal</td>
<td>Total Rendezvous packets that rvtrace missed.</td>
</tr>
<tr>
<td></td>
<td>rvtrace tracks the serial numbers of Rendezvous packets. This object</td>
</tr>
<tr>
<td></td>
<td>counts the missing packets in each sequence gap of multicast or broadcast</td>
</tr>
<tr>
<td></td>
<td>data packets. For more information, see <strong>Gaps Diagnoses</strong> on page 368.</td>
</tr>
<tr>
<td>rvTrdpMCBadPktTotal</td>
<td>Total bad multicast data packets.</td>
</tr>
<tr>
<td></td>
<td>Bad packets lack UDP checksums, or are corrupt in some other way.</td>
</tr>
</tbody>
</table>

⚠️

See Bad Packets on page 364.
### SNMP Objects in rvTrace (Sheet 2 of 8)

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rvTrdpRtPktTotals</td>
<td>Grouping for network totals related to retransmitted multicast or broadcast packets.</td>
</tr>
<tr>
<td>rvTrdpRtReqPktTotal</td>
<td>Total retransmission request packets. Each request packet counts separately, even if several request packets specify the same data packet numbers for retransmission. For example, if two daemons each request retransmission of the data packets numbered 121–125, and a third daemon requests retransmission of the data packets numbered 100–144, then this total increases by 3.</td>
</tr>
<tr>
<td>rvTrdpRtReqSeqTotal</td>
<td>Total of requested sequence numbers summed over all retransmission requests. Each retransmission request packet can solicit one or more data packets for retransmission. This object sums the number of data packets for which retransmission is requested. If some data packets are requested several times, each data packet counts separately each time it is requested. For example, if three daemons request retransmission of data packets numbered 121–125, then this total increases by 15. For more information, see Diagnoses on page 372.</td>
</tr>
<tr>
<td>rvTrdpRtRejPktTotal</td>
<td>Total retransmission rejections. Although Rendezvous daemons comply with retransmission requests whenever possible, sometimes the requested packets are no longer available. This object counts the number of packets that contain retransmission rejections.</td>
</tr>
<tr>
<td>rvTrdpRtBadPktTotal</td>
<td>Total bad multicast retransmission packets. Bad packets lack UDP checksums, or are corrupt in some other way. See Bad Packets on page 364.</td>
</tr>
<tr>
<td>rvPtpPktTotals</td>
<td>Grouping for network totals related to point-to-point packets.</td>
</tr>
<tr>
<td>rvPtpDataPktTotal</td>
<td>Total point-to-point data packets.</td>
</tr>
<tr>
<td>rvPtpDataByteTotal</td>
<td>Total payload bytes in all point-to-point data packets.</td>
</tr>
</tbody>
</table>
### Table 37  SNMP Objects in rvTrace (Sheet 3 of 8)

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rvPtpAckRPktTotal</td>
<td>Total acknowledgement request packets. Sending Rendezvous daemons explicitly request positive acknowledgment for groups of point-to-point data packets. This object counts the number of packets containing acknowledgment requests for point-to-point data packets.</td>
</tr>
<tr>
<td>rvPtpAckPktTotal</td>
<td>Total acknowledgement packets. Receiving Rendezvous daemons explicitly acknowledge groups of point-to-point data packets upon request from sending daemons. This object counts the number of packets containing acknowledgments for point-to-point data packets.</td>
</tr>
<tr>
<td>rvPtpNakPktTotal</td>
<td>Total negative acknowledgement packets. Receiving Rendezvous daemons use negative acknowledgments (NAKs) to request retransmission of missing data point-to-point packets. This column displays the number of packets containing retransmission requests for point-to-point data packets. For more information, see Nak Diagnoses on page 381.</td>
</tr>
<tr>
<td>rvPtpBadPktTotal</td>
<td>Total bad point-to-point packets. Bad packets lack UDP checksums, or are corrupt in some other way. [\text{⚠️}] See Bad Packets on page 364.</td>
</tr>
<tr>
<td>rvSubjectTotals</td>
<td>Grouping for network totals related to Rendezvous subjects.</td>
</tr>
<tr>
<td>rvSubjMsgTotal</td>
<td>Total number of messages (summed over all subjects).</td>
</tr>
<tr>
<td>rvSubjByteTotal</td>
<td>Total number of payload bytes (summed over all subjects).</td>
</tr>
</tbody>
</table>

### Thresholds

<table>
<thead>
<tr>
<th>Threshold Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rvTrdpRtThreshold</td>
<td>Threshold for retransmit trap. When the ratio $\text{rvTrdpMCRtByteTotal}/\text{rvTrdpMCDDataByteTotal}$ exceeds this threshold (expressed as a percentage), rvtrace sets the SNMP trap $\text{rvTrdpNotifyRetransmit}$. The default threshold is 2%.</td>
</tr>
</tbody>
</table>
Threshold for bad packet trap.

When the number of bad packets during an interval exceeds this threshold, *rvtrace* sets the SNMP trap *rvNotifyBadPkts*. Bad packets of any type count toward this threshold.

The default threshold is zero—that is, trap even for one bad packet.

Multicast Host Snapshot

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>rvTrdpMCHostTable</em></td>
<td>Table of multicast statistics, with one entry for each triple of sending host (<em>rvTrdpMCHostAddr</em>), service port (<em>rvTrdpMCPort</em>), and destination address (<em>rvTrdpMCDestAddr</em>).</td>
</tr>
<tr>
<td><em>rvTrdpMCPort</em></td>
<td>Service port of packets counted in the table entry.</td>
</tr>
<tr>
<td><em>rvTrdpMCHostAddr</em></td>
<td>IP address of the host that is the <em>source</em> of multicast or broadcast packets counted in the table entry.</td>
</tr>
<tr>
<td><em>rvTrdpMCDestAddr</em></td>
<td>Destination address of packets counted in the table entry. The destination can be a multicast address or an IP (broadcast) address.</td>
</tr>
<tr>
<td><em>rvTrdpMCHostDataPkts</em></td>
<td>Number of data packets for this triple.</td>
</tr>
<tr>
<td><em>rvTrdpMCHostDataBytes</em></td>
<td>Number of payload bytes in data packets for this triple.</td>
</tr>
<tr>
<td><em>rvTrdpMCHostRtPkts</em></td>
<td>Number of retransmission packets for this triple.</td>
</tr>
<tr>
<td><em>rvTrdpMCHostRtBytes</em></td>
<td>Number of payload bytes in retransmission packets for this triple.</td>
</tr>
<tr>
<td><em>rvTrdpMCHostNullPkts</em></td>
<td>Number of null packets for this triple.</td>
</tr>
<tr>
<td><em>rvTrdpMCHostSeqGaps</em></td>
<td>Number of requested sequence numbers summed over all retransmission requests for this triple.</td>
</tr>
</tbody>
</table>

Each retransmission request packet can solicit one or more data packets for retransmission. This object sums the number of data packets for which retransmission is requested.

For more information, see Diagnoses on page 372.

*rvTrdpMCHostBadPkts* Number of bad multicast or broadcast packets for this triple.
Table 37  SNMP Objects in rvTrace (Sheet 5 of 8)

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Multicast Retransmission Host Snapshot</strong></td>
<td></td>
</tr>
<tr>
<td>rvTrdpRtHostTable</td>
<td>Table of multicast retransmission statistics, with one entry for each pair of host (rvTrdpRtHostAddr) and service port (rvTrdpRtPort). Entries in this table count several types of packets related to the retransmission request protocol for multicast or broadcast data.</td>
</tr>
<tr>
<td>rvTrdpRtPort</td>
<td>Service port of packets counted in the table entry.</td>
</tr>
<tr>
<td>rvTrdpRtHostAddr</td>
<td>IP address of the host that is the source or destination of packets counted in the table entry.</td>
</tr>
<tr>
<td>rvTrdpRtReqPktsSrc</td>
<td>Number of retransmission request packets sent by the host (that is, the host is the source of the request packet). These packets indicate that the receiving Rendezvous daemon on the host missed inbound data packets, and has requested retransmission. Each request packet counts separately, even if several request packets specify the same data packet numbers for retransmission.</td>
</tr>
<tr>
<td>rvTrdpRtReqSeqSrc</td>
<td>Number of data packet sequence numbers requested by the host (that is, the host is the source of the request). Each retransmission request packet can solicit one or more data packets for retransmission. This item counts the number of data packets for which the host requested retransmission, summing them over the request packets (as counted by rvTrdpRtReqPktsSrc). If the host requests some data packets several times, each data packet counts separately each time the host requests it. For more information, see Diagnoses on page 372.</td>
</tr>
<tr>
<td>rvTrdpRtRejPktsSrc</td>
<td>Number of retransmission rejection notices sent by the host (that is, the host is the source of the rejection packet). Although Rendezvous daemons comply with retransmission requests whenever possible, sometimes the requested data packets are no longer available.</td>
</tr>
</tbody>
</table>
Table 37  SNMP Objects in rvTrace (Sheet 6 of 8)

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rvTrdpRtBadPktsSrc</td>
<td>Number of bad retransmission request packets sent by the host (that is, the host is the source of the bad packet). Bad packets lack UDP checksums, or are corrupt in some other way.</td>
</tr>
<tr>
<td></td>
<td>See Bad Packets on page 364.</td>
</tr>
<tr>
<td>rvTrdpRtReqPktsDest</td>
<td>Number of retransmission request packets indicating the host as destination. These packets indicate that receiving Rendezvous daemons missed data, and requested retransmission from the host.</td>
</tr>
<tr>
<td></td>
<td>Each request packet counts separately, even if several request packets specify the same data packet numbers for retransmission.</td>
</tr>
<tr>
<td>rvTrdpRtReqSeqDest</td>
<td>Number of data packet sequence numbers requested from the host. Each retransmission request packet can solicit one or more data packets for retransmission. This item counts the number of data packets requested, summing them over the request packets (as counted by rvTrdpRtReqPktsDest).</td>
</tr>
<tr>
<td></td>
<td>If several daemons request a data packet several times, each request counts separately. For more information, see Diagnoses on page 372.</td>
</tr>
<tr>
<td>rvTrdpRtRejPktsDest</td>
<td>Number of retransmission rejection notices indicating the host as destination. Although Rendezvous daemons comply with retransmission requests whenever possible, sometimes the requested data packets are no longer available.</td>
</tr>
<tr>
<td>rvTrdpRtBadPktsDest</td>
<td>Number of bad retransmission request packets indicating the host as destination. Bad packets lack UDP checksums, or are corrupt in some other way.</td>
</tr>
<tr>
<td></td>
<td>See Bad Packets on page 364.</td>
</tr>
</tbody>
</table>
### Table 37  SNMP Objects in rvTrace (Sheet 7 of 8)

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rvPtpHostTable</td>
<td>Table of point-to-point statistics, with one entry for each pair of host (rvPtpHostAddr) and service port (rvPtpPort)</td>
</tr>
<tr>
<td>rvPtpPort</td>
<td>Service port of packets counted in the table entry.</td>
</tr>
<tr>
<td>rvPtpHostAddr</td>
<td>IP address of the host that is the source or destination of packets counted in the table entry.</td>
</tr>
<tr>
<td>rvPtpPktsSrc</td>
<td>Number of point-to-point data packets sent by the host (that is, the host is the source of the data packet).</td>
</tr>
<tr>
<td>rvPtpBytesSrc</td>
<td>Number of payload bytes summed over point-to-point data packets sent by the host (that is, the host is the source of the data).</td>
</tr>
<tr>
<td>rvPtpAckRPktsSrc</td>
<td>Number of packets requesting acknowledgement sent by the host (that is, the host is the source of the acknowledgement request packet).</td>
</tr>
<tr>
<td>rvPtpAckPktsSrc</td>
<td>Number of ACK packets sent by the host.</td>
</tr>
<tr>
<td>rvPtpNakPktsSrc</td>
<td>Number of NAK packets sent by the host.</td>
</tr>
<tr>
<td>rvPtpBadPktsSrc</td>
<td>Number of bad packets sent by the host.</td>
</tr>
<tr>
<td>rvPtpPktsDest</td>
<td>Number of point-to-point data packets indicating the host as destination.</td>
</tr>
<tr>
<td>rvPtpBytesDest</td>
<td>Number of payload bytes summed over data packets indicating the host as destination.</td>
</tr>
<tr>
<td>rvPtpAckRPktsDest</td>
<td>Number of acknowledgement request packets indicating the host as destination.</td>
</tr>
<tr>
<td>rvPtpAckPktsDest</td>
<td>Number of ACK packets indicating the host as destination.</td>
</tr>
<tr>
<td>rvPtpNakPktsDest</td>
<td>Number of NAK packets indicating the host as destination.</td>
</tr>
<tr>
<td>rvPtpBadPktsDest</td>
<td>Number of bad packets indicating the host as destination.</td>
</tr>
</tbody>
</table>
SNMP Agent Configuration

You can configure the agent and traps in the file `rvtracesnmp.conf`.

- On UNIX platforms, the default location of the configuration file is
  
  `/usr/local/etc/snmp/rvtracesnmp.conf`

- On Windows platforms, the default location of the configuration file is

  `/usr/rvtracesnmp.conf`

Environment variables can instruct SNMP to search for the configuration file in a different location; for details, see this web page:

  http://net-snmp.sourceforge.net/docs/man/snmp_config.html

Table 38 on page 394 presents the configuration parameters. To change the default values of these parameters, create this file, and configure it appropriately.
### Table 38  SNMP Configuration Parameters (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rocommunity community [location [oid]]</td>
<td>You must specify at least one rocommunity and at least one rwcommunity. You may specify more than one of each. Each such line controls SNMP access from a set of SNMP managing systems, granting either read-only or read-write access to a set of rvtrace SNMP objects.</td>
</tr>
<tr>
<td>rwcommunity community [location [oid]]</td>
<td></td>
</tr>
</tbody>
</table>

- `community` (required) must be a sequence of alphanumeric characters (without quote marks, and without intervening spaces). This argument restricts SNMP access, allowing queries only from managing systems that present an identical string.

- `location` (optional) further restricts access to managing systems on a designated host computer or network. You may supply an argument in these three forms:
  - A hostname or IP address restricts access to managing systems on the specified host computer.
  - An IP network address with mask bits restricts access to managing systems within the specified IP network, as determined by matching the specified number of mask bits. For example, 10.1.1.0/24 allows any computer with an address matching 10.1.1.*.
  - `default` (a literal) does not restrict access by location; you may use it as a place-holder when you supply an `oid` argument.

- `oid` (optional) is an object identifier, which determines an object or subtree of the rvtrace MIB tree. You may supply a numeric OID (for example, .1.3.6.1.4.1.2000.7.1.100) or a symbolic OID (for example, `rvTrdpMCPktTotals`).
When present, the SNMP agent accepts requests and queries on this port.

Port can have either of two forms:
- 1234 (represents a UDP port)
- tcp:1234 (represents a TCP port)

When absent, the default port is 161.

When this line is present, the SNMP agent enables trap objects, and sends trap notification to the SNMP trap daemon on host.

When the optional community property is present, the trap daemon must present an identical property before it can receive trap notifications. The community property must be a sequence of alphanumeric characters (without quote marks, and without intervening spaces).

When the optional port property is present, the agent sends notifications on this port. The port can have the same two forms as with agentaddress, above. When absent, the default port is 162.

To send notification to several trap daemons, specify a separate informsink line for each destination.

When informsink is absent, the agent disables trap objects.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agentaddress</td>
<td>When present, the SNMP agent accepts requests and queries on this port. Port can have either of two forms:</td>
</tr>
<tr>
<td>port</td>
<td>• 1234 (represents a UDP port)</td>
</tr>
<tr>
<td></td>
<td>• tcp:1234 (represents a TCP port)</td>
</tr>
<tr>
<td></td>
<td>When absent, the default port is 161.</td>
</tr>
<tr>
<td>informsink</td>
<td>When this line is present, the SNMP agent enables trap objects, and sends trap notification to the SNMP trap daemon on host.</td>
</tr>
<tr>
<td>host [community] [port]</td>
<td>When the optional community property is present, the trap daemon must present an identical property before it can receive trap notifications. The community property must be a sequence of alphanumeric characters (without quote marks, and without intervening spaces).</td>
</tr>
<tr>
<td></td>
<td>When the optional port property is present, the agent sends notifications on this port. The port can have the same two forms as with agentaddress, above. When absent, the default port is 162.</td>
</tr>
<tr>
<td></td>
<td>To send notification to several trap daemons, specify a separate informsink line for each destination.</td>
</tr>
<tr>
<td></td>
<td>When informsink is absent, the agent disables trap objects.</td>
</tr>
</tbody>
</table>
Perl programs can call Rendezvous API functions directly using a Perl 5 loadable module called `Tibrv`, which extends the Perl language.

Topics

- Features and Benefits, page 398
- Installing the Perl 5 Interface, page 399
- Using the Perl 5 Interface, page 400
- Perl Example Programs, page 401
Features and Benefits

Tibrv presents the C API. All API details (for example, function names, parameter lists, types, return codes) are the same as for C programs, so Perl programmers can use it to:

- Prototype Rendezvous programs in Perl for later translation into C.
- Write Rendezvous programs in Perl.

System administrators often choose Perl for data organization tasks. Combining Perl with Rendezvous software yields a powerful tool for:

- Gathering and organizing information about the operation of distributed systems.
- Administering physically distant computers across a network.
Installing the Perl 5 Interface

For installation instructions, see the Rendezvous Perl README file:

rv_installation_dir/src/perl/README
Using the Perl 5 Interface

Be sure that the library path variable is set properly; for instructions, see the README file.

Place this module reference near the top of each Perl program file that calls Rendezvous API functions:

```perl
use Tibrv;
```

If the program uses Rendezvous fault tolerance features, add this module reference:

```perl
use TibrvFt;
```

If the program uses Rendezvous certified message delivery features or distributed queues, add this module reference:

```perl
use TibrvCm;
```
Perl Example Programs

The Rendezvous example directory (/src/examples/perl/) contains Perl example programs.
Chapter 16  **Certified Message Delivery**

Topics

- Forward RVCVM Administrative Messages across Network Boundaries, page 404
- Ledger File Location, page 405
Forward RVCMS Administrative Messages across Network Boundaries

Rendezvous certified message delivery software depends on administrative announcements (as well as point-to-point messages) between CM transports. These messages travel freely within a single network segment. However, if your network consists of several segments connected by Rendezvous routing daemons, then you must instruct the routing daemons to forward the subjects in Table 39.

Routing daemons must forward these subjects in both directions—import and export.

Similarly, if clients in your network use SSL to connect to rvsd or rvsrd, you must configure the secure daemon to authorize the subjects in Table 39.

Table 39 Critical Subjects for Certified Delivery

<table>
<thead>
<tr>
<th>Subject</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>RVCMS.</em>&gt;</td>
<td>Rendezvous certified delivery software uses administrative messages with these subjects. Routing daemons must forward these subjects in both directions.</td>
</tr>
</tbody>
</table>
Ledger File Location

The ledger file must reside on the same host computer as the program that uses it. **Do not** use network-mounted storage for ledger files.

Remember that certified message delivery protects against component or network failure. Placing ledger files across a network (for example, on a separate file server) introduces a new dependency on the network, leaving components vulnerable to network failures.
Chapter 17  Fault Tolerance

Topics

- Network Placement, page 408
- Forward Fault Tolerance Messages across Network Boundaries, page 409
Network Placement

When you deploy a fault-tolerant application, it is important to distribute the member processes appropriately across computers and across network segments. Independence increases the effectiveness of redundant processes. For details, see Distribute Members on page 222 in TIBCO Rendezvous Concepts.

It is also important to use files in a way that does not jeopardize fault tolerance. For guidelines, see Member File Access on page 223 in TIBCO Rendezvous Concepts.
Forward Fault Tolerance Messages across Network Boundaries

Rendezvous fault tolerance software depends on messages between group members, and on some Rendezvous system advisory messages. These messages travel freely within a single network segment. However, if your network consists of several segments connected by Rendezvous routing daemons, then you must instruct the routing daemons to forward the subjects in Table 40. Routing daemons must forward all these subjects in both directions—import and export.

Similarly, if clients in your network use SSL to connect to rvsd or rvsrd, you must configure the secure daemon to authorize the subjects in Table 40.

Table 40  Critical Subjects for Fault Tolerance

<table>
<thead>
<tr>
<th>Subject</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_RVFT.*.group_name</td>
<td>Rendezvous fault tolerance software uses messages with these subjects to communicate among group members. Routing daemons must forward these subjects in both directions.</td>
</tr>
</tbody>
</table>
Chapter 18  Distributed Queues

Topics

- Forward Administrative Messages across Network Boundaries, page 412
Forward Administrative Messages across Network Boundaries

Rendezvous distributed queue software depends on administrative announcements (as well as point-to-point messages) between distributed member transports.

These messages travel freely within a single network segment. However, if your network consists of several segments connected by Rendezvous routing daemons, then you must instruct the routing daemons to forward the subjects in Table 41.

We do not recommend sending messages across network boundaries to a distributed queue, nor distributing queue members across network boundaries. However, when crossing network boundaries in either of these ways, you must configure the Rendezvous routing daemons to exchange _RVCM and _RVCMQ administrative messages.

Routing daemons must forward the subjects in Table 41 in both directions—import and export.

Similarly, if clients in your network use SSL to connect to rvsd or rvsrd, you must configure the secure daemon to authorize the subjects in Table 41.

Table 41 Critical Subjects for Distributed Queues (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Subject</th>
<th>Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_RVCMQ</td>
<td>Rendezvous distributed queue software uses administrative messages with these subjects.</td>
<td>Whenever potential scheduler members run in one network, and potential listener members of the same distributed queue run in a second network, then routing daemons must forward these subjects in both directions between the two networks. Similarly, whenever potential listener members of the same distributed queue run in two separate networks, then routing daemons must forward these subjects in both directions between the two networks.</td>
</tr>
<tr>
<td>_RVCM</td>
<td>Rendezvous distributed queue software uses administrative messages with these subjects.</td>
<td>Whenever a process in one network sends task messages to potential scheduler members in a second network, then routing daemons must forward these subjects in both directions between the two networks.</td>
</tr>
</tbody>
</table>
Table 41  Critical Subjects for Distributed Queues (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Subject</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_RVFT. &gt;</td>
<td>Rendezvous distributed queue software uses administrative messages with these subjects.</td>
</tr>
<tr>
<td></td>
<td>Whenever <em>potential scheduler members</em> of the same distributed queue run in two separate networks, then routing daemons must forward these subjects in both directions between the two networks.</td>
</tr>
</tbody>
</table>
Appendix A  Windows Services

You can run Rendezvous components automatically by registering them as Microsoft Windows services. This appendix describes two utility programs to arrange registration.

Topics

- rvntscfg, page 416
- rvntsreg, page 417

See Also

Register Windows Services on page 8
rvntscfg

Utility

Purpose  Graphical user interface to configure Rendezvous components as Windows services.

Restrictions  You must have administrator privileges to change the Windows registry.

Remarks  Locate this utility program as an executable file in the Rendezvous bin\ directory.

Rendezvous components belong to one of two categories:
• Base communications components. This category consists of rvd and rvrd—the two components that enable network communications.
• Dependent components. All other components are in this category—they depend on the presence of a base communications component.

Before you can start one of these programs as a service, one of the base communications components must already be running as a service.

Register  To register a component as a Windows service:
1. Start rvntscfg.exe.
2. Select the Rendezvous component.
3. Specify manual or automatic start. In most cases, the appropriate choice is automatic—that is, start the component automatically when starting Windows.
4. Check the path to the Rendezvous bin\ directory (or the correct location of the component).
5. Check the arguments to the program.
   In particular, ensure that this field correctly specifies the store file and HTTP access.
6. Click the Install button to register, or the Update button to change an existing registration.
7. When prompted to start the new service, click Yes.

Remove  To unregister a service:
1. Start rvntscfg.exe.
2. Select the Rendezvous component.
3. Click the Remove button to unregister the service.
**rvntsreg**

**Utility**

**Purpose**
Register Rendezvous components as Windows services.

**Restrictions**
You must have administrator privileges to change the Windows registry.

**Remarks**
Locate this utility program as an executable file in the Rendezvous bin\ directory.

The `rvntscfg` utility achieves the same end as this program, adding an intuitive graphical user interface.

**Register**
To register a component as a Windows service, run the utility with this command line:

```
rvntsreg /i service_name ctrl_dir daemon_dir arguments
```

For example:

```
rvntsreg /i rvrd c:\tibco\tibrv\8.3.0\bin c:\tibco\tibrv\8.3.0\bin "-store C:\tibco\tibrv\localadm\rvrd.adm"
```

**Table 42  Critical Subjects for Fault Tolerance**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>service_name</code></td>
<td>Name of the Windows service.</td>
</tr>
<tr>
<td><code>ctrl_dir</code></td>
<td>Directory path to the controller executable <code>rvntsctl.exe</code>. Usually this file is in the same location as the Rendezvous daemon, though you may copy it to another location (for example, the Windows directory).</td>
</tr>
<tr>
<td><code>daemon_dir</code></td>
<td>Directory path to the Rendezvous daemon executable.</td>
</tr>
<tr>
<td><code>arguments</code></td>
<td>The controller executable passes these command line arguments to the Rendezvous daemon.</td>
</tr>
<tr>
<td></td>
<td>Supply the arguments as a quoted string. If an argument is a file or directory name that contains space characters, you must enclose that name within escaped quotes (`).</td>
</tr>
</tbody>
</table>

**Remove**
To unregister a service, run the utility with this command line:

```
rvntsreg /r service_name
```

**Command Summary**
To view a command line summary, run the utility with this command line:

```
rvntsreg
```

**See Also**
`rvntscfg` on page 416
Appendix B  Store Files

Many daemons associated with TIBCO Rendezvous use store files to maintain configuration or state. This appendix presents details about store files that are common to all daemons.

Topics

- Locking, page 420
Daemons access store files using a cooperative file locking regimen, which guarantees unique sequential access when two or more daemon processes attempt to share the same store file. This guarantee depends on the daemon processes’ adherence to the locking regimen.

Since non-Rendezvous processes do not adhere to the locking regimen, they can cause store file corruption—for example, by replacing the store file at inappropriate times.

To prevent store file corruption, ensure that non-Rendezvous processes do not replace or modify a store file while any instance of its daemon is running. For example, avoid replacing a store file with a saved backup version while the daemon is running.

See Also

rvrd on page 124
rvsd on page 180
rvsrd on page 184
rvrad on page 262
rva on page 266
rvcache on page 291

RVDM, see Storage on page 258
Index

Numerics

64-bit daemon 49

A

accept any as neighbor, rvrd 92
Ack column, rvtrace 378, 379, 380
AckR column, rvtrace 378, 379, 380
active neighbor, rvrd 91
address
  Address column, rvtrace 367, 371, 378, 379, 380, 384
  filter, rvtrace 355
  information, rvtrace 358
administration interface, browser 32
administrator name and password 145, 195
administrator’s checklist 1
advisory message
  See also, the book TIBCO Rendezvous Concepts
  and, boolean filters, rvtrace 361
applet, Java 274
asymmetric multicast, RVDM 47, 130, 183, 187, 218
authorization, secure daemon 173
auto, rvperfm parameter 308
automatic mode, rvperf 300, 301, 308

B

backlog, routing 120
Bad column, rvtrace 368, 372, 381
barring remote connections 29

batch
  interval, rvlat parameter 336
  interval, rvperfm parameter 309
  size, rvlat parameter 336
  size, rvperfm parameter 309
  batch, rvperf 299
binary executable files 4
boolean operators in filters, rvtrace 361
border routing 110
broadcast filter, rvtrace 361
browser administration interface 32
Bytes column, rvtrace 367, 378, 379, 380, 385
  usage 362

C

cache 279–296
  cache modes, store vs memory-only 289
capture file, rvtrace 352
CA-signed certificate 175, 176
  certificate
    CA-signed 175, 176
    security factor 175
certified delivery
  routing daemons 403
  rvperf 316, 322
  rvperfm parameters 310
  rvperf parameters 312
checklist for administrators 1
checksum 6
cm, See certified delivery
cm-sync, rvperf parameter 310, 313
collision, rvperfm 307
core dump file, security factor 175
current value cache 279–296
customer support xxxi
D

daemon list, RVDM 228
daemon manager, RVDM 209
daemon parameter 18, 28
   rvlat 334
   rvperf 307
   rvperf 312
daemon utility script 49
daemon, rvd 41–74
data capture file, rvtrace 352
Data column, rvtrace 367, 378, 379, 380
   usage 362
data loss advisory, rvperf 302
datapoints, rvlat parameter 337
deep merge (rvcache) 287
default port and service numbers 32
destination filter, rvtrace 356
distributed queues and routing daemons 411
dividing large messages, rvperf 323
duplicate messages from rvcache 283

E

elapsed time, rvperf 317
errors 359
execution search path 4

F

fault tolerance
   rvcache 286
   rvrd 94
   rvrd serving FT programs 407
feedback source and destination, rvtrace 372
file descriptor
   inheritance 9
   limit 9

filter
   details, rvtrace 360
   parameter, rvtrace 356
   pcap, rvtrace 350
firewalls
   rva 274
   rvrd 106
fixed subject interest, rvrd (obsolete) 88
flush time, rvperf 317
-foreground, rvtrace parameter 359

G

Gaps column, rvtrace 368
diagnosing 368
gating, subject
   rva 266
   rvrd 85

H

h (help)
   rvlat parameter 335
   rvperf parameter 310
   rvperf 313
   rvtrace parameter 359
hardware, gauging capabilities, rvperf 319
host filter, rvtrace 360
hostmsgs, rvtrace parameter 358
HP Tru64 UNIX, -reuse-port 52
HTTP, default ports 32

I

idle
   rva 269
   rvcache 293
   rvrd 121
inactive daemon, RVDM 228
inbox, rvperfm parameter 308
independent routes 98
input file, rvtrace parameter 355
installation, See the book TIBCO Rendezvous Installation
  interface
    network, rvtrace 347
    rvtrace parameter 354
interpreting the report
  rvperf 314
  rvtrace 362
interval between batches, rvlat
  interval parameter 336
interval between batches, rvperf 299
  interval parameter 309
interval, update, rvtrace parameter 357
IP filter, rvtrace 361
IPC sockets 30
IPM
  performance measuring tools 341
  transport batch size 49

J
Java applet 274
JConsole and RVDM 259
JMX and RVDM 259

L
latency 328
lead wildcards, excluding 45, 127
ledger file, rvperf 322
  rvperfm parameter 310
  rvperfs parameter 312
license 11–15
  installing 3
  purchasing 15
  ticket file 13
limitations
  rvtrace 347
limits
  performance assessment, warning 324
  -listen parameter 28
  load, assessing, rvtrace 362
  local network, rvrd 84
  local port, rvrd 89
  -logfile, rvtrace parameter 358
  logging, rvrd 122
  -log-max-rotations, -log-max-size, rvtrace
    parameters 359

M
managed daemon, RVDM 210
  utility script 49
maximum backlog, routing 120
maximum throughput, rvperf 301
maximum transfer rate, rvperf 319
memory-only mode (rvcache) 289
merge (rvcache) 287
message latency 328
message size
  rvlat 336
  rvperf 309
    very large 323
messages
  rvlat parameter 335
  rvperfm parameter 308
  metered protocol 303
mode, rvperf 300
  automatic, operational narrative 301
Msgs column, rvtrace 385
multicast addressing 25
multicast filter, rvtrace 361
multicast, disabling 45, 127

N
Nak column, rvtrace 378, 379, 380
  usage 381
navigation panel 189
negative acknowledgement 303
neighbor, rvrd 89
network filter, rvtrace 360
network interface (-i), rvtrace parameter 354
network parameter 18, 23, 23
  multicast addressing 25
  rvlat 334
  rvperf 307
  rvperfs 312
network stress, warning, rvperf 318
-no-display, rvtrace parameter 357
-no-lead-wc, rvd option 45, 127
no-mcast, rvtrace parameter 358
-no-multicast 45, 127
non-vectored
  rvperfm parameter 308
  rvperfs parameter 312
no-subjects, rvtrace parameter 358
not, boolean filters, rvtrace 361
Null column, rvtrace 367

O
open file limit 9
optimal sustained receive rate, rvperf 319
or, boolean filters, rvtrace 361
output file, rvtrace parameter 357

P
packet
  capture file, rvtrace 352, 355, 357
  checksum 6
  filtering, rvtrace 350, 360
passive monitor, rvtrace 349
passive neighbor, rvrd 91
password, administrator 145, 195
password, security factor 175
path 4
pause between batches, rvperf 299
payload data, rvperf 299
pcap, rvtrace 350
performance assessment tool (rvperf) 297–325
  principles of operation 299
performance effects, rvtrace, warning 349
performance obstacles, locating, rvperf 325
Perl 397–401
PGM and TRDP, bridging networks 108
PGM default network 23
physically secure environment 175
platform support, rvtrace 348
point-to-point messages 308
point-to-point, rvtrace parameter 358
Port column, rvtrace 366, 370, 377, 384
port filter, rvtrace 356, 361
port number, UDP or PGM 21
  defaults 32
port, reuse 52
positive acknowledgement 303
POSIX local IPC sockets 30
prerequisites, rvtrace 347
process identification, rvperf 316
prohibiting remote connections 29
promiscuous mode, rvtrace 349, 359
protocol 303
protocol filter, rvtrace 361
protocol monitor 345–385
ptp, rvtrace parameter 358

R
R column, rvtrace 368
rate
  optimal, rvperf 319
  rvtrace parameter 358
raw storage device, as rvcache store file 291
Rbytes column, rvtrace 367
Rcolumn, rvtrace 368
read packets from file, rvtrace 355
register Windows service 8
relay agent 261–264
-reliability (command line parameter) 36
raw storage device, as rvcache store file 291
read packets from file, rvtrace 355
register Windows service 8
relay agent 261–264
-reliability (command line parameter) 36
TIBCO Rendezvous Administration
reliability interval 34
reliability, rvtrace column 368
remote port, rvrd 89
Rendezvous agent 265–277
Rendezvous daemon 41–74
Rendezvous routing daemon 75–166
replace (rvcache) 287
report
  interpreting rvperf 314
  interpreting rvtrace 362
request reliability 36
resource inheritance 9
retention time and reliability 34
retransmission control 46, 50
retransmit diagnoses, rvtrace 372
-reuse-port 52
root certificate 175
router name
  See also, Routing table entry
routing daemon 75–166
  and certified delivery 403
  and distributed queues 411
  with fault-tolerant applications 409
  with RVDM 231
routing table entry 83
Rrej column, rvtrace 371
Rreq column, rvtrace 371
Rseq column, rvtrace 371
  usage 372
run length
  rvlat 335
  rvperf 308
rva 265–277
  command summary 266
  idle 269
rvcache 279–296
  command summary 291
  fault tolerance 286
  idle 293
  with rvrd 285
_RVCM.> (administrative subjects) 403, 412
_RVCMQ.> (administrative subjects) 412
rvd 41–74
  command summary 42
RVDM 209
  asymmetric multicast 47, 130, 183, 187, 218
  daemon list 228
  JConsole 259
  JMX MBean 259
rvdm server
  command summary 233
-rvdm, option to manage daemons 47, 129, 183, 187
_RVDM.> (administrative subjects) 231
_RVFT.> (administrative subjects) 409, 413
rvlat 327
  rvlat command 333
rvntreg 8
rvperf 297–325
  rvperf command 306
  rvperfs command 311
rvrad 261–264
  command summary 262
rvrd 75–166
  accept any as neighbor 92
  active neighbor 91
  command summary 124
  fault tolerance 94
  idle 121
  independent routes 98
  logging 122
  maximum backlog 120
  passive neighbor 91
  seek neighbor with any name 92
  topology errors 103
  with fault-tolerant applications 409
  with rvcache 285
  with RVDM 231
rvsd 167
  command line 180
rvsrd 167
  command line 184
rvtrace 345–385
  command reference 354
RXC 50
S

cample_rate, rvlat 335
SCid column, rvtrace 367, 371, 378, 379, 380, 385
search path 4
secondary effects, rvperf 324
secure daemon 167
   differences from non-secure behavior 178
security factors 175
seek neighbor with any name, rvrd 92
self-signed certificate 175, 176
sequence gap, rvtrace 368
service communication ID, rvtrace 367, 371, 378, 379, 380, 385
service port, reuse 52
service, UDP or PGM
defaults 32
defining services 5
parameter 18, 20
rvlat 334
rvperfm 307
rvperfs 311
service, Windows 8
shallow merge (rvcache) 287
single mode, rvperf 300, 308
size
   batch, rvperf 299
message
   rvlat 336
   rvperf 309
SNMP objects in rvtrace 386
-snmp, rvtrace parameter 359
socket inheritance 9
socket limit 9
source filter, rvtrace 355
spikes, rvlat parameter 337
store file, security factor 175
store mode (rvcache) 289
subject
   rvperfm parameter 308
   rvperfs parameter 312
Subject column, rvtrace 385
subject gating
   rva 266
   rvrd 85

subject interest, rvrd 80
subject map, RVDM 214
subjects, administrative 231, 409
support, contacting xxxi
switched networks, rvtrace 347
sync ledger file, rvperf 310, 313

T

TCP, default ports 32
technical support xxxi
terse
   rvlat parameter 337
   rvperfm parameter 308
throughput, maximum, rvperf 301
TIBCO_HOME xxviii
tibrv.tkt (license ticket file) 13
time, rvlat parameter 336
too many open files (OS error) 9
transport batch size, IPM 49
TRDP and PGM, bridging networks 108
Tru64 UNIX, -reuse-port 52
trust chain, certificate 175, 176

U

UDP filter, rvtrace 361
UDP mulitcast buffer size 49
UDP or PGM, default services 32
UNIX domain sockets (IPC) 30
unmetered protocol 303
update interval, rvtrace parameter 357
user, secure daemon 357
utility script 49

V

vectored, rvlat parameter 335
very large messages, rvperf 323
W

WAN, rvperf 321
warning
  limits of performance assessment 324
  network stress, rvperf 318
  performance effects, rvtrace 349
  security factors 175
web site 274
wide-area network, rvperf 321
wide-area routing daemon (rvrd) 75–166
wildcard
  excluding subjects with lead wildcards 45, 127
  importing and export in rvrd 86, 87
  publishing and RVDM 245
Windows, register service 8
-w-max-size, -w-max-rotations, rvtrace
  parameters 357
write packets to file, rvtrace 357